CS725 : Foundations of Machine learning - Lecture Notes

Ajay Nagesh

Contents

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 2: Introduction

Instructor: *Ganesh Ramakrishnan* — Date: *26/07/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

1 Basic notions and Version Space

1.1 ML : Definition

Definition (from Tom Mitchell's book): *A computer program is said to learn from experience E w.r.t some set of tasks T and performance measure P, if its performance at T improves with E as measured by P.*

Consider the sample dataset `iris.arff`. Given a set of observations of Iris flowers (like sepal length, width), our goal is to predict whether a given flower is Iris-Setosa or Iris-Versicor ...

Mapping the definition to the Iris flower-type prediction problem:

- E: Observations on Iris flowers (sepal length, width, ...)
- T: Identify the type of Iris flower
- P: Accuracy

Consider another dataset `soybean.arff`. The mapping is as follows:

- E: Observations on the soybean crop (date, plant-stand, ...)
- T: Given that the soybean is diseased, to predict the type of disease that it has.
- P: Accuracy

Consider any hypothesis h which is a function, $h : x \rightarrow class$. For instance:

$$h_1(x) = \text{stem-cancer} \quad \text{if} \quad x.date = \text{october}$$

Is this a good hypothesis ? How do we measure that ? One measure (P) could be the following:

$$P(h_1) = \frac{\#((x.date = \text{october}) \text{ AND } (h(x) = \text{stem-canker}))}{\#(x.date = \text{october})} = \frac{5}{90}$$

Consider the following other hypotheses and their P values:

$$h_2(x) = \text{stem-cancer} \quad \text{if} \quad x.plant\_stand = \text{normal}; \quad P(h_2) = \frac{20}{354}$$

$$h_3(x) = \text{stem-cancer} \quad \text{if} \quad ((x.plant\_stand = \text{normal}) \text{ AND } (x.precip = \text{gt-norm})); \quad P(h_2) = \frac{20}{354}$$

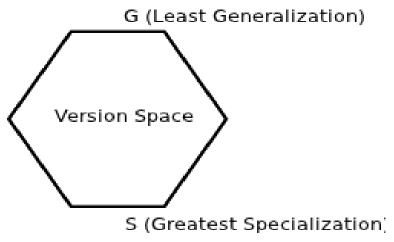


Figure 1: Version Space

1.2 Structure of hypotheses space: Version Space

Let us forget about getting to "the best hypothesis" and consider the structure of the hypothesis space. This space is also be termed as *Version Space.*

Formally defined as "*Space of h(x) such that* $\forall x \in E$, $h(x) = class(x)$". Pictorially it can be depicted as in figure 1:

A part of the version space for the soybean example is shown in figure 2.

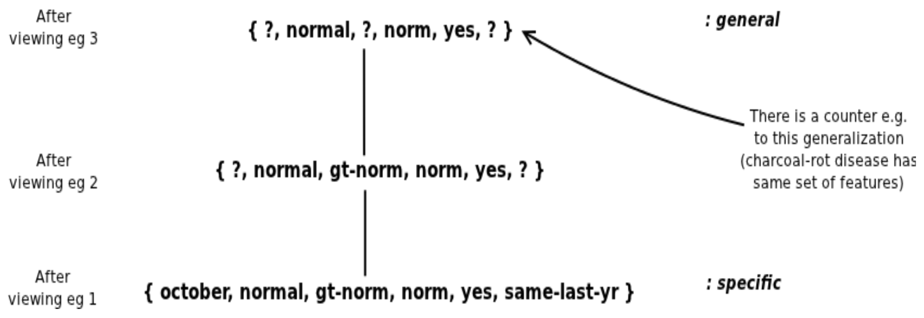

After viewing eg 3
{ ?, normal, ?, norm, yes, ? }
: general
There is a counter e.g. to this generalization (charcoal-rot disease has same set of features)
After viewing eg 2
{ ?, normal, gt-norm, norm, yes, ? }
After viewing eg 1
{ october, normal, gt-norm, norm, yes, same-last-yr }
: specific

Figure 2: Part of Version Space for Soybean example

Consider the cooked-up dataset shown is table 1. Our goal is to find a hypothesis for class C_1. If our hypothesis language is only a conjunction of atomic statements (i.e. they are conjunctions of stmts. of the form $x.attr = value$ or $x.attr = ?$), then the version space for this example is empty. In otherwords, we cannot find a hypothesis that belongs to the hypothesis language that we have defined, such that all the positive examples are covered and none of the negative examples are covered. (However, note that if we decide to include negation ($\neg$), then we can find a satisfying hypothesis for class C_1: $\neg(Y, N, ?) \Rightarrow C_1$).

Now consider the hypothesis $h_1 = (?, ?, N) \Rightarrow C_1$. What is $P(h)$? If we consider that our performance measure is defined as follows:

$$P = \frac{|\{x|h(x) = cl(x) = C_1\} \quad \bigcup \quad \{x|h(x) \neq C_1 \ \& \ cl(x) \neq C_1\}|}{|\{x\}|} = \frac{3}{4}$$

In short, this performance measure counts all those instances of C_1 that are correctly classified by the hypothesis and all those instance that are not C_1 and is not classified as C_1 by the hypothesis.

	F1	**F2**	**F3**	**Class**
D1	Y	Y	N	C1
D2	N	N	N	C1
D3	Y	Y	Y	C1
D4	Y	N	Y	C2

Table 1: A toy dataset

Ideally, we are in the search for that hypothesis that maximizes $P(h)$ i.e.

$$\arg\max_{h \in H} P(h)$$

Incremental building of version space

Consider the same dataset shown in table 1. As we see the examples one after another starting from $D1$, the progressive construction of version spaces is shown in diagram 3.

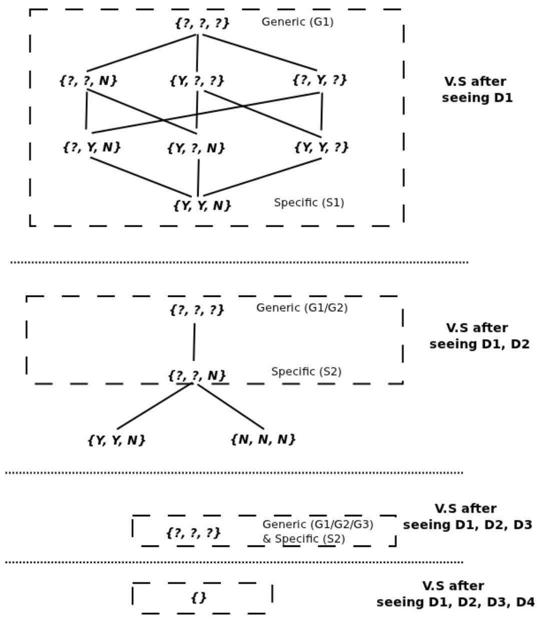


Figure 3: Progressive construction of Version Spaces for table 1

We can observe from diagram 1, that the version space becomes empty after seeing $D4$. If we expand our hypothesis language (for instance, to also include disjunctions of conjunctions), we can construct version spaces that are not empty.

In the lecture we saw three main type of extensions to the hypothesis language:

1. $(Y \wedge Y \wedge ?) \vee (N \wedge N \wedge ?) \Rightarrow C_1$. This, in the class, was termed as *lazy learner*. This can also be called *conjunctive normal form (CNF) learner*
2. $(? \wedge ? \wedge (N \vee O) \Rightarrow C_1$. (If, we change (D3,F3) to O)
3. $(attribute1 = attribute2) \Rightarrow C_1$

Some Observations

1. Version space may be empty. Generally, we cannot always find one hypothesis that will explain everything. So, as an approximation, we try to maximize P our performance measure.
2. Hypothesis language: There could be a bias in h. In our previous example, h_1 only consisted of a conjunction of atomic statements and we were avoiding disjunctions.
3. Active Learning: This is a learning technique where the machine prompts the user (an oracle who can give the class label given the features) to label an unlabeled example. The goal here is to gather as differentiating (diverse) an experience as possible. In a way, the machine should pick those examples which challenge the existing hypothesis learnt by it.

   For instance, if the machine has seen 2 examples : $\{Y, Y, Y\} \Rightarrow C_1$ and $\{Y, N, N\} \Rightarrow C_1$, then it should ask the user, $\{N, Y, Y\} \Rightarrow ?$.

1.3 Exercise

Construct version spaces for the 3 different hypothesis language extensions listed above along the lines of diagram 3 for the same dataset (Table 1).

Further Reading

1. Machine Learning, *Tom Mitchell*, McGraw Hill, 1997. (`http://www.cs.cmu.edu/~tom/mlbook.html`). Chapters 1 and 2.
2. Datasets: `http://storm.cis.fordham.edu/~gweiss/data-mining/datasets.html` (`iris.arff`, `soybean.arff`)

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 3: Bias, Course outline, Decision Trees

Instructor: *Ganesh Ramakrishnan* — Date: *29/07/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

Notation

A small change in notation to ensure conformity with the material to be covered in the future and ease of understanding. Previously we had denoted hypothesis by h, features by $x.feature\_name$ (where x was the example data) and class label by $c_1, c_2, ...$ From here onwards, we will denote features by $\phi(x_i)$, class label by $y(x_i)$ and the hypothesis by f. So our data with its features will be as shown in Table 2.

x_1	$\phi_1(x_1)$	$\phi_2(x_1)$	...	$\phi_m(x_1)$	$y(x_1)$
x_2	...	...	...	...	...
..					...
..	...	...	...	...	...
..					...
x_n	...	...	...	...	$y(x_n)$

Table 2: Data Representation

Our objective is to maximize $P(f)$ and we search in the hypothesis space H for that hypothesis f that maximizes the performance P. In otherwords,

$$\arg\max_{f \in H} P(f)$$

1.4 Version Space (cont.)

In the previous lecture, we saw the following types(families) of f for the toy dataset in Table 1.

1. $f : \wedge_i \phi_i(x) = 1,$ then $y(x) = c_1$ else c_2
2. $f : \wedge_{i,j} \phi_i(x) = \phi_j(x),$ then $y(x) = c_1$ else c_2
3. $f : \wedge_i (\phi_i(x) = a \vee \phi_j(x) = b),$ then $y(x) = c_1$ else c_2
4. $f : \vee_j (\phi_{i_1 j}(x) \wedge \phi_{i_2 j}(x)),$ then $y(x) = c_1$ else c_2

The version space for the fourth hypotheses language is as shown in Figure 4. Version space is usually represented by a *lattice* (which is a partially ordered set with relations *greatest lower bound* and *least upper bound* defined on every pair of elements).

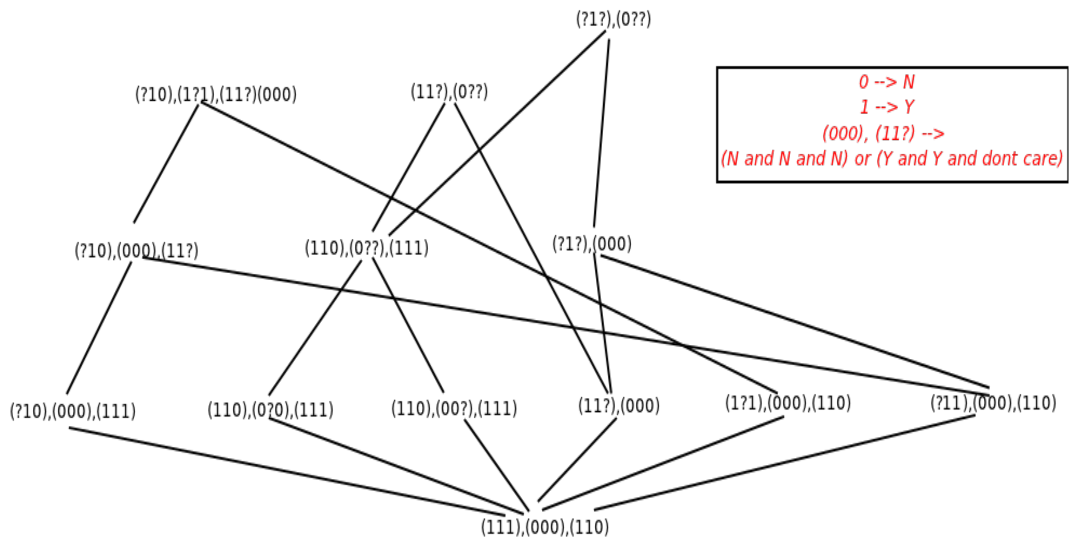


Figure 4: Version Space for Hypothesis language 4 (CNF)

1.5 Bias

Bias B is the assumptions we make about the target function f. Some of the effects of bias is as follows:

1. Can help reduce version space (to the point of emptiness).
2. Can lead to better coverage of new examples.

Bias and Coverage

Given a larger version space, the chances of contradictions while classifying a new example will be higher. Thus, while "more" examples will be covered in a larger version space, "fewer" examples will be *less ambiguously* covered in the smaller version space. In general, *smaller the version space lesser is the ambiguity in coverage.*

For just the first two rows of the example dataset (Table 1), the simple conjunction hypothesis language gives a smaller version space than the CNF hypothesis language. Note that *version space is defined as the space of hypothesis which is consistent with* ***all*** *the training examples.* Obviously, we would like **all** (or atleast a majority of them) to be consistent with any new example as far as possible.

But it can be proved that for any new example, exactly half the hypotheses in the CNF version space will cover it and half will not cover. Now what is the point of such a coverage? This is in contrast to the smaller version space where for most new examples, the ambiguity in coverage is much less.

For instance: Consider a new tuple (N, Y, N). In the smaller version space (Figure 3, 2nd sub-figure)), we see that the new datapoint is covered by both the hypotheses in the version space (no *ambiguity). However, in the larger version space (Figure 4), it is covered by 9 out of 13 hypotheses and not covered by the remaining 4 (There is more ambiguity in this case).*

This ambiguity can actually be captured through the concept of variance, which is what we will look at in greater detail when we talk about the *bias-variance dilemma* (in our discussions on probabilistic models).

2 Course Outline

Given bias B and the resulting version space from the bias ($V.S(B, D)$), the central question in machine learning is which f to pick ? Depending on how we do this, there are a host of techniques. Some of the classsification techniques that we cover in the course are as shown in Figure 5

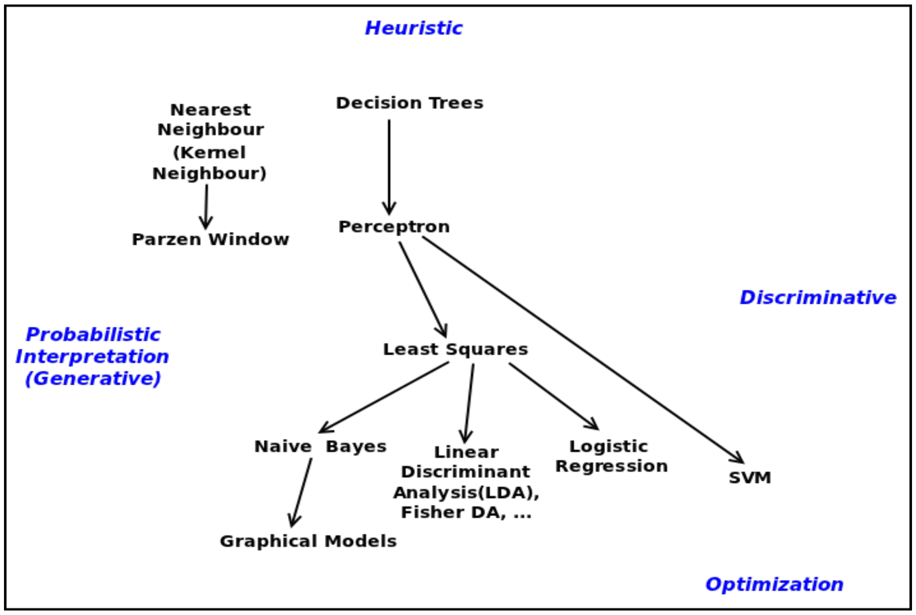


Figure 5: Classification techniques to be covered in the course

3 Decision Trees

The bias in a decision tree is as shown in Figure 6

Some characteristics / considerations:

1. Each example will take a definite path. (There is no ambiguity)
2. Which ϕ_i to pick ?

For a binary classification problem, we have $p(\oplus|x) + p(\ominus|x) = 1$. For the `vote.arff` example ($\oplus$ = *republicans* or $\ominus$ = *democrats*), $p(\oplus|x) = \frac{169}{268+169}$ and $p(\ominus|x) = \frac{268}{268+169}$.

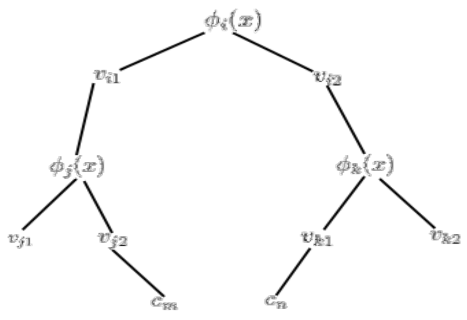


Figure 6: Decision Tree: Bias

We need to "Code up" the information to classify an example. Using information theory, we get the following equation:

$$-p(\oplus|x)\log_2 p(\oplus|x) - p(\ominus|x)\log_2 p(\ominus|x) = E_p(x)$$

This is the amount of uncertainty associated (also known as *entropy* $E_p(x)$). What we are interested is the relative chanage in entropy given a feature's value which is $E_{P_{pf}}(x)$ as shown in Figure 7

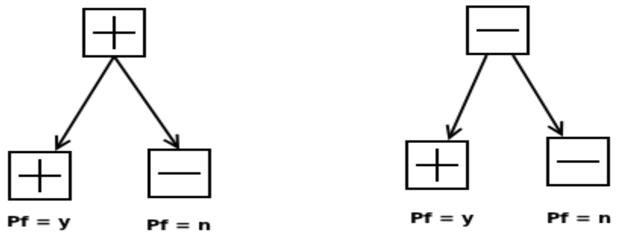


Figure 7: Decision Tree: Information Gain ($E_{P_{pf}}(x)$)

In otherwords, use that attribute that has maximum decrease in uncertainty. This measure is also called *Information Gain*.

3.1 Exercise

Is the version space in Figure 4 complete ? If not, complete it.

Further Reading

1. Lattice: `http://www.cse.iitb.ac.in/~cs717/notes/classNotes/completeNotes.pdf` (Use the same username and passwd). Pages 6-11.
2. Decision Trees (applets with illustrative examples): `http://webdocs.cs.ualberta.ca/~aixplore/learning/DecisionTrees/`

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 4: Decision Trees, Probability primer

Instructor: *Ganesh Ramakrishnan* — Date: *02/08/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

Hypothesis space

It is a disjunction of conjunctions. With respect to the `spect_train.arff` dataset, we observed the decision tree learnt as shown in Figure 8. We can serialize the decision tree as

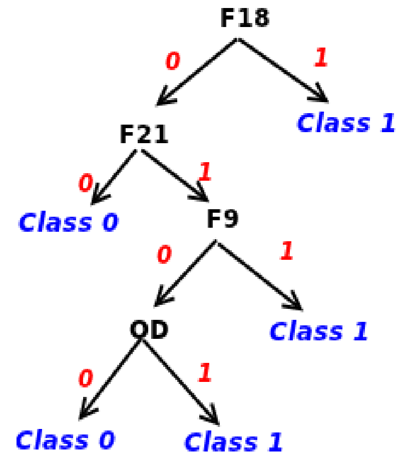


Figure 8: Decision Tree: `spect_train.arff`

$$\Big(\big((F_{18}=0)\wedge(F_{21}=0)\Rightarrow \text{class } 0\big)\vee\big((F_{18}=0)\wedge(F_{21}=1)\wedge(\text{OD}=0)\big)\Rightarrow \text{class } 0\big)\ldots\Big)$$

3.2 DTree Construction: splitting attribute

Mapping to Probability theory constructs

Refer to Section 4 for the corresponding definitions and explanation.

- $S=\{\oplus,\ominus\}$
- $E=\{\{\oplus,\ominus\},\{\oplus\},\{\ominus\},\{\}\}$
- Random variable mapping : $X(\{\oplus\})=1$, $X(\{\ominus\})=0$, $X(\{\oplus,\ominus\})=2$, $X(\{\})=3$
- With respect to the `spect_train` example file:
  $Pr(\{\oplus\})=Pr(X=1)=\frac{26}{80}$ $\quad Pr(\{\ominus\})=P(X=0)=\frac{54}{80}$

- To confirm, $Pr(\{\oplus, \ominus\}) = Pr(\{\oplus\}) + Pr(\{\ominus\}) = 1$
- Are $\{\oplus\}$ and $\{\ominus\}$ independent events ?
  No. $\because Pr(\{\oplus \cap \ominus\}) = Pr(\{\}) = 0 \neq Pr(\{\oplus\}) \bullet Pr(\{\ominus\})$
  (Intuition: If an instance has $\oplus$ label, then it cannot have a $\ominus$ label.)

New event and sample space

Consider the values of F_{18} and class label (first and second attribute is the following):

- $S^{'} = \{(0,0),(0,1),(1,0),(1,1)\}$
- $E^{'} = 2^{S^{'}} = \Big\{\{(0,0)\},\{(1,1)\},\cdots,\{(0,0),(0,1),(1,0),(1,1)\}\Big\}$
- $Pr(\{(0,0)\}) = \frac{51}{80}, \quad Pr(\{(0,1)\}) = \frac{21}{80}, \quad Pr(\{(1,1)\}) = \frac{5}{80}, \quad Pr(\{(1,0)\}) = \frac{3}{80}$
  (w.r.t `spect_train.arff`; here the 1st arg. is F_{18} and 2nd arg. is class label)
- $Pr(\{(0,1) \cup (0,0)\}) = Pr(\{(0,?)\} = \frac{51+21}{80} = \frac{72}{80}$
- Use of *Bayes Theorem*:

  Let $E^{''} = \big\{\{(0,0)\},\{(0,1)\},\{(1,0)\},\{(1,1)\},\{\}\big\}$ and
  Let $B_i = \{(1,0)\}$ and $A = \{(1,?)\} = \{(1,1) \cup (1,0)\}$

  $Pr(B_i|A) = \frac{Pr(B_i \cap A)}{Pr(A)} = \frac{Pr(\{(1,0)\})}{Pr(\{(1,0)\})+Pr(\{(1,1)\})} = \frac{3}{5+3}$ (*Marginalization*)
- (#bits needed to convey class label as $\oplus$) $\propto$ ($-\log_2 Pr(\oplus)$),
  (#bits needed to convey class label as $\ominus$) $\propto$ ($-\log_2 Pr(\ominus)$)
- Using the concept of Expectation $\mathcal{E}$:
  $\mathcal{E}[\log_2(\bullet)] = -Pr(\oplus) \bullet \log_2 Pr(\oplus) - Pr(\ominus) \bullet \log_2 Pr(\ominus)$
  This is also called *Entropy*. It represents the uncertainty asssociated with the encoding.

3.3 Splitting attribute : Impurity

There are various measures used for the selection of the splitting attribute in a decision tree. They all have the same goal of maximizing the reduction in a quantity called *Impurity* (Imp).(One of the measures of impurity is Entropy) Impurity represents the relative change in amount of bits needed to code up the information before and after a split happens in the decision tree. The idea being to choose only those split points which require the least amount of information to be coded up as a decision tree. The criterion for splitting is given by the following expression:

$$\arg\max_{\phi_j} Imp(D) - Imp(D_1, D_2)$$

$Imp(D_1, D_2)$ is given by:

$$Imp(D_1, D_2) = \frac{|D_1|}{|D|} Imp(D_1) + \frac{|D_2|}{|D|} Imp(D_2)$$

where, D_1 and D_2 are the subsets of data resulted from the split. The different measures of entropy are as shown in Table 3:

Name	**Imp (D)**
Entropy	$-\sum_{C_i} Pr(C_i) \bullet \log(Pr(C_i))$
Gini Index	$\sum_{C_i} Pr(C_i)(1 - Pr(C_i))$
Min Prob. Error	$\arg\min(1 - Pr(C_i))$

Table 3: Decision Tree: Impurity measurues

3.4 Exercise: Illustration of impurity

Consider the decision tree shown in Figure 9. Using *entropy* as measure, calculate the initial impurity and the impurity after the split.

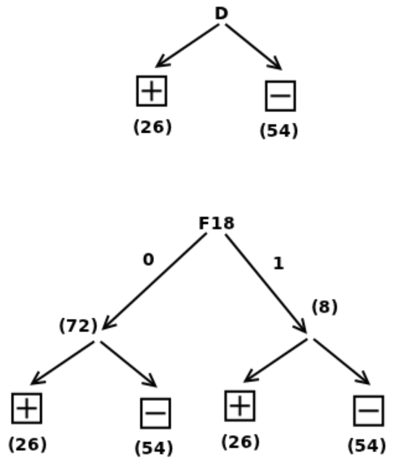


Figure 9: Impurity : before and after split

4 Probability Primer

A review of some basics of probability theory.

4.1 Basic Definitions

Definition 4.1. *Sample space (S) : A sample space is defined as a set of all possible outcomes of an experiment. Example of an experiment would be a coin pair toss. In this case* $S = \{HH, HT, TH, TT\}$.

Definition 4.2. *Event (E) : An event is defined as any subset of the sample space. Total number of distinct events possible is* 2^S, *where S is the number of elements in the sample space. For a coin pair toss experiment some examples of events could be*

$$\text{for at least one head, E} = \{HH, HT\}$$
$$\text{for all tails, E} = \{TT\}$$
$$\text{for either a head or a tail or both, E} = \{HH, HT, TH, TT\}$$

Definition 4.3. *Random variable (X) : A random variable is a mapping (or function) from set of events to a set of real numbers. Continuous random variable is defined thus*

$$X : 2^S \rightarrow \mathbb{R}$$

On the other hand a discrete random variable maps events to a countable set (e.g. discrete real numbers)

$$X : 2^S \rightarrow \textit{Discrete } \mathbb{R}$$

4.2 The three axioms of probability

Probability Pr is a number corresponding to events . It satisfies the following three axioms,

1. For every event E, $Pr(E) \in [0, 1]$
2. $Pr(S) = 1$ where, S is the sample space. (Equivalently, $P(\emptyset) = 0$)
3. If $E_1, E2, \ldots, E_n$ is a set of pairwise disjoint events, then

$$Pr(\bigcup_{i=1}^{n} E_i) = \sum_{i=1}^{n} Pr(E_i)$$

4.3 Bayes' Theorem

Let $B_1, B_2, ..., B_n$ be a set of mutually exclusive events that together form the sample space S. Let A be any event from the same sample space, such that $P(A) > 0$. Then,

$$Pr(B_i/A) = \frac{Pr(B_i \cap A)}{Pr(B_1 \cap A) + Pr(B_2 \cap A) + \cdots + Pr(B_n \cap A)} \tag{1}$$

Using the relation $P(B_i \cap A) = P(B_i) \cdot P(A/B_i)$

$$Pr(B_i/A) = \frac{Pr(B_i) \cdot Pr(A/B_i)}{\sum_{j=1}^{n} Pr(B_j) \cdot Pr(A/B_j)} \tag{2}$$

4.4 Independent events

Two events E_1 and E_2 are called independent iff their probabilities satisfy

$$P(E_1\, E_2) = P(E_1) \cdot P(E_2) \tag{3}$$

where $P(E_1\, E_2)$means$P(E_1 \cap E_2)$

In general, events belonging to a set are called as mutually independent iff, for every finite subset, $E_1, \cdots, E_n$, of this set

$$Pr(\bigcap_{i=1}^{n} E_i) = \prod_{i=1}^{n} Pr(E_i) \tag{4}$$

4.5 Probability Mass Function (pmf) and Probability Density Function(pdf)

Definition 4.4. *pmf :- It is a function that gives the probability that a discrete random variable is exactly equal to some value(Src: wiki).*

$$p_X(a) = Pr(X = a)$$

Definition 4.5. *Cumulative distribution function(Discrete case) : $F(a) = Pr(X <= a)$*

Definition 4.6. *pdf :- A probability density function of a continuous random variable is a function that describes the relative likelihood for this random variable to occur at a given point in the observation space(Src: wiki).*

$$Pr(X \in D) = \int_D p(x)dx \textit{ where D is set of reals and p(x) is density function.}$$

Definition 4.7. *Cumulative distribution function(Continuous case):*

$$F(a) = Pr(X <= a) = \int_{-\infty}^{a} p(x)dx$$

$$f(a) = \frac{dF(x)}{dx}|_{x=a}$$

Joint distribution function

If p(x,y) is a joint pdf i.e. for continuous case:
$F(a, b) = Pr(X <= a, Y <= b) = \int_{-\infty}^{b} \int_{-\infty}^{a} p(x, y)dxdy$
$p(a, b) = \frac{\partial^2 F(x,y)}{\partial x \partial y}|_{a,b}$

For discrete case i.e. p(x,y) is a joint pmf:
$F(a, b) = \sum_{x<=a} \sum_{y<=b} p(x, y)$

Marginalization

Marginal probability is then the unconditional probability P(A) of the event A; that is, the probability of A, regardless of whether event B did or did not occur. If B can be thought of as the event of a random variable X having a given outcome, the marginal probability of A can be obtained by summing (or integrating, more generally) the joint probabilities over all outcomes for X. For example, if there are two possible outcomes for X with corresponding events B and B', this means that $P(A) = P(A \bigcap B) + P(A \bigcap B')$. This is called marginalization.

Discrete case:
$P(X = a) = \sum_y p(a, y)$

Continuous case:
$P_x(a) = \int_{-\infty}^{\infty} p(a, y)dy$

4.6 Expectation

Discrete case: Expectation is equivalent to probability weighted sums of possible values.
$E(X) = \Sigma_i x_i Pr(x_i)$ where X is a random variable

Continuous case: Expectation is equivalent to probability density weighted integral of possible values.
$E(X) = \int_{-\infty}^{\infty} xp(x)dx$

If the random variable is a function of x, then Discrete case:
$E(f(X)) = \Sigma_i f(x_i) Pr(x_i)$ where X is a random variable

Continuous case:
$E(X) = \int_{-\infty}^{\infty} f(x)p(x)dx$

Properties of E(x)

$E[X + Y] = E[X] + E[Y]$

For any constant c and any random variable X
$E[(X - c)^2] \geq E[(X - \mu)^2]$
where $\mu = E[X]$

$E[cX] = cE[X]$

4.7 Exercise

Example 1. A lab test is 99% effective in detecting a disease when in fact it is present. However, the test also yields a false positive for 0.5% of the healthy patients tested. If 1% of the population has that disease, then what is the probability that a person has the disease given that his/her test is positive?

Soln. Let, H be the event that a tested person is actually healthy.
D be the event that a tested person does have the disease.
T be the event that the test comes out positive for a person.
We want to find out $Pr(D/T)$
H and D are disjoint events. Together they form the sample space.
Using Bayes' theorem,

$$P(D/T) = \frac{Pr(D) \cdot Pr(T/D)}{Pr(D) \cdot Pr(T/D) + Pr(H) \cdot Pr(T/H)} \tag{5}$$

Now, Pr(D) = 0.01 (Given)
Since Pr(D)+Pr(H)=1, Pr(H)=0.99
The lab test is 99% effective when the disease is present. Hence, Pr(T/D)=0.99

There is 0.5% chance that the test will give false positive for a healthy person. Hence, Pr(T/H)=0.005 Plugging these values in equation (5) we get,

$$\begin{aligned} Pr(D/T) &= \frac{0.01 * 0.99}{0.01 * 0.99 + 0.99 * 0.005} \\ &= \frac{2}{3} \end{aligned}$$

What does this mean? It means that there is 66.66% chance that a person with positive test results is actually having the disease. For a test to be good we would have expected higher certainty. So, despite the fact that the test is 99% effective for a person actually having the disease, the false positives reduce the overall usefulness of the test.

CS 725 : Foundations of Machine Learning Autumn 2011

Lecture 5: Decision Trees: Construction, Other considerations

Instructor: *Ganesh Ramakrishnan* Date: *06/08/2011*

Computer Science & Engineering Indian Institute of Technology, Bombay

5 Decision Trees (continued)

The splitting attribute is selected greedily as mentioned in the last class and is based on maximum reduction in impurity. The expression is given by:

$$\underset{V(\phi_i),\phi_i}{\arg\max}\left(Imp(D) - \sum_{v\in V(\phi_i)} \frac{|D_v|}{|D|} Imp(D_v)\right)$$

The second term in the above expression is the expected new impurity. V is a function which returns the split values given an attribute ϕ_i. So $V(\phi_i)$ can be varied for any ϕ_i. It could have many values or a range of values. A example of $V(\phi_i)$ is $V(\phi_i) = \{1, 3, 5\}$ which translates to the following split points $\phi_i < 1,\ 1 \leq \phi_i < 3,\ 3 \leq \phi_i < 5,\ \phi_i \geq 5$

5.1 An empirical observation

Since smaller range if attribute valuesin each split in V_i tends to lead to more skewed class distribution in that split, larger $|V(\phi_i)|$ generally yields larger reduction in impurity. This makes the algorithm to choose more skewed and complex trees and leads to the problem of `overfitting`, if not fixed. In overfitting, the system learns a model which is specific to the training collection and does not generalize well on unseen data.

Need to address this empirical observation

This can be done by the maximization of the following expression:

$$Imp(D) \ - \ \left(\frac{\sum_{v\in V(\phi_i)} \frac{|D_v|}{|D|} Imp(D_v)}{-\sum_{v\in V(\phi_i)} \frac{|D_v|}{|D|} \log(D_v)}\right)$$

The second term in the above expression is called $\triangle Imp(D)$. The intuition for using this term is that, *if the skew is more, the lower will be the denominator and is better at countering lowered impurity.* In otherwords, it prefers a less skewed tree as shown in Figure 10.

5.2 Decision Tree: construction algorithm

The algorithm goes on until all the data points at the leaf are of the one class and does not terminate if such a condition is violated. But the two stopping criterion added to the algorithm make it terminate even if such a condition does not occur.

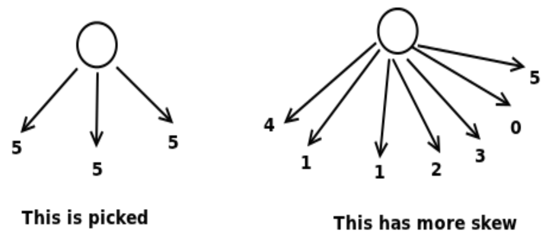


Figure 10: Skewness and empirical observation

Algorithm 1 T = dtree (D, ϕ_i, V)

```
if $\phi$ = empty then
    return a tree with only one branch $c_j$, where $c_j$ is the majority class in D {Stopping criterion}
end if
if all instances in D have label = $c_i$ then
    return a tree with only one branch $c_i$ {Stopping criterion}
end if
$\phi_j = \arg\max_{V(\phi_i),\phi_i} \big(\triangle Imp(D)\big) \quad \forall v \in V(\phi_i)$
$T_v = dtree(D_v, \phi - \phi_i, V)$
return a tree rooted at $\phi_i$ and having all the $T_v$ branches
```

5.3 Pruning

Simpler trees are preferred over their complex counterparts for the following reasons:

1. They are faster to execute
2. They perform better on unseen data. In otherwords, they "generalize well". For instance, a simpler tree learnt in the class had lower accuracy on the train set but higher accuracy on the unseen test set.

Trees can get unnecessarily deep and complex. So they are various strategies/heuristics to decrease the complexity of the tree learnt. Some of the options are as follows:

1. Early termination. Stop if $\triangle Imp(D) < \theta$, where θ is some threshold.
2. Majority class $\geq \alpha\%$, for some value of α
3. Pruning: The idea is to build complex trees and prune them. This is a good option, since the construction procedure can be greedy and does not have to look ahead. Some concepts of Hypothesis testing is used to achieve this. (For instance, χ^2-test).
4. Use an objective function like

$$max_{\phi_i}\Big(\triangle Imp(D,i) - Complexity(tree)\Big)$$

The complexity is characterised by the *description length principle (MDL)*

5.4 Exercise

Become familiar with the chebyshev's inequality, law of large numbers, central limit theorem and some concepts from Linear Algebra like vector spaces.

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 6: Probability distributions, Hypothesis Testing

Instructor: *Ganesh Ramakrishnan* — Date: *09/08/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

5.5 Splitting criterion modified only for pruning

1. Use only part of the data for pruning
2. Stop when $\triangle Imp < \theta$ (Problem: How to decide on θ ?)
3. *Minimum Description Length (MDL)* principle. ($\triangle Imp - complexity(tree) < \theta$)
4. *Significance tests (or hypothesis testing).* Can be considered as the statistical version of 2.

6 Probability Distributions

6.1 Bernoulli Random Variable

Bernoulli random variable is a discrete random variable taking values 0,1
Say, $Pr[X_i = 0] = 1 - q$ where $q\epsilon[0,1]$
Then $Pr[X_i = 1] = q$
$E[X] = (1-q) * 0 + q * 1 = q$
$Var[X] = q - q^2 = q(1-q)$

6.2 Binomial Random Variable

A binomial distribution is the distribution of n-times repeated bernoulli trials. A binomial random variable is discrete variable where the distribution is of number of 1's in a series of n experiments with {0,1} value, with the probability that the outcome of a particular experiment is 1 being q.

$Pr[X = k] = \binom{n}{k} q^k (1-q)^{n-k}$
$E[X] = \Sigma_i E[Y_i]$ where Y_i is a bernoulli random variable
$E[X] = nq$

$Var[X] = \Sigma_i Var[Y_i]$ (since Y_i's are independent)
$Var[X] = nq(1-q)$

An example of Binomial distribution is the distribution of number of heads when a coin is tossed n times.

6.3 Central Limit Theorem

If $X_1, X_2, .., X_m$ is a sequence of i.i.d. random variables each having mean μ and variance σ^2
Then for large m, $X_1 + X_2 + .. + X_m$ is approximately normally distributed with mean mμ and variance mσ^2
If $X \sim N(\mu, \sigma^2)$
Then $P[x] = \frac{1}{\sigma \sqrt[2]{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$

It can be shown by CLT

- $\frac{X_1+X_2+..+X_n-n\mu}{\sigma \sqrt[2]{n}} \sim N(0,1)$
- Sample Mean: $\hat{\mu} \sim N(\mu, \frac{\sigma^2}{m})$

6.4 Gaussian Distribution

Information Theory

Let us denote I(X=x) as the measure of information conveyed in knowing value of X=x.

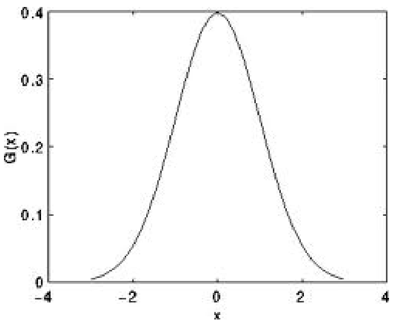


Figure 11: Figure showing curve where Information is not distributed all along.

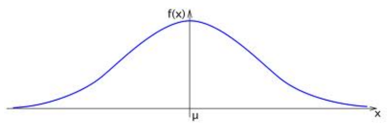


Figure 12: Figure showing curve where Information is distributed.

Question: Consider the two graphs above. Say you know probability function $p(x)$. When is knowing value of X more useful (that is, carries more information)?

Ans: It is more useful in the case(2), because more information is conveyed in Figure 11 than in Figure 12.

Expectation for I(X=x):

- If X and Y are independant random variables from the same distribution.

$$I(X = x, Y = x) = I(X = x) + I(Y = y) \tag{6}$$

  One way of expressing the above is:

$$I(P(x)P(y)) = I(P(x)) + I(P(y)) \tag{7}$$

  where P(x),P(y) are the probability functions respectively.

- If $p(x) > P(y)$, then

$$I(p(x)) < I(p(y))$$

  There is only one function which satisfies the above two properties.

$$I(p(x)) = -c\log(p(x)) \tag{8}$$

- The Entropy in the case of discrete random variable can be defined as:

$$E_P\left[I(p(x))\right] = \sum_x -c\log[p(x)] \tag{9}$$

- In the case of continuous random variable it is,

$$E_P\left[I(p(x))\right] = \int_x -c\log[p(x)] \tag{10}$$

  The constant 'C' in the above two equations is traditionally 1.

Observations:

- For a discrete random variable (with countable domain), the information is maximum for the uniform distribution.
- For Continuous random variable (with finite mean and finite variance), the information is maximum for the Gaussian Distribution.

Finding $\underset{p}{argmax}\ E_p$ in an infinite domain, subject to

$$\int xp(x)dx = \mu$$

and

$$\int (x-\mu)^2 p(x)dx = \sigma^2$$

The solution would be

$$p(x) = \frac{e^{\frac{-(x-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

Properties of gaussian univariate distribution

- If $X \sim N(\mu, \sigma^2)$

  $$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \ where - \infty < x < \infty$$

  then $w_1X + w_0 \sim N(w_1\mu + w_0, w_1^2\sigma^2)$
  (can prove this using moment generating function)

  $$\Phi(N(\mu, \sigma^2)) = E_{N(\mu,\sigma^2)}[e^{tx}] = e^{\mu t + \frac{(\sigma t)^2}{2}}$$

  <u>*Recall*</u>
  $E(X) = \frac{d\phi(p)}{dt}$
  $var(x) = \frac{d^2\phi(p)}{dt^2}$

  $$E_{N(\mu,\sigma^2)}[e^{t(w_1x+w_0)}] = (w_1\mu t + w_0 t + \frac{(\sigma t)^2}{2} \times w_1^2) \sim N(w_1\mu + w_0, w_1^2\sigma^2)$$

- Sum of i.i.d $X_1, X_2, ..., X_n \sim N(\mu, \sigma^2)$ is also normal (gaussian)

  $X_1 + X_2 + ... + X_n \sim N(n\mu, n\sigma^2)$

  In genaral if $X_i \sim N(\mu_i, \sigma_i^2) \implies \sum_{i=1}^n X_i \sim N(\sum \mu_i, \sum \sigma_i^2)$

- Corollary from (1) If $X \sim N(\mu, \sigma^2)$

  $$z = \frac{X-\mu}{\sigma} \sim N(0,1) \text{ (Useful in setting interval estimate)}$$

  (take $w_1 = \frac{1}{\sigma}$ *and* $w_0 = \frac{\mu}{\sigma}$)

  <u>Note:-</u> If $X_1, X_2, ...X_m \sim N(0,1)$

  1. $y = \sum_i X_i^2 \sim \chi_m^2$. That is, y follows the chi-square distribution with m-degrees of freedom.
  2. $y = \frac{z}{\sum X_i^2} \sim t_n$. (where $z \sim \mathcal{N}(0,1)$)). That is, y follows the *students-t* distribution.

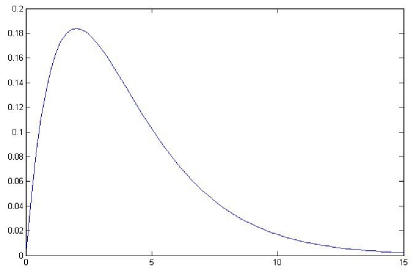


Figure 6.4 : Figure showing the nature of the $(chi - square)$ distribution with 5 degrees of freedom

- Maximum Likelihood estimate for μ and σ^2

  Given $X_1, X_2, ...X_m.....$ Random Sample.

$$\hat{\mu}_{MLE} = argmax_{\mu} \prod_{i=1}^{m} [\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(X_i-\mu)^2}{2\sigma^2}}]$$

$$= argmax_{\mu} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-\sum(X_i-\mu)^2}{2\sigma^2}}$$

  $\hat{\mu}_{MLE} = \frac{\sum_{i=1}^{m} X_i}{m}$ = sample mean

- With out relaying on central limit theorem Properties (2) and (1)

  i.e. Sum of i.i.d's $X_1, X_2, ..., X_n \sim N(\mu, \sigma^2)$

  $\hat{\mu}_{MLE} = N(\mu, \frac{\sigma^2}{m})$

  Similarly

  $\hat{\sigma}^2_{MLE} = \frac{\sum_{i=1}^{m}(X_i-\hat{\mu}_{MLE})^2}{m}$ is χ^2 distrbution

  $\sim \chi^2_m$

- Coming up with conjugate prior of $N(\mu, \sigma^2)$

  Case (1) σ^2 is fixed and prior on μ

  $\Rightarrow \mu \sim N(\mu_0, \sigma_0^2)$

Case (2) μ is fixed and σ^2 has prior

$$\Rightarrow \sigma^2 \sim \Gamma$$

case (3) if μ and σ^2 both having the prior

$$\Rightarrow (\mu, \sigma^2) \sim \text{Normal gamma distribution} \sim \text{Students-t distribution}$$

7 Hypothesis Testing

7.1 Basic ideas

Given a random sample for a random variable $X = (X_1, X_2, \dots X_n)$, we define a function $S : (X_1, X_2, \dots X_n) \rightarrow \mathbb{R}$. This function is called *statistic* (or *sample statistic*).

For instance, the *sample mean* is $\frac{\sum X_i}{n}$ and sample variance is $\frac{\sum_i \left(X_i - \frac{\sum X_i}{n}\right)^2}{n-1}$
Given X and 2 hypotheses H_0 and H_1 defined by:

$$H_0 : (X_1, X_2, \dots X_n) \in C$$

$$H_1 : (X_1, X_2, \dots X_n) \notin C$$

where, C is some tolerance limit also called the confidence region. It is generally defined in terms of some statistic.

The following types of errors are defined as a consequence of the above hypotheses. They are:

- Type I error: Probability of rejecting H_0, if H_0 was actually true.
  This is given by: $Pr_{H_0}(\{X_1, X_2, \dots X_n\} \notin C)$
- Type II error: Probability of accepting (or not-rejecting) H_0, if H_0 was actually false.
  This is given by: $Pr_{H_0}(\{X_1, X_2, \dots X_n\} \in C)$

Given a significance level α (some bound on the Type I error), we want to find a C such that,

$$Pr_{H_0}(\{X_1, X_2, \dots X_n\} \notin C) \leq \alpha$$

7.2 Example and some notation

Given a random sample $\{X_1, X_2, \dots X_n\}$ and each taking on one of k different categorical values. We denote $Pr(X_i = j)$ by p_j, which is the probability of random variable X_i taking the categorical value j. This distribution is called *categorical distribution* (also called as *multinomial distribution* - `http://en.wikipedia.org/wiki/Categorical_distribution`). This can be viewed as a *multivariate bernoulli* distribution.

Suppose we define a statistic: $Y_1, Y_2, \dots Y_k$ such that,

$$Y_i = \# \text{ of } X_j\text{'s that took the value } i$$

There are 2 types of representations possible to enocde such random variables.

Dirac-delta function

Here we represent the statistic Y_i as

$$Y_i = \sum_{j=1}^{n} \delta(X_j, i)$$

where $\delta(X_j, i) = 1$ when $X_j = i$th attribute and 0 otherwise.
For example, if 5 of $(X_1, \dots X_n)$ have value = 3, then $Y_3 = 5$.

Random vectors

In this representation, X is represented by random vector x^k as follows:

$$M = \begin{array}{c} x^1 \\ \vdots \\ x^k \end{array} \begin{pmatrix} 1 & 0 & \dots & 0 \\ & \vdots & & \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

Here X_j's are the vectors and the statistic Y_i is given by:

$$Y_i = (\sum_{j=1}^{n} X_j)[i]$$

where $[i]$ is the ith element of the vector obtained by summing individual random vectors.

A goodness of fit test

Let H_0 the hypothesis be defined as *the distance between the sample and expected average is within some tolerance.* if μ_i be the probability of $X_j = i$ i.e. $\mu_i = p_i$. Then, expectation $\mathcal{E}(Y_i) = n\mu_i$ and H_0 is mathematically derived as :

$$H_0 : t = \sum_{i=1}^{k} \Big(\frac{(Y_i - n\mu_i)^2}{n\mu_i} \Big) \le c$$

Here, t is called the *test statistic.* Alternatively,

$$Pr\left(\sum_{i=1}^{k} \frac{(Y_i - n\mu_i)^2}{n\mu_i} \ge c \right) \le \alpha$$

Given α, we need to compute c. Assuming large values of n, the above summation will be a chi-square distribution with $n-1$ degrees of freedom. $\left[\left(\sum_{i=1}^{k} \frac{(Y_i - n\mu_i)^2}{n\mu_i} \right) \sim \chi^2_{n-1} \right]$. There exits a table for determining c given the value of α.

CS 725 : Foundations of Machine Learning Autumn 2011

Lecture 7: Hypothesis testing and splitting criterion in DTrees

Instructor: *Ganesh Ramakrishnan* Date: *12/08/2011*
Computer Science & Engineering Indian Institute of Technology, Bombay

7.3 Statistical hypothesis testing in decision trees

Previously, we discussed the problem of complex decision trees and the need to obtain simpler trees by pruning. One important question to answer while constructing a simpler decision tree is *Do we have to split a node (using some attribute) or not ?* Consider the situation in Figure 13. The numbers n_1, n_2 etc. indicate the number of tuples of the particular type.

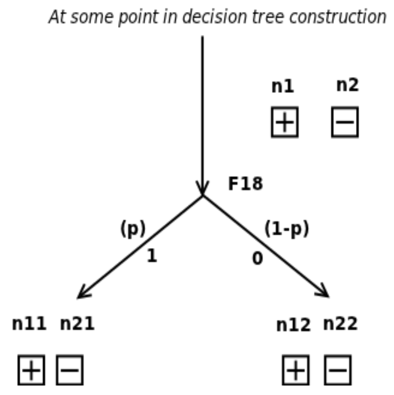


Figure 13: Splitting criterion

If the ratio of the separation of tuples remains the same (or is similar) to that before the split, then we might not gain much by splitting that node. To quantify this idea, we use the concept of significance testing. The idea is to compare 2 probability distributions.

Let us consider a 2-class classification problem. If p be the probability of taking the left branch, then the probablity of taking the right branch is $1 - p$. Then we obtain the following:

$$
\begin{aligned}
n_{11} &= pn_1 \\
n_{21} &= pn_2 \\
n_{12} &= (1-p)n_1 \\
n_{22} &= (1-p)n_2
\end{aligned}
$$

Consider the original ratio (also called reference distribution) of positive to total no. of tuples. If the same ratio is obtained after the split, we will **not** be interested in such splits. i.e.,

$$\frac{n_1}{n_1 + n_2} = \frac{n_{11}}{n_{11} + n_{21}}$$

or in general

$$\frac{n_j}{n_1+n_2} = \frac{n_{j1}}{n_{1i}+n_{2i}} \qquad \forall j \text{ no. of classes } \& \ i = 1,2$$

Since, these splits only add to the complexity of the tree and does not convey any meaningful information not already present. Suppose we are interested not only in equal distribution but also approximately equal distributions i.e.,

$$\frac{n_1}{n_1+n_2} \approx \frac{n_{11}}{n_{11}+n_{21}}$$

The idea is to compare two probability distributions. It is here that the concept of hypothesis testing is used.

The ratio $\frac{n_1}{n_1+n_2}$ is called the reference distribution. In general, it is:

$$p(c_j) = \mu_j = \frac{n_j}{\sum_i n_i} \qquad \text{for a given class } j \text{ (pre-splitting distribution)}$$

Hypothesis testing: problem [1]

Let $X_1, \ldots X_n$ be i.i.d random samples. For our example, class labels of instances that have gone into the left branch. The representation of these random variables can be any of the 2 types discussed in the last class. As per the random vector representation, statistic is $Y_j = \sum_{i=1}^{n} X_i[j]$. As per the dirac-delta representation, the statistic is $Y_j = \sum_{i=1}^{n} \delta(X_i, j)$. The statistic for the example, *# of instances in the left branch that have class j.*

The hypotheses:

$$\begin{aligned} H_0: &\quad X_1, \ldots, X_n \in C \\ H_1: &\quad X_1, \ldots, X_n \notin C \end{aligned}$$

The distribution of samples in the left branch is same as before splitting i.e. $\mu_1, \ldots \mu_k$.

Given a random sample, we need to test our hypothesis i.e., given an $\alpha \in [0,1]$, we want to determine a C such that

$$Pr_{H_0}(\{X_1, \ldots X_n\} \notin C) \le \alpha \qquad \text{Type I error}$$

Given an $\alpha \in [0,1]$, probability that we decide that the pre-distribution and the left-branch distribution are different, when in fact they are similar, is less than or eqaul to α.

[Currently we are not very much interested in the Type II error, i.e. $Pr_{H_1}(\{X_1, \ldots X_n\} \in C)$].

Here, C is the set of all possible "interesting" random samples. Also,

$$Pr_{H_0}(\{X_1, \ldots X_n\} \notin C') \le Pr_{H_0}(\{X_1, \ldots X_n\} \notin C) \qquad \forall C' \supseteq C$$

We are interested in the "smallest" / "tightest" C. This is called the critical region C_α. Consequently,

$$Pr_{H_0}(\{X_1, \ldots X_n\} \notin C_\alpha) = \alpha$$

[1] text written in red refers to the decision tree problem

Goodness-of-fit test for DTrees

We are interested in the hypothesis H_0 where new-distribution = old-distribution$(\mu_1, \dots \mu_j)$.

$$\begin{aligned}\mathcal{E}_{H_0}[Y_j] &= \sum_{i=1}^{n} \mathcal{E}_{H_0}[\delta(X_i, j)] \\ &= \sum_{i=1}^{n} \mu_j * 1 + (1 - \mu_j) * 0 \\ &= n\mu_j\end{aligned}$$

$$\begin{aligned}C &= \left\{ (X_1, \dots X_n) \middle| \sum_{i=1}^{k} \frac{(Y_j - \mathcal{E}_{H_0}(Y_j))^2}{\mathcal{E}_{H_0}(Y_j)} \leq c \right\} \quad \text{where } c \text{ is some constant} \\ &= \left\{ (X_1, \dots X_n) \middle| \sum_{i=1}^{k} \frac{(Y_j - n\mu_j)^2}{n\mu_j} \leq c \right\}\end{aligned}$$

As we have seen before, the above expression $\sim \chi^2_{k-1}$. We then use the chi-square tables to find c given the value of α.

Heuristic used for DTree construction

- Compute $\sum_{i=1}^{k} \frac{(Y_j - n\mu_j)^2}{n\mu_j} \sim t_s \quad \forall\ splits$, where t_s is the test statistic.
- Stop building the tree, if for a given α, $\quad t_s \leq c_\alpha \quad \forall\ splits$
- Compute c_α such that $Pr_{\chi^2_{k-1}}(x \geq c_\alpha) = \alpha$

8 Estimation

In *estimation*, we try to determine the probability values (till now we used to consider the ratio as the probability). Essentially, our μ_j should actually denoted by $\widehat{\mu_j}$, since it is an *estimate* of the actual probability μ_j (which we do not know).

The question posed in estimation is *How to determine* $\widehat{\mu_j}$ *?*
If $\{X_1, \dots X_n\}$ are the class labels from which we want to estimate μ, then:

$$\begin{bmatrix} \widehat{\mu_1} \\ \vdots \\ \widehat{\mu_k} \end{bmatrix} = \operatorname*{arg\,max}_{\mu_1, \dots, \mu_k} \left[Pr\left(\prod_{i=1}^{n} \prod_{j=1}^{k} \mu_j^{\delta(X_i, j)} \right) \right] \qquad s.t. \quad \sum_{i=1}^{k} \mu_i = 1$$

This is also known as the *maximum likelihood estimator (MLE)* for the multivariate bernoulli random variable. For $k = 2$,

$$
\begin{aligned}
&= \arg\max_{\mu_1,\mu_2} \left[\prod_{i=1}^{n} \mu_1^{\delta(X_i,1)} \mu_2^{\delta(X_i,2)} \right] \\
&= \arg\max_{\mu_1} \left[\mu_1^{\sum_i \delta(X_i,1)} \left(1 - \mu_1^{\sum_i \delta(X_i,2)} \right) \right]
\end{aligned}
$$

Further Reading

Read section 4.1.3 from the convex optimization notes
(`http://www.cse.iitb.ac.in/~cs725/notes/classNotes/BasicsOfConvexOptimization.pdf`)

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 8: Maximum Likelihood Estimation, Optimization Basics

Instructor: *Ganesh Ramakrishnan* — Date: *16/08/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

8.1 Maximum Likelihood Estimation (MLE)

Continuing from the prvious lecture, Let the random sample X be as follows

$$\text{Random Sample } (X) \sim (X_1, \ldots X_i \ldots X_n)$$

where each random variable X_i takes on of k values from $V_1 \ldots V_k$

The maximux likelihood objective function will be as follows:

$$\widehat{\mu}_{ML} = \arg\max_{\mu_1 \ldots \mu_k} \sum_{j=1}^{k} \sum_{i=1}^{n} \delta(X_i, V_i) \log_2 \mu_j$$

such that,

$$\sum_i \mu_i = 1 \quad ; \quad 0 \leq \mu_i \leq 1 \quad , \text{ for } i = 1 \ldots k$$

For $k = 2$ (i.e. we have $\mu_2 = 1 - \mu$),

$$\widehat{\mu}_{ML} = \arg\max_{\mu} \left[\Big(\sum_{i=1}^{n} \delta(X_i, V_1) \log_2 \mu \Big) \Big(\sum_{i=1}^{n} \delta(X_i, V_2) \log_2 (1 - \mu) \Big) \right]$$

such that, $\mu \in [0, 1]$

The expression in between [ ] is called the *objective function* f, which need to be optimized (maximised in this case). This particular form of the objective function is called the *log-likelihood* function. The interval from which the values of μ can come from is called the domain D. In this case it is a closed and bounded interval. To find the optimized value we will use the principles from Optimization theory.

9 Optimization

Refer to the optimization theory notes (`http://www.cse.iitb.ac.in/~cs725/notes/classNotes/BasicsOfConvexOptimization.pdf`). All references to sections, theorems will be with respect to these notes. It will be denoted for instance as *CoOpt,4.1.1*.

For some basic definitions of optimization, see sections *CoOpt,4.1.1* and *CoOpt,4.1.2*. To find the maximum minimum values of an objective function, see section *CoOpt,4.1.3* from Theorem 38 to Theorem 56. A set of systematic procedures to find optimal value is listed in Procedures 1, 2, 3 and 4 (*CoOpt, pages 10, 16, 17*).

For our optimization problem, the optimum value occurs when $f^{'}(\widehat{\mu}_{ML}) = 0$, which implies

$$\Rightarrow \frac{\sum_{i=1}^{n}\delta(X_i, V_1)}{\widehat{\mu}_{ML}} - \frac{\sum_{i=1}^{n}\delta(X_i, V_2)}{(1-\widehat{\mu}_{ML})} = 0$$

$$\Rightarrow \frac{\sum_{i=1}^{n}\delta(X_i, V_1)}{\sum_{i=1}^{n}\delta(X_i, V_1) + \sum_{i=1}^{n}\delta(X_i, V_2)} = \widehat{\mu}_{ML}$$

So, necessary condition for local extreme value is

$$\widehat{\mu}_{ML} = \frac{\sum_{i=1}^{n}\delta(X_i, V_1)}{n}$$

In this case, this is the only possible value. However, there a couple of issues that needs to be sorted out.

- Is it an extreme value value at all ? If so, is it local or a global extreme value ?
- Is it a maximum or a minimum value ?

The theorems listed above will give the necessary and sufficient conditions for maximum value of the objective function f is as follows:

- Necessary Conditions
  1. $f^{'} = 0$ at $\widehat{\mu}_{ML}$
  2. $f^{'} \geq 0$ before $\widehat{\mu}_{ML}$
  3. $f^{'} \leq 0$ after $\widehat{\mu}_{ML}$
- Sufficient Conditions
  1. $f^{'} = 0$ at $\widehat{\mu}_{ML}$
  2. $f^{'} > 0$ before $\widehat{\mu}_{ML}$
  3. $f^{'} < 0$ after $\widehat{\mu}_{ML}$

We would like our log-likelihood function to be strictly concave function in $[0, 1]$.

Using *CoOpt, page 17, Procedure 4*, we can narrow down on the possible values (where the function attains maximum value) as

$$\widehat{\mu}_{ML} \in \left\{0, 1, \frac{\sum_{i=1}^{n}\delta(X_i, V_1}{n}\right\}$$

However, $f(0) = -\infty$, $f(1) = -\infty$. From this we can conclude, it is indeed the case that,

$$\widehat{\mu}_{ML} \text{ is } \frac{\sum_{i=1}^{n}\delta(X_i, V_1)}{n}$$

We can come to same conclusion through the another path, that of *CoOpt, Theorem 55.* The reasoning is as follows. Let the log-likelihood objective function be denoted by $LL(\mu)$

$$\left.\begin{aligned} LL^{''}(\mu) &= \frac{-\left(\sum_{i=1}^{n}\delta(X_i, V_1)\right)}{\mu^2} \\ &= \frac{-\left(\sum_{i=1}^{n}\delta(X_i, V_2)\right)}{(1-\mu)^2} \end{aligned}\right\} \quad < 0 \quad , \quad \mu \in [0, 1]$$

So, minimum value is at 0 or 1, since $LL^{''}(\mu) < 0, \quad \forall \mu \in [0,1]$, which implies that

$$\text{if } LL^{''}(\mu) = 0, \text{ for } \mu \in [0,1] \text{ then } \widehat{\mu} \text{ is argmax } LL$$

The more general LL objective function with the constraints $\sum_i \mu_i = 1,\ 0 \leq \mu_i \leq 1,$ for $i = 1 \ldots k$ geometrically forms what is known as a simplex. (`http://en.wikipedia.org/wiki/Simplex`)

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 9: Optimization Basics, Linear Algebra

Instructor: *Ganesh Ramakrishnan* — Date: *19/08/2011*
Computer Science & Engineering — Indian Institute of Technology, Bombay

9.1 Optimization for multiple variables

$$\begin{aligned}\arg\max_{\mu_1\dots\mu_k} LL(\mu_1\dots\mu_k|x_1\dots x_n) &= \arg\max_{\mu_1\dots\mu_k}\sum_{i=1}^{n}\sum_{j=1}^{k}\delta(X_i,V_j)log\mu_j\\ &= \arg\max_{\mu_1\dots\mu_k}\sum_{j=1}^{k} n_j log\mu_j \text{ , where } n_j=\sum_{j=1}^{m}\delta(X_i,V_j)\end{aligned}$$

such that, $0 \le \mu_i \le 1$ and $\sum_i \mu_i = 1$

Directional derivative, Level curves, Gradient

The notion of *directional derivative* (D), can be obtianed from *CoOpt, Section 4.1.4* and that of level curves from *CoOpt, Figure 4.12* and the associated explanation. Briefly, for a function $f(\bar{x})$, defined on domain D, is given by

$$\text{Level Curve } (C) = \{\bar{x} \mid \bar{x} \in D, f(\bar{x}) = c\}$$

The level curves applet can be found at `http://www.slu.edu/classes/maymk/banchoff/LevelCurve.html`.

Let us consider the simple case of 2-dimensions, and say $n_1 = 10$ and $n_2 = 15$, and $k = 2$, if $LL(\mu_1, \mu_2) = c$, then

$$\begin{aligned} n_1 \log_e \mu_1 + n_2 \log_e \mu_2 = c\\ \Rightarrow \mu_2 = e^{\frac{c-n_1\log_e \mu_1}{n_2}}\end{aligned}$$

Directional derivative along the standard axis for the sample objective function is

$$D_{[1\ 0](LL)=\frac{n_1}{\mu_1}}$$
$$D_{[0\ 1](LL)=\frac{n_2}{\mu_2}}$$

By, *CoOpt, Theorem 57*,

$$D_{[\frac{1}{\sqrt{2}}\ \frac{1}{\sqrt{2}}]}(LL(\mu_1,\mu_2)) = \frac{1}{\sqrt{2}}\frac{n_1}{\mu_1} + \frac{1}{\sqrt{2}}\frac{n_2}{\mu_2}$$

Gradient ∇ is given as

$$\nabla LL = \begin{bmatrix} \frac{\partial LL}{\partial \mu_1} \\ \frac{\partial LL}{\partial \mu_2} \end{bmatrix} = \begin{bmatrix} \frac{n_1}{\mu_1} \\ \frac{n_2}{\mu_2} \end{bmatrix}$$

In general for arbitrary k, (∇LL is in n-dim space)

$$\nabla LL = \begin{bmatrix} \frac{\partial LL}{\partial \mu_1} \\ \vdots \\ \frac{\partial LL}{\partial \mu_i} \\ \vdots \\ \frac{\partial LL}{\partial \mu_k} \end{bmatrix} = \begin{bmatrix} \frac{n_1}{\mu_1} \\ \vdots \\ \frac{n_i}{\mu_i} \\ \vdots \\ \frac{n_k}{\mu_k} \end{bmatrix}$$

By, *CoOpt, Theorem 58*,

$$\arg\max_v D_v(LL) = D_{\frac{\nabla LL}{\|\nabla LL\|}}(LL)$$

Hyperplane, norms, local minima and maxima, Hessian

Hyperplane H is given by

$$H = \{p \mid (p-q)^T v = 0\}$$

By *CoOpt, Theorem 59*, Grad vector is perpendicular to the tangent hyperplance at the level curve.

Norm is function denoted by $\|x\|$ and has the following properties

1. $\|x\| \geq 0$
2. $\|cx\| = |c|\|x\|$
3. $\|x+y\| \leq \|x\| + \|y\|$

Some common norms

1. L_1 norm $= |x_1| + \cdots + |x_n|$
2. L_2 norm $= \sqrt[2]{(x_1^2 + \cdots + x_n^2)}$
3. L_k norm $= (|x_1^k| + \ldots |x_n^k|)^{\frac{1}{k}}$

By, *CoOpt, Theorem 60*, if $\mu_1^0 \ldots \mu_k^0$ is a point of local min/max, then $\nabla LL(\mu_0) = 0$, if it exists. Note, that this is only a necessary but not a sufficient condition.

CoOpt, Theorem 61, defines the notion of the Hessian matrix, given as follows:

$$\nabla^2 LL = \begin{bmatrix} \frac{\partial^2 LL}{\partial \mu_1^2} & \frac{\partial^2 LL}{\partial \mu_2 \partial \mu_1} & \cdots & \frac{\partial^2 LL}{\partial \mu_k} \\ \frac{\partial^2 LL}{\partial \mu_1 \partial \mu_2} & \frac{\partial^2 LL}{\partial \mu_2^2} & \cdots & \\ \vdots & & \ddots & \\ \vdots & & \cdots & \frac{\partial^2 LL}{\partial \mu_k^2} \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{n_1}{\mu_1^2} & & & \\ & -\frac{n_2}{\mu_2^2} & & 0 \\ 0 & & \ddots & \\ & & & -\frac{n_k}{\mu_k^2} \end{bmatrix}$$

9.2 Linear Algebra

Some basics of linear algebra relavant to our optimization problem are listed here. The link for the linear algbra notes is `http://www.cse.iitb.ac.in/~cs725/notes/classNotes/LinearAlgebra.pdf`. The notation used to refer to this linear algebra notes is *LinAlg, Section 3.11.*

Eigenvalues and Eigenvectors

Refer to *LinAlg, Section 3.11* for the same. Some of their properties are as follows:

- If $\mathbf{Ax} = \lambda\mathbf{x}$ then $(\mathbf{A} + k\mathbf{I})\mathbf{x} = (\lambda + k)\mathbf{x}$. This is very imporatant in ML. We will revisit this property in *gaussian discriminant analysis, SVMs* etc.
- If $\mathbf{A}$ is a triangular matrix, then $\lambda_i = a_{ii}$
- if $\mathbf{Ax} = \lambda\mathbf{x}$ then $\mathbf{A}^2\mathbf{x} = \lambda^2\mathbf{x}$
- Also $\sum \lambda_1 = trace(A) = \sum_j a_{jj}$ and $\prod \lambda_i = det(A)$

Matrix factorization using Eigenvectors, positive definite matrices

Refer to *LinAlg, Section 3.12.* Positive definite matrices are denoted by $\mathbf{A} \succ_{pd} 0$ and positive semi-definite matrices are denoted by $\mathbf{A} \succeq_{psd} 0$.

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 10: Optimization continued

Instructor: *Ganesh Ramakrishnan* — Date: *23/08/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

From the previous lecture, our objective function was

$$\arg\max LL(\mu_1 \dots \mu_k | n_1, \dots n_j) = \arg\max \sum_j n_j \log \mu_j$$

subject to the constraints $\sum \mu_j = 1$ and $\mu_j \in [0,1]$. Here $n_1, \dots n_j$ are accumulated from $X_1, \dots X_n$. From *CoOpt, Theorem 61*, we have the one of the conditions for local optimality (maximum) as the Hessian to be positive definite. i.e

$$\nabla^2 LL = \begin{bmatrix} -\frac{n_1}{\mu_1^2} & & & 0 \\ & -\frac{n_2}{\mu_2^2} & & \\ 0 & & \ddots & \\ & & & -\frac{n_k}{\mu_k^2} \end{bmatrix} \succ 0$$

CoOpt, Theorem 62 and Corollary 63 list the necessary and sufficient conditions for local maxima (resp. minima) as follows: (reproduced here for convenience, please refer to *CoOpt* notes for details and proof)

Necessary Conditions

$$\nabla f(x^*) = 0$$
$$\nabla^2 f(x^*) \preccurlyeq 0 \quad (\nabla^2 f(x^*) \succcurlyeq 0 \text{ , minima resp.})$$

Sufficient Conditions

$$\nabla f(x^*) = 0$$
$$\nabla^2 f(x^*) \prec 0 \quad (\nabla^2 f(x^*) \succ 0 \text{ , minima resp.})$$

Some points to note here. If $\nabla^2 f(x^*)$ is neither $\succcurlyeq 0$ nor $\preccurlyeq 0$, then x^* is called a saddle point. Also to visualize, *negative definiteness* implies *upward curvature* of the function surface.

However, for our given function, the gradient does not disappear. Then, how do we find the maximum value for our objective function ? How about taking a relook at the method for finding the maximum value. Some considerations:

1. How about restricting the attention to the domian set of interest ? (for us it is $\mu_i \in [0,1]$ with the constraint $\sum \mu_i = 1$

2. In this restricted region compute the extreme values.

By *CoOpt. Theorem 65*, we know that for a closed and bounded set, there exists a global maximum and global minimum. Restricting ourselves to Euclidean spaces, some definitions are as follows:

- Bounded: $\exists$ sphere S in R^k s.t $D \subseteq S$
- Open: $\forall x \in D$, $\exists$ a sphere $S(x)$ of radius $\epsilon > 0$ centered at x, s.t. $S(x) \subseteq D$
- Closed: complement D^c is open.
- Boundary: $B(D) = \{y \mid \nexists \epsilon > 0, \text{ s.t } sphere(\in, x) \subseteq D\}$

So, the naive method adapted to our case is as follows

- Step 1: See if $\nabla f = 0$
- Step 2: Find max of f along boundaries (relative to R^{k-1})
- Step 3: Find max of f along boundaries (relative to R^{k-2})
- Step 4: Find max of f along boundaries (relative to R^{k-3})
- $\ldots$

For definitions of relative boundary, relative interior and an example concerning the same. refer to *CoOpt, pg 250, 251.*

Example

Let $k = 4$. The objective function is

$$n_1 \ln \mu_1 + n_2 \ln \mu_2 + n_3 \ln \mu_3 + n_4 \ln \mu_4$$

such that,

$$\mu_1 + \mu_2 + \mu_3 + \mu_4 = 1 \text{ and } \mu_i \in [0, 1]$$

In step 1 we calculate the ∇f. In step 2, we find the boundary relative to R^3, where $\mu_x = 1$ or $\mu_x = 0$. So we get the following equations

$$\begin{aligned} \mu_1 + \mu_2 + \mu_3 &= 1 \\ \mu_4 &= 1 \\ \mu_1 + \mu_2 + \mu_4 &= 1 \\ \mu_3 &= 1 \\ &\ldots \end{aligned}$$

In step 3, we find boundaries relative to R^2 and so on. As an exercise, complete this example.

Another approach (using level curves)

In general we have

$$\begin{aligned} max\ f(x) &\quad (LL(\mu_1, \ldots \mu_k)) \\ \text{s.t } g(x) = 0 &\quad (\textstyle\sum \mu_i - 1 = 0) \end{aligned}$$

Consider level curves for $f(x)$. If $x^* \in$ local $\arg\max f(x)$, subject to constraints, we have the following observations

- If ∇f has component perpendicular to the direction of ∇g
- The above statement implies that we can move along $g(x) = 0$ while increasing value above $f(x^*)$
- This is in violation of the assumption that x^* is local argmax

From this, we can conclude that if x^* is local argmax, then ∇f has no component perpendicular to ∇g. Therefore, at x^*,

$$\begin{aligned} g(x^*) &= 0 \\ \nabla f(x^*) &= \lambda \nabla g(x^*) \end{aligned}$$

i.e. they are parallel to each other.

Read *CoOpt, Section 4.4.1* on Lagrange Multipliers for a detailed exposition of the material covered next. Say $x^* \in$ local $\arg\max f(x)$ such that some equality constraints are satisfied. i.e.

$$\begin{aligned} x^* \in \text{ local } &\arg\max f(x) \\ \text{s.t. } &g_1(x) = 0 \\ &g_2(x) = 0 \\ &\vdots \\ &g_n(x) = 0 \end{aligned}$$

Components of $\nabla f(x^*)$ perpendicular to the space spanned by $\{\nabla g_i(x^*)\}_{i=1,\ldots m}$ should be 0. i.e.

$$\begin{aligned} \nabla f(x^*) &= \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) \quad \text{(Linear Combination)} \\ g_i(x^*) &= 0 \quad , \forall i = 1, \ldots m \end{aligned}$$

Suppose we have inequality constraints instead of equality. i.e.

$$\begin{aligned} x^* \in \text{ local } &\arg\max f(x) \\ \text{s.t. } &g_1(x) \leq 0 \end{aligned}$$

We have the following possibilities:

P1

$$\begin{aligned} g(x^*) &= 0 \\ \nabla f(x^*) = \lambda \nabla g(x^*) \quad &, \lambda \geq 0 \end{aligned}$$

P2

$$\begin{aligned} g(x^*) &< 0 \\ \nabla f(x^*) &= 0 \end{aligned}$$

Summarising P1 and P2, we have,

$$\begin{aligned} \nabla f(x^*) - \lambda \nabla g(x^*) &= 0 \\ \lambda &\geq 0 \\ \lambda g(x^*) &= 0 \end{aligned}$$

For multiple inequality constraints, $g_1, \ldots g_m$, we have,

$$\begin{aligned} \nabla f(x^*) - \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) &= 0 \\ \lambda_i \geq 0 & \quad , \forall i = 1, \ldots m \\ \lambda_i g_i(x^*) = 0 & \quad , \forall i = 1, \ldots m \end{aligned}$$

Exercise

For the log-likelihood function (our objective function), compute the above form of the equation (Lagrangian)

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 11: Optimization, KKT conditions

Instructor: *Ganesh Ramakrishnan* — Date: *26/08/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

For the log-likelihood function $f = \sum_{j=1}^{k} n_j \mu_j$, we get the following necessary condition for optimality (Lagrangian)

$$\begin{bmatrix} \frac{n_1}{\mu_1} \\ \vdots \\ \frac{n_i}{\mu_i} \\ \vdots \\ \frac{n_k}{\mu_k} \end{bmatrix} - \lambda\Big(\sum_{i=1}^{k} \mu_i - 1\Big) - \sum \alpha_i \frac{\partial \mu_i}{\partial \mu_i} - \sum \beta_i \frac{\partial \mu_i}{\partial \mu_i} = 0$$

The constraints on α_is and β_i are as follows:

$$\begin{aligned} g_i \le 0 \quad &: \quad -\mu_i \le 0 \quad \sim \quad \alpha_i \\ g_{k+j} \le 0 \quad &: \quad \mu_j - 1 \le 0 \quad \sim \quad \beta_i \\ h = 0 \quad &: \quad \sum_{j=1}^{k} \mu_j - 1 = 0 \quad \sim \quad \lambda \end{aligned}$$

9.3 KKT conditions

Refer to *CoOpt, Section 4.4.4* for a detailed explanation and proofs. Suppose we have the problem

$$\begin{aligned} \widehat{\mu} = \arg\min_{\mu} f(\mu) \\ \text{s.t. } g_i(\mu) \le 0 \\ h_(\mu) = 0 \end{aligned}$$

We will have the necessary conditions for optimality as follows (also known as the Karash-Kuhn-Tucker or KKT conditions)

1. $\begin{bmatrix} -\frac{n_1}{\mu_1} \\ \vdots \\ -\frac{n_i}{\mu_i} \\ \vdots \\ -\frac{n_k}{\mu_k} \end{bmatrix} + \sum_{j=1}^{k} \alpha_j(-1) + \sum_{j=1}^{k} \beta_j(1) + \lambda = 0$

2. $-\widehat{\mu_j} \leq 0$
   $\widehat{\mu_j} - 1 \leq 0$

3. $\alpha_j \geq 0$
   $\beta_j \geq 0$

4. $-\alpha_j \widehat{\mu_j} = 0$
   $\beta_j(\widehat{\mu_j} - 1) = 0$

5. $\sum_{j=1}^{k} \widehat{\mu_j} + 1 = 0$

From 4., we have $\mu_j \neq 0 \Rightarrow \alpha_j = 0$. Also from 4., if no $n_j = 0$, $\beta_j = 0$, $\sum \widehat{\mu_j} = 1$. i.e. $\frac{\sum n_j}{\lambda} = 1 \Rightarrow \lambda = \sum n_j \Rightarrow$

$$\widehat{\mu_j} = \frac{n_j}{\sum n_j}$$

We still need to verify that $\widehat{\mu_j} = \frac{n_j}{\sum n_j}$ is indeed the globally optimal value. For this we have to use *CoOpt, Theorem 82*, which provides us the sufficient conditions for global optimality.

Convex Analysis

For the 1-dim case and $\alpha \in [0,1]$, the condition for convexity is as follows:

$$f(\alpha x + (1-\alpha)y \leq \alpha f(x) + (1-\alpha) f(y)$$

For strict convexity :

$$f(\alpha x + (1-\alpha)y < \alpha f(x) + (1-\alpha) f(y)$$

Convex sets: S is convex if

$$\forall x, y \in S, \quad \alpha x + (1-\alpha) y \in S \text{ where } \alpha \in [0,1]$$

For the n-dim case, $\mathbf{x}$, $\mathbf{y}$ and $\nabla \mathbf{f}$ are vectors (so denoted in bold). We have the condition of convexity as:

$$f(\alpha \mathbf{x} + (1-\alpha)\mathbf{y} < \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y})$$

For detailed definitions of convex sets, affine sets, convex functions and examples of the same refer from *CoOpt. Section 4.2.3* onwards.

9.4 Necessary and Sufficient Conditions : Summary

We will consider both the 1-dim and n-dim cases. In both the cases the domain D must be convex. Also the conditions involving gradient and hessian should be held only for all x in the interior of the domain. The conditions are summarised in Table 4.

If we reverse all the inequalities we get the conditions for concavity.

Our function (LL)

The domain $D = \{\mu_j | \sum \mu_j = 1, \ \mu_j \in [0,1]\}$ is convex. Hessian is positive definite.Constraints g_j's are convex and constraint $h \ : \ \mathbb{1}^T \mu - 1 = 0$ is affine. It satisfies all the conditions for optimality.

1-dim	***N-dim***	***Theorem (CoOpt)***
$(f^{'}(x) - f^{'}(y))(x-y) \geq 0$	$(\nabla f(x) - \nabla f(y))^T(x-y) \geq 0$	51 / 78
$f(y) \geq f(x) + f^{'}(x)(y-x)$	$f(y) \geq f(x) + \nabla^T f(x)(y-x)$	46 / 75
$f^{''} \geq 0$	$\nabla^2 f(x) \succcurlyeq 0$	51 / 52 / 79 (Sufficient)
$f^{''} \geq 0$	$\nabla^2 f(x) \succcurlyeq 0$	51 / 52 / 79 (Necessary)

Table 4: Necessary and Sufficient conditions for Optimility

9.5 Duality

Dual function

Let L be the primal function given by:

$$L(\mu, \alpha, \beta, \lambda) = -\sum_{j=1}^{k} n_j \log \mu_j + \lambda(\sum_{i=1}^{k} \mu_i - 1) - \sum_i \alpha_i \mu_i + \sum_i \beta_i(\mu_i - 1)$$

Here, μ is called the primal variable and α, β, λ are called the dual variables. The dual function of L is given by:

$$L^*(\alpha, \beta, \lambda) = min_\mu L(\mu, \alpha, \beta, \lambda)$$

This dual objective function is always concave. Also note the following

- For a concave problem : Local Max = Global Max
- For a convex problem : Local Min = Global Min

Duality Gap

Consider the primal problem:

$$\begin{aligned} p^* = min - \sum n_j \log \mu_j \\ \text{s.t. } \mu_i \in [0,1] \\ \sum \mu_i = 1 \end{aligned}$$

The corresponding dual problem is given by

$$d^* = max_{\alpha_j \geq 0, \beta_j \geq 0} L^*(\alpha, \beta, \lambda)$$

We can show that $p^* \geq d^*$ and the expression $p^* - d^*$ is called the *duality gap*.

Exercise

Prove the functions in *CoOpt, Table 4* are convex.

CS 725 : FOUNDATIONS OF MACHINE LEARNING AUTUMN 2011

Lecture 12: Naive Bayes Classifier

Instructor: *Ganesh Ramakrishnan* Date: *30/08/2011*

COMPUTER SCIENCE & ENGINEERING INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY

10 Naive Bayes Classifier

Until now we only considered the case of only one feature function per data tuple. Now we will consider the case where each data tuple itself contains a lot of different features. The notation of this is as follows

X_i : **Data tuple that belongs to class c_j with probability $Pr(c_j)$ (class prior)**

$\phi_1(X_i), \phi_2(X_i) \dots \phi_m(X_i)$: **Set of m features on the data tuple**

$[V_1^1 \dots V_{k_1}^1][V_1^2 \dots V_{k_2}^2] \dots [V_1^m \dots V_{k_m}^m]$: **Set of values taken by each feature function**

$[\mu_{1,j}^1 \dots \mu_{k_1,j}^1][\mu_{1,j}^2 \dots \mu_{k_2,j}^2] \dots [\mu_{1,j}^m \dots \mu_{k_m,j}^m]$: **Set of parameters of distribution that characterizes the values taken by a feature for a particular class**

As an example from document classification task we have the following:

X_i : A particular document

c_j : Document is categorized as sports document

ϕ_1 : Some word form for hike; Corresponding values V_i^1 can be *hikes, hiking, ...*

ϕ_k : Presence or absence of word *race*. Here values are binary ($|k_m| = 2$)

10.1 Problem formulation

We are interested in the following probability distribution:

$$Pr\Big(\phi_1(X_i) \dots \phi_m(X_i) | c_j\Big)$$

By the naive bayes assumption which states that *features are conditionally independent given the class*, we have the following

$$Pr\Big(\phi_1(X_i) \dots \phi_m(X_i) | c_j\Big) = \prod_{l=1}^{m} Pr\Big(\phi_l(X_i) | c_j\Big)$$

As the name suggests, this is a naive assumption especially if there is lots of training data. However, it works very well in practice. If use a lookup table using all the feature values together, we have $[k_1 * k_2 \cdots * k_m * (\#classes)]$ parameters to estimate. In contrast, in the naive bayes we have to estimate $[(k_1 + k_2 + \cdots + k_m) * (\#classes)]$ parameters. This is a great simplification in size of parameter space.

To find the "best" class given x_i, i.e.

$$\begin{aligned} Pr(c_j|X_i) &= \frac{Pr\Big(c_j, \phi_1(X_i), \ldots, \phi_m(X_i)\Big)}{\sum_{c'_j} Pr\Big(c'_j, \phi_1(X_i), \ldots, \phi_m(X_i)\Big)} \\ &= \frac{Pr(c_j) \prod_{l=1}^{m} Pr\Big(\phi_l(X_i)|c_j\Big)}{Pr(\phi_i(X_i) \ldots \phi_m(X_i))} \end{aligned}$$

We have to minimize the "risk".

Misclassification Risk

A risk is a function which we try to minimize to assign the best class label to a datapoint. The treatment of risk minimization is related to *decision theory.* This function R takes as arguments a policy p and the datapoint X_i. For the misclassification risk, the policy is *the point came from c_i but the algorithm assigned it class c_j*. We have the following expression:

$$R(p, X_i) = \sum_{j=1}^{|c|} pol(c_j)(1 - Pr(c_j|X_i))$$

where $pol(c_j)$ is given by

$$\begin{aligned} pol(c_j) &= 1 \quad \text{if policy says classify into } c_j \\ pol(c_i) &= 0 \quad \forall i \neq j \end{aligned}$$

As per misclassification risk (i.e. in order to minimize it) we have

$$\begin{aligned} c^* &= \arg\max_{c_j} \frac{Pr(c_j) \prod_{l=1}^{m} Pr\Big(\phi_l(X_i)|c_j\Big)}{Pr(\phi_i(X_i) \ldots \phi_m(X_i))} \\ &= \arg\max_{c_j} Pr(c_j) \prod_{l=1}^{m} Pr\Big(\phi_l(X_i)|c_j\Big) \\ &= \arg\max_{c_j} Pr(c_j) \prod_{l=1}^{m} \Big[\sum_{p=1}^{k_l} \delta\Big(\phi_l(X_i), V_p^l\Big) * \mu_{p,j}^l\Big] \end{aligned}$$

where $Pr(c_j)$ is the class prior as defined before.

10.2 Estimating the Parameters

We have the data set D given by

$$D = \Big[(X_1, c_{X_1}), \ldots (X_n, c_{X_n})\Big]$$

The maximum likelihood estimators are denoted by $\left[\hat{\mu}_{ML}, \hat{Pr}_{ML}(c_j)\right]$. They are calculated as follows

$$\begin{aligned}\hat{\mu}_{ML}, \hat{Pr}(c_j) &= \underset{\mu, Pr(c)}{\arg\max} \prod_{i=1}^{m} Pr(c(X_i)) * \prod_{l=1}^{m} Pr(\phi_l(X_i)|c(X_i)) \\ &= \underset{\mu, Pr(c)}{\arg\max} \prod_{i=1}^{|c|} \big(Pr(c_j)\big)^{\#c_j} * \prod_{l=1}^{m} \prod_{p=1}^{k_l} \big(\mu_{p,j}^l\big)^{n_{p,j}^l}\end{aligned}$$

where,

$$\begin{aligned}\#c_j &= \text{ No. of times } c(X_i) = c_j \text{ across all } i\text{'s in the dataset} \\ n_{p,j}^l &= \text{ No. of times } \phi_l(X_i) = V_p \text{ and } c(X_i) = c_j \text{ across all the } i\text{'s} \\ n_{p,j}^l &= \sum_i \delta\big(\phi_l(X_i), V_p^l\big)\delta\big(c(X_i), c_j\big) \\ Pr(c(X_i)) &= \sum_{j=1}^{|c|} \delta\big(c(X_i), c_j\big) Pr(c_j) \\ Pr(\phi_l(X_i)|c(X_i)) &= \sum_{p=1}^{k_l} \delta\big(\phi_l(X_i), V_p^l\big) * \mu_{p,c(X_i)}^l\end{aligned}$$

So, the final log-likelihood objective function is:

$$\boxed{\underset{\mu, Pr(c)}{\arg\max} \Big[\sum_{j=1}^{|c|} (\#c_j) \log Pr(c_j) + \sum_{l=1}^{m} \sum_{p=1}^{k_p} n_{p,j}^l \log(\mu_{p,j}^l)\Big]} \tag{11}$$

under the constraints

$$\begin{aligned}\sum_{j=1}^{|c|} Pr(c_j) &= 1 \\ \sum_{p=1}^{k_l} \mu_{p,j}^l &= 1 \quad \forall l, j \\ Pr(c_j) &\in [0,1] \quad \forall j \\ \mu_{p,j}^l &\in [0,1] \quad \forall p, l, j\end{aligned}$$

Intuitively, working out the KKT conditions on the above objective function, we get the estimators as follows

$$\begin{aligned}\hat{\mu}_{p,j}^l &= \frac{n_{p,j}^l}{\sum_{p'=1}^{n} n_{p',j}^l} \\ \widehat{Pr}(c_j) &= \frac{\#c_j}{\sum_k \#c_k}\end{aligned}$$

However, there is a problem with the above estimators. To illustrate that let us take the example of a pair of coins being tossed. Let coin1 be tossed 5 times and suppose we get heads all the 5 times. Then if feature $\phi = heads$, $\mu^1_{h,1} = 1$ and $\mu^1_{t,1} = 0$. But we know that

$$Pr(coin1|\phi(X) = t) \propto Pr(coin1) * \mu^1_{t,1}$$

But from the above parameters, the probability $Pr(coin1|\phi(X) = t) = 0$. This may fine if we have made a million observations. However, in this case it is not correct.

We can modify Equation 11 and get a slightly modified objective function that can alleviate this problem in a hacky manner by considering:

$$\boxed{\arg\max_{\mu, Pr(c)} \Big[\sum_{j=1}^{|c|} (\#c_j) \log Pr(c_j) + \sum_{l=1}^{m} \sum_{p=1}^{k_p} n^l_{p,j} \log(\mu^p_{p,j}) - \sum_{p,j,l} \left(\mu^l_{p,j}\right)^2 \Big]} \tag{12}$$

The new term introduced in Equation 12 is called the *regularization* term. It is done to introduce some bias into the model.

A small note: Suppose $\phi_l(X_i) \sim Ber(\mu^l_{1,c(X_i)} \dots \mu^l_{k_l,c(X_i)})$, then $\hat{\mu}^l_{p,j}$ is a random variable since it is a function of the X_i's and it will belong to the normal distribution as $n \to \infty$

Exercise

1. Work out the KKT conditions for Equation 11
2. Work out the KKT conditions for Equation 12
3. Is the objective function in Equation 12 concave ?

Further Reading

1. `http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf`
2. `http://www.cse.iitb.ac.in/~cs725/notes/classNotes/misc/CaseStudyWithProbabilisticModels.pdf`, Sections 7.6.1, 7.7.1, 7.7.2, 7.7.3, 7.4, 7.7.4
3. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.2111&rep=rep1&type=pdf`
4. *Christopher Bishop*, PRML (Pattern Recognition and Machine Learning), Sections 2.1.1, 2.2.1, 2.4.2 and 2.4.3

10.3 Conjugate prior

Suppose we have a multivariate bernoulli distribition of the μ's and let $Pr(\mu = \mu_1) = Pr(\mu = \mu_2) = \dots Pr(\mu = \mu_k) = \frac{1}{k}$. As an example condider the toss of a dice. Suppose at ∞, all observations are say a particular value V_i then, we will have $Pr(\mu = \mu_1) = 0, \dots Pr(\mu = \mu_i) = 1 \dots Pr(\mu = \mu_k) = 0$

Using Bayes rule

$$Pr([\mu_1 \dots \mu_k]|D) = \frac{Pr(D|\bar{\mu})Pr(\bar{\mu})}{\sum_{\bar{\mu'}} Pr(D|\bar{\mu'})Pr(\bar{\mu'})}$$

If $Pr(\mu)$ has a form such that $Pr(\mu|D)$ has the same form, we say that $Pr(\mu)$ is the *conjugate prior* to the distribution defining $Pr(D|\mu)$.

Some of the conjugate priors that we will see in the next class

- Dirichlet and Multivariate Bernoulli
- Beta and Bernoulli
- Gaussian and Gaussian

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 13: Naive Bayes (cont.)

Instructor: *Ganesh Ramakrishnan* — Date: *02/09/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

Naive Bayes with Dirichlet prior[2]

Naive Bayes assumption: $\Pr(\phi_i(x_i)...\phi_n(x_i)|c_j) = \prod_{l=1}^{m} Pr(\phi_l(x_i)|c_j)$
Minimum Likelihood Estimation: $(\hat{\mu^l_{p,j}})_{ML} = \frac{n^l_{p,j}}{\sum_{q=1}^{k_l} n^l_{q,j}}$
Now,
Prior $P(\mu) \in [0,1] \& \int_0^1 p(\mu)d\mu = 1$
Now, $P(\mu|X_1,X_2,..X_n) = \frac{P(X_1,X_2,..X_n|\mu).P(\mu)}{\int_0^1 P(X_1,X_2,..X_n|\mu).P(\mu)d\mu}$
$= \frac{\mu^{n_1}(1-\mu)^{n_2}}{\int \mu^{n_1}(1-\mu)^{n_2}d\mu}$
$= \mu^{n_1}(1-\mu)^{n_2}\frac{(n_1+n_2+1)!}{n_1!n_2!}$
HW. If $P(\mu)$ had the form $\frac{\mu^{a-1}(1-\mu)^{b-1}(a=b-1)!}{(a-1)!(b-1)!}$ what will be the form of $P(\mu|D)$? $P(\mu|X_1,X_2,..X_n) = \frac{\mu^{a-1+n_1}(1-\mu)^{b-1+n_2}(n_1+n_2+a+b-1)!}{(n_1+a-1)!(n_2+b-1)!}$ $Beta(\mu|a+n_1,b+n_2)$
so $P(\mu=1)$ $Beta(\mu|1,1)$ Expected value of Beta $E(Beta(\mu|a,b)) = \frac{a}{a+b}$
Why it is rasonable:

- $E[\mu]_{Beta(\mu|a,b)} = \frac{a}{a+b}$ is intuitive
- a=b=1 gives uniform distribution
- Form of the posterior and prior are the same.
- As $n_1+n_2 \leftarrow \infty$, spread of the distribution $\leftarrow 0$, a and b becomes immaterial. $(\frac{n_1+a}{n_1+n_2+a+b} = \frac{n_1}{n_1+n_2})$
  $\underset{B(\mu|n_1+a,n_2+b)}{E(\mu)} \quad \leftarrow \hat{\mu_{ML}}$
  $\hat{\mu_{MAP}} = \underset{\mu}{argmax} P(\mu|D) = \frac{a+n_1-1}{a+n_1+b+n_2-2}$
  As $n_1, n_2 \leftarrow \infty, \hat{\mu_{MAP}} = \hat{\mu_{ML}}$

$P(X_{n+1},..X_{n+t}|\mu) = \prod_{i=1}^{t} P(X_{i+1}|\mu)$
$P(X_{n+1}|X_1..X_n) = \int P(X_{n+1}|\mu)P(\mu|X_1..X_n)d\mu = \int \mu^{X_n+1}(1-\mu)^{1-X_{n+1}}P(\mu|X_1..X_n)d\mu$
$= E[\mu]$ if $X_{n+1} = 1$
$= 1 - E[\mu]$ if $X_{n+1} = 0$
$Beta_{(\mu_1,\mu_2|a,b)} = \frac{\mu_1\mu_2(n_1+n_2-1)!}{(a-1)!(b-1)!}$
$Dir(\mu_1,\mu_2,..\mu_k|a_1,a_2,..a_k) = \frac{\prod_i \mu_i^{a_i-1}\sqrt{\sum_j a_j}}{\prod_j \sqrt{a_j}}$

[2]**lecture scribed by Amrita and Kedhar with inputs from Ajay Nair**

$E(\mu)_{Dir} = [\frac{a_1}{\sum_i a_i} \frac{a_2}{\sum_i a_i} .. \frac{a_k}{\sum_i a_i}]^T$
Expression for $\mu_{B\hat{a}yes} = E(\mu) = [\frac{a_1+n_1}{\sum_i a_i+n_i} \frac{a_2+n_2}{\sum_i a_i+n_i} .. \frac{a_k+n_k}{\sum_i a_i+n_i}]^T$
Expression for $\mu_{\hat{M}L} = E(\mu) = [\frac{n_1}{\sum_i n_i} \frac{n_2}{\sum_i n_i} .. \frac{n_k}{\sum_i n_i}]^T$
Expression for $\mu_{\hat{M}AP} = E(\mu) = [\frac{a_1+n_1-1}{(\sum_i a_i+n_i)-K} \frac{a_2+n_2-1}{(\sum_i a_i+n_i)-K} .. \frac{a_k+n_k-1}{(\sum_i a_i+n_i)-K}]^T$
$P(X_{n+1}|X_1..X_n) = [E(\mu)]$ if $X_{n+1} = V_i$
$Dir_{l,j}(\mu^l_{1_ij}..\mu^l_{l_ij}|a^l_{1_ij}..a^l_{1_ij}) = \frac{\prod_i (\mu^l_{i,j})^{a^l_{i,j}-1}}{\prod_i \Gamma(a^l_{i,j})} \Gamma(a^l_{i,j} + (a^l_{2,j}) + ..(a^l_{k,j})$
$E(\mu)_{Dir} = [\frac{a^l_{1,j}}{\sum_i a^l_{i,j}} \frac{a^l_{k,j}}{\sum_i a^l_{i,j}} .. \frac{a^l_{k,j}}{\sum_i a^l_{i,j}}]^T$
$(\mu^l_j)_{Bayes} = [\frac{a^l_{1,j}+n^l_{1,j}}{\sum_i a^l_{i,j}+n^l_{i,j}} \frac{a^l_{2,j}+n^l_{2,j}}{\sum_i a^l_{i,j}+n^l_{i,j}} .. \frac{a^l_{k,j}+n^l_{k,j}}{\sum_i a^l_{i,j}+n^l_{i,j}}]^T$
$(\mu^l_j)_{MAP} = [\frac{a^l_{1,j}+n^l_{1,j}}{\sum_i(a^l_{i,j}+n^l_{i,j})-K_l} \frac{a^l_{2,j}+n^l_{2,j}}{(\sum_i a^l_{i,j}+n^l_{i,j})-K_l} .. \frac{a^l_{k,j}+n^l_{k,j}}{(\sum_i a^l_{i,j}+n^l_{i,j})-K_l}]^T$
$(\mu^l_j)_{Bayes} = [\frac{a^l_{1,j}+n^l_{1,j}}{\sum_i a^l_{i,j}+n^l_{i,j}} \frac{a^l_{2,j}+n^l_{2,j}}{\sum_i a^l_{i,j}+n^l_{i,j}} .. \frac{a^l_{k,j}+n^l_{k,j}}{\sum_i a^l_{i,j}+n^l_{i,j}}]^T$

Assume X is the event of a coin toss. Let X1=0 (TAILS say), X2=1, X3=0, X4=1, X5=1. We are interested in predicting the event X6=1 given the above. This can be calculated by different approaches. The ML, MAP and the Bayes Estimator are called the pseudo Bayes, and Bayesian estimator is called the pure Bayes.

Maximum likelihood

$\mu_{\hat{M}L}$ is the probability of $X = 1$ from the data.

$$P(X6|X1..X5) = \mu_{\hat{M}L}{}^{X6}(1 - \mu_{\hat{M}L})^{(1-X6)}$$

MAP

$\mu_{\hat{M}AP}$ is the probability of $X = 1$ from the data.

$$P(X6|X1..X5) = \mu_{\hat{M}AP}{}^{X6}(1 - \mu_{\hat{M}AP})^{(1-X6)}$$

Bayes Estimator

$\mu_{b\hat{a}yes}$ is the probability of $X = 1$ from the data.

$$P(X6|X1..X5) = \mu_{b\hat{a}yes}{}^{X6}(1 - \mu_{b\hat{a}yes})^{(1-X6)}$$

Bayesian method

$$P(X6|X1..X5) = \int_0^1 \mu^{X6}(1-\mu)^{(1-X6)} P(\mu|X1..X5) \mathrm{d}\mu$$

The explanation for this equation is as follows:

$$P(X6|X1..X5) = \int \frac{P(X6|\mu, X1, ..X5) P(\mu|X1..X5) P(X1..X5) \mathrm{d}\mu}{P(X1..X5)}$$

this marginalises on the probability μ. Simplifying futher,

$$P(X6|X1..X5) = \int P(X6|\mu, X1, ..X5)P(\mu|X1..X5)\mathrm{d}\mu$$

Thus

$$P(X6|X1..X5) = \int_0^1 \mu^{X6}(1-\mu)^{(1-X6)}P(\mu|X1..X5)\mathrm{d}\mu$$

CS 725 : Foundations of Machine Learning Autumn 2011

Lecture 14: Naive Bayes Classifier (cont.)

Instructor: *Ganesh Ramakrishnan* Date: *06/09/2011*
Computer Science & Engineering Indian Institute of Technology, Bombay

10.4 Naive Bayes with Dirichlet Priors

Uptil now the set of distribitions we have seen are as follows:

1. Bernoulli distribution (whose conjugate is Beta)
2. Multivariate Bernoulli distribution (whose conjugate is Dirichlet)
3. Naive Bayes with Multivariate Bernoulli distribution
4. Naive Bayes with Dirichlet prior on Multivariate Bernoulli distribution (whose prior is per-class set of Dirichlet)

The setting for the Naive Bayes with Dirichlet prior on Multivariate Bernoulli distribution is as follows:

- For each data point X_i which belongs to class c_j there are a set of m features given by $\phi_1(X_i)\dots\phi_l(X_i)\dots\phi_m(X_i)|c_j$
- Each of these features have a probability distribution given by $p(\mu_j^1)\dots p(\mu_j^l)\dots p(\mu_j^m)$ where $p(\mu_j^1)\dots p(\mu_j^m)$ are the parameters to be determined from the data.

Final Estimators

For the problem under consideration, the final estimators are as follows:

$$\left(\hat{\mu}_{p,j}^l\right)_{MLE} = \frac{n_{p,j}^l}{\sum_{q=1}^{k_l} n_{q,j}^l}$$

$$\left(\hat{\mu}_{p,j}^l\right)_{Bayes} = \left[\mathcal{E}(\mu|X_1\dots X_n)\right]_{p,j}^l = \frac{n_{p,j}^l + a_{p,j}^l}{\sum_{q=1}^{k_l}\left(n_{q,j}^l + a_{q,j}^l\right)} \text{ also called } \textit{Laplace Smoothing}$$

$$\left(\hat{\mu}_{p,j}^l\right)_{MAP} = \left[\arg\max p(\mu|X_1\dots X_n)\right]_{p,j}^l = \frac{n_{p,j}^l + a_{p,j}^l - 1}{\sum_{q=1}^{k_l}\left(n_{q,j}^l + a_{q,j}^l - 1\right)}$$

where $a_{p,j}^l$ are multivariate dirichlet prior parameters

Essentially we are trying to find $p(x|D)$ but we approximate it to $p(x|\hat{(}\mu)$ and we determine various types of estimators from this like MLE, MAP and Bayes.

If we consider a purely bayesian perspective from first principles, we have:

$$\begin{aligned} p(x|D) &= \int_\mu p(x,\mu|D)d\mu \\ &= \int_\mu p(x|\mu, D) * p(\mu|D)d\mu \\ &= \int_\mu p(x|\mu) * p(\mu|D)d\mu \text{ From the i.i.d assumption} \end{aligned}$$

So finally we have

$$\boxed{p(x|D) = \int_\mu \prod_{i=1}^{k} \mu_i^{\delta(x,V_i)} Dir(n_1 + a_1, \dots n_k + a_k)d\mu_1 \dots d\mu_k} \tag{13}$$

$\hat{\mu}_{MLE}$ is the loosest approximation from a Bayesian perspective. (since there is no prior in MLE). For instance, in coin toss example, if we don't see tails $p(tails) = 0$ in MLE. In practice, $\hat{\mu}_{MAP}$ and $\hat{\mu}_{Bayes}$ are the most important estimators.

10.5 Naive Bayes with Gaussian Priors

Until now we assumed that each feaure can take k different discrete values. Now we extend our notions to continuous distributions. Specifically we consider Gaussian or Normal Distribution for our exposition here. The treatment for continuous distributions closely follows that of our discrete case. It is as follows:

1. Gaussian
2. Independent Gaussian
3. Naive Bayes with Independent Gaussians $(\phi_1(X_i) \dots \phi_m(X_i)|c_j)$
4. Naive Bayes with Gaussian priors on Independent Gaussians
   $\phi_1(X_i) \dots \phi_m(X_i)|c_j$
   $p(\mu_j^1) \dots\dots p(\mu_j^m)$

As an aside, we will use the following conventions

- $\hat{\mu}(X_1 \dots X_n) \sim$ estimator. It is a random variable and will have a distribution
- $\hat{\mu}(x_1 \dots x_n) \sim$ estimate. It is a value.

Gaussian

Let us consider the first case of X_is following a normal distribution. , The pdf of a normal distribution is

$$p(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Given $D = \{X_1, \dots X_n\}$, We have

$$LL(\mu, \sigma^2|D) = \log\Big[\prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(X_i-\mu)^2}{2\sigma^2}}\Big] \tag{14}$$

$$
\begin{aligned}
&= -\sum_{i=1}^{n}\Big[\frac{(X_i-\mu)^2}{2\sigma^2} - n\log\sigma\sqrt{2\pi}\Big] \\
\{\hat{\mu}_{ML}, \hat{\sigma}^2_{ML}\} &= \arg\max_{\mu,\sigma^2} -\sum_{i=1}^{n}\Big[\frac{(X_i-\mu)^2}{2\sigma^2} - n\log\sigma - n\log\sqrt{2\pi}\Big]
\end{aligned}
$$

The critical question is whether the objective function in Equation 14 concave ? As an exercise compute the Hessian of Eq. 14 and check if it is negative semi-definite. Once this is ascertained, we can apply the KKT conditions for optimality:

$$
\nabla LL = \begin{bmatrix} -\sum_{i=1}^{n}\frac{(X_i-\hat{\mu})^2}{\hat{\sigma}^2} \\ \sum_{i=1}^{n}\frac{(X_i-\hat{\mu})^2}{\hat{\sigma}^3} - \frac{n}{\hat{\sigma}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
$$

$$
\Rightarrow \sum_{i=1}^{n} = n\hat{\mu}_{MLE}
$$

$$
\boxed{\hat{\mu}_{MLE} = \frac{\sum X_i}{n} \qquad \sim \mathcal{N}(\mu, \frac{\sigma^2}{n})}
$$

Also

$$
\Rightarrow \sum_{i=1}^{n}\frac{(X_i-\hat{\mu})^2}{\hat{\sigma}^3} = \frac{n}{\hat{\sigma}}
$$

$$
\boxed{\hat{\sigma}^2_{MLE} = \sum_{i=1}^{n}\frac{(X_i-\hat{\mu})^2}{n} \qquad \sim \alpha\chi^2_{n-1}}
$$

Priors for μ (σ^2 is fixed and known)

$$
\mu \sim \mathcal{N}(\mu_0, \sigma_0)
$$

$$
\begin{aligned}
p(\mu|X_1,\dots X_n) &\propto p(X_1,\dots X_n|\mu)p(\mu) \\
&= \Big(\frac{1}{\sigma\sqrt{2\pi}}\Big)^n \prod_{i=1}^{n} e^{-\frac{(X_i-\mu)^2}{2\sigma^2}} e^{\frac{(\mu-\mu_0)^2}{2\sigma_0^2}} \\
&\propto e^{\frac{(\mu-\mu_n)^2}{2\sigma_n^2}} \quad \text{(this can be proved)} \\
&\text{where } \mu_n = \Big(\frac{\sigma^2}{n\sigma_0^2+\sigma^2}\Big)\mu_0 + \Big(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\Big)\hat{\mu}_{MLE} \\
&\text{and } \frac{1}{\sigma_n^2} = \frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}
\end{aligned}
$$

$$
Pr(\mu|X_1,\dots X_n) \sim \mathcal{N}(\mu_n, \sigma_n^2)
$$

As $n \to \infty$, μ tends to assume only $\hat{\mu}_{MLE}$ i.e. spread is zero. So $Pr(\mu|X_1,\dots X_n) \sim \mathcal{N}(\hat{\mu}_{MLE}, 0)$. Also both $\hat{\mu}_{Bayes}$ and $\hat{\mu}_{MAP}$ peak at μ_n

Naive Bayes with Independent Gaussian

Now let us consider the following:

$$P(X_i|c_j, \bar{\mu}_j^l, (\bar{\sigma}_j^l)^2)$$
$$\text{where each } X_i \text{ has m features} \phi_1(X_i) \dots \phi_m(X_i)$$

Each of these features follow a gaussian distribution.
By the naive bayes assumption, we have :

$$Pr(X_i|c_j, \bar{\mu}_j^l, (\bar{\sigma}_j^l)^2) = \prod_{t=1}^{m} Pr(\phi_t(X_i)|c_j, \bar{\mu}_j^l, (\bar{\sigma}_j^l)^2)$$

The ML estimators are as follows:

$$(\hat{\mu}_j^l)_{ML} = \frac{\sum_{i=1}^{n} \phi_l(X_i)}{n}$$
$$(\hat{\sigma_j^l})^2_{ML} = \frac{\sum_{i=1}^{l} \left(\phi_l(X_i) - \hat{\mu_j^l}\right)^2}{n}$$

For fixed $(\sigma_j^l)^2$ in Bayesian setting and $\mu_j^l \sim \mathcal{N}(\mu_{0,j}^l, (\sigma_{0,j}^l)^2)$, the MAP and Bayes estimators for mean are as follows:

$$\boxed{(\hat{\mu_j^l})_{Bayes} = (\hat{\mu_j^l})_{MAP} = \mu_n = \left(\frac{(\sigma_j^l)^2}{n(\sigma_{0,j}^l)^2 + (\sigma_0^l)^2}\right)\mu_{0,j}^l + \left(\frac{n(\sigma_{0,j}^l)^2}{n(\sigma_{0,j}^l)^2 + (\sigma_j^l)^2}\right)(\hat{\mu}_j^l)_{MLE}}$$

As an exercise write down the expression for $(\hat{\sigma}_j^l)_{MAP}$.

Exercise

1. Compute the form of Equation 13
2. Compute the Hessian of Equation 14. Prove that the function in Eq. 14 is concave.
3. Derive the expression for $(\hat{\sigma}_j^l)_{MAP}$ in the Naive Bayes with independent Gaussians' setting.

CS 725 : Foundations of Machine Learning | Autumn 2011

Lecture 15: Nature of Separating Surfaces, Multivariate Gaussian Gaussian Discriminant Analysis

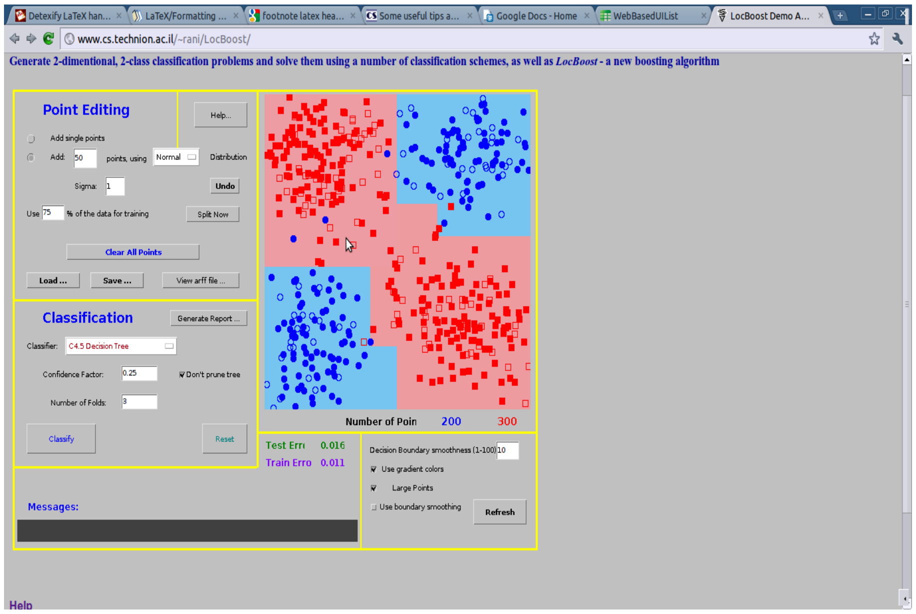
Instructor: *Ganesh Ramakrishnan* | Date: *09/09/2011*

Computer Science & Engineering | Indian Institute of Technology, Bombay

10.6 Nature of Separating Surfaces

Decision Trees

In the case of decision trees, the separating surface was intersection of axis parallel hyperplanes. It is as shown in Figure 14.



Generate 2-dimentional, 2-class classification problems and solve them using a number of classification schemes, as well as *LocBoost* - a new boosting algorithm

Add single points

Add: 50 points, using Normal Distribution

Sigma: 1

Use 75 % of the data for training

Classifier: C4.5 Decision Tree

Confidence Factor: 0.25

☑ Don't prune tree

Number of Folds: 3

Number of Poin 200 300

Test Erro 0.016

Train Erro 0.011

Decision Boundary smoothness (1-100) 10

☑ Use gradient colors

☑ Large Points

☐ Use boundary smoothing

Figure 14: Decision Surface of DTrees

Naive Bayes with Bernoulli

Here $\forall x$ in decision surface,

$$\begin{aligned}
&p(c_{j_1}|x) = p(c_{j_2}|x) \quad \text{for some 2 classes } c_{j_1} \text{ and } c_{j_2}\\
&\Rightarrow p(x|c_{j_1})p(c_{j_1}) = p(x|c_{j_2})p(c_{j_2})\\
&\Rightarrow \prod_{l=1}^{m}\prod_{p=1}^{k_l}\left(\mu_{p,j_1}^{l}\right)^{\delta(\phi_l(x),V_p^l)} p(c_{j_1}) = \prod_{l=1}^{m}\prod_{p=1}^{k_l}\left(\mu_{p,j_2}^{l}\right)^{\delta(\phi_l(x),V_p^l)} p(c_{j_2})\\
&\Rightarrow \sum_{l=1}^{m}\sum_{p=1}^{k_l}\delta(\phi_l(x),V_p^l)\Big[\log\left(\mu_{p,j_1}^{l}\right) - \log\left(\mu_{p,j_2}^{l}\right)\Big] + \log\frac{p(c_{j_1})}{p(c_{j_2})} = 0 \quad \text{(taking log on both sides)}
\end{aligned}$$

If $\phi_l = 0|1$,

$$\sum_{l=1}^{m}\sum_{p=1}^{k_l}\phi_l(x)\Big[\log\left(\mu_{p,j_1}^{l}\right) - \log\left(\mu_{p,j_2}^{l}\right)\Big] + \log\frac{p(c_{j_1})}{p(c_{j_2})} = 0$$

This surface looks like a general hyperplane. It is linear in ϕ's

Naive Bayes with Independent Univariate Gaussian

Here $\forall x$ in decision surface,

$$\begin{aligned}
&p(c_{j_1}|x) = p(c_{j_2}|x) \quad \text{for some 2 classes } c_{j_1} \text{ and } c_{j_2}\\
&\Rightarrow p(x|c_{j_1})p(c_{j_1}) = p(x|c_{j_2})p(c_{j_2})\\
&\Rightarrow \left[\prod_{l=1}^{m}\Big(\frac{1}{\sigma_{j_1}^{l}\sqrt{2\pi}}e^{\frac{-(\phi_l(x)-\mu_{j_1}^{l})^2}{2(\sigma_{j_1}^{l})^2}}\Big)\right]p(c_{j_1}) = \left[\prod_{l=1}^{m}\Big(\frac{1}{\sigma_{j_2}^{l}\sqrt{2\pi}}e^{\frac{-(\phi_l(x)-\mu_{j_2}^{l})^2}{2(\sigma_{j_2}^{l})^2}}\Big)\right]p(c_{j_2})\\
&\Rightarrow \sum_{l=1}^{m}\Big[\frac{-(\phi_l(x)-\mu_{j_1}^{l})^2}{2(\sigma_{j_1}^{l})^2} + \frac{(\phi_l(x)-\mu_{j_2}^{l})^2}{2(\sigma_{j_2}^{l})^2}\Big] + \log\frac{p(c_{j_1})}{p(c_{j_2})} + \sum_{l=1}^{m}\Big[\log\frac{\sigma_{j_2}^{l}}{\sigma_{j_1}^{l}}\Big] = 0
\end{aligned}$$

The decision surface is quadratic in ϕ's. For the same set of points as in Figure 14, the decision surface for Naive Bayes with Gaussians is as shown in Figure 15. A very clear decision surface does not emerge for this case unlike the case for decision trees. For another set of points the decision surface looks as shown in Figure 16.

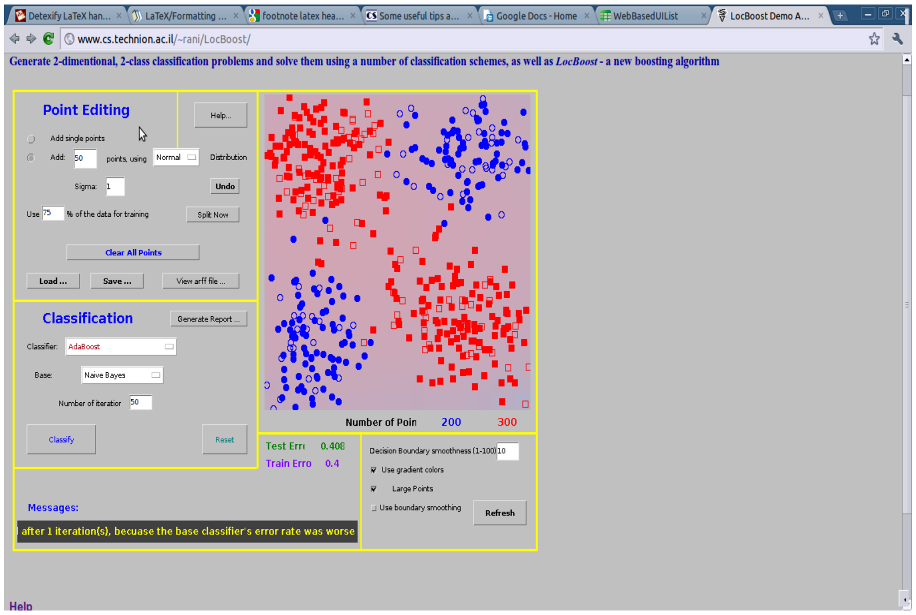
10.7 Multivariate Gaussian

Next let us consider the case of multivariate normal distribution also known as multivariate gaussian. Consider the features as

$$\phi_1(X)\dots\phi_m(X) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

where $\boldsymbol{\mu}$ is a $1 \times x$ vector and $\boldsymbol{\Sigma}$ is a $m \times m$ matrix called the co-variance matrix. (In this section μ and Σ are vectors and matrix respectively even if they are not denoted in bold-face)

$$\mathcal{N}(\boldsymbol{x}_{1\times m}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{m}{2}}|\Sigma|^{\frac{1}{2}}}e^{-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}}$$



www.cs.technion.ac.il/~rani/LocBoost/

Generate 2-dimentional, 2-class classification problems and solve them using a number of classification schemes, as well as *LocBoost* - a new boosting algorithm

Point Editing

Add single points

Add: 50 points, using Normal Distribution

Sigma: 1

Use 75 % of the data for training

Classification

Classifier: AdaBoost

Base: Naive Bayes

Number of iteratior 50

Number of Poin 200 300

Test Erro 0.408

Train Erro 0.4

Decision Boundary smoothness (1-100) 10

[✓] Use gradient colors

[✓] Large Points

Use boundary smoothing

Messages:

after 1 iteration(s), becuase the base classifier's error rate was worse

Figure 15: Decision Surface of Naive Bayes with Gaussians

For example if $m = 2$, and

$$\mu = \begin{bmatrix} \mu_1 & \mu_2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

$$\begin{aligned} \mathcal{N}(x|\mu, \Sigma) &\propto e^{-\frac{(x_1-\mu_1)^2}{2\sigma_1^2} - \frac{(x_2-\mu_2)^2}{2\sigma_2^2}} \\ &= \mathcal{N}(x_1|\mu_1, \sigma_1^2)\mathcal{N}(x_2|\mu_2, \sigma_2^2) \end{aligned}$$

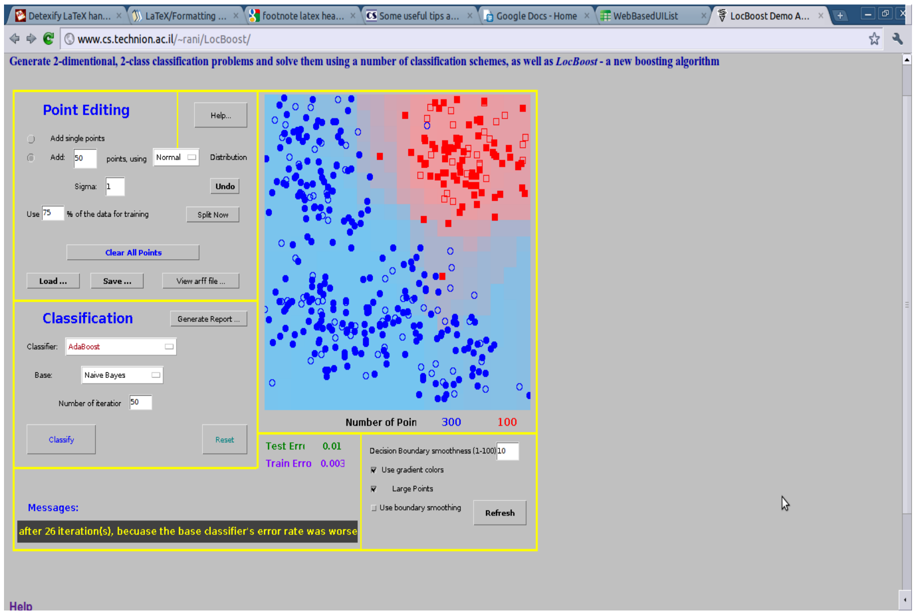
In general, if

$$\Sigma = \begin{bmatrix} \sigma_1^2 & & & \text{\huge 0} \\ & \sigma_2^2 & & \\ & & \ddots & \\ \text{\huge 0} & & & \sigma_m^2 \end{bmatrix}$$

then,

$$\mathcal{N}(x|\mu, \Sigma) = \prod_{i=1}^{m} \mathcal{N}(x_i|\mu_i, \sigma_i^2)$$

So Naive Bayes assumption is a special case of mutivatiate gaussian.



Generate 2-dimentional, 2-class classification problems and solve them using a number of classification schemes, as well as *LocBoost* - a new boosting algorithm

Add single points
Add: 50 points, using Normal Distribution
Sigma: 1
Use 75 % of the data for training

Classifier: AdaBoost
Base: Naive Bayes
Number of iteratior 50

Number of Poin 300 100

Test Erri 0.01
Train Erro 0.003

Decision Boundary smoothness (1-100) 10
☑ Use gradient colors
☑ Large Points
☐ Use boundary smoothing

Messages:
after 26 iteration(s), becuase the base classifier's error rate was worse

Figure 16: Decision Surface of Naive Bayes with Gaussians (another set of points)

Estimators for mutivariate Gaussian

Let $D = \{X_1, \ldots X_i \ldots X_n\}$ be the set of data points and let

$$\phi(X_i) = \begin{bmatrix} \phi_1(X_i) & \ldots & \phi_m(X_i) \end{bmatrix}$$

So, $\phi(X_i)$ represents a vector. MLE for μ and Σ assuming they came from the same multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$:

$$\begin{aligned} \hat{\mu}_{ML}, \hat{\Sigma}_{ML} &= \arg\max_{\mu,\Sigma} p(D) \\ &= \arg\max_{\mu,\Sigma} \prod_{i=1}^{n} \frac{1}{(2\pi)^{\frac{m}{2}} |\Sigma|^{\frac{1}{2}}} e^{\frac{-(\phi(x_i)-\mu)^T \Sigma^{-1} (\phi(x_i)-\mu)}{2}} \end{aligned}$$

$$\boxed{= \arg\max_{\mu,\Sigma} \sum_{i=1}^{m} \left[\frac{-(\phi(x_i) - \mu)^T \Sigma^{-1} (\phi(x_i) - \mu)}{2} - \frac{n}{2} \log |\Sigma| - \frac{mn}{2} \log(2\pi) \right]} \tag{15}$$

We assume symmetric Σ. It can be shown thar a non-symmetric Σ has an equivalent symmetric Σ (see further reading) . The necessary conditions for local maxima can be computed by finding the ∇LL function above. As an exercise find ∇LL for Equation 15.

The ML estimators are:

$$\hat{\mu}_{ML} = \frac{\sum_{i=1}^{n} \phi(x_i)}{n}$$

$$\hat{\Sigma}_{ML} = \frac{\sum_{i=1}^{n} \left[\big(\phi(x_i) - \hat{\mu}_{ML}\big)\big(\phi(x_i) - \hat{\mu}_{ML}\big)^T \right]}{n} \qquad \text{(sum of rank 1 matrices and is symmetric)}$$

Bias of the Estimator

By definition, an estimator $e(\theta)$ is called unbiased estimator of θ if $\mathcal{E}[e(\theta)] = \theta$.

$\hat{\mu}_{ML}$ is an unbiased estimator. This is shown as follows:

$$\mathcal{E}(\hat{\mu}_{ML}) = \frac{\sum_{i=1}^{n} \mathcal{E}(\phi(x_i))}{n} = \frac{\sum_{i=1}^{n} \mu}{n} = \mu$$

However $\hat{\Sigma}_{ML}$ is a biased estimator since it can be shown that

$$\mathcal{E}(\hat{\Sigma}_{ML}) = \frac{n-1}{n} \Sigma$$

If we define a new estimator say $\hat{\Sigma}_{new}$ which is given by:

$$\hat{\Sigma}_{new} = \frac{\sum_{i=1}^{n} (\phi(x_i - \hat{\mu}_{ML}))(\phi(x_i - \hat{\mu}_{ML}))^T}{n-1}$$

This can be shown to be an unbiased of Σ. (See further reading about the bias of an estimator.)

Exercise

1. Find ∇LL for equation 15 w.r.t $[\mu_1 \dots \mu_m \ \Sigma_{11} \dots \Sigma_{1m} \ \Sigma_{21} \dots \Sigma_{mm}]^T$
2. Were $\hat{\theta}_{ML}$, the parameters of bernoulli, multivariate bernoulli and gaussian, unbiased ?

Further Reading

1. Equivalence of symmetric and non-symmetric matrices: Chapter 2, PRML, Christopher Bishop
2. Unbiased Estimator: `http://en.wikipedia.org/wiki/Bias_of_an_estimator`

11 Gaussian Discriminant Analysis

Gaussian Discriminant Analysis is another classifier which is like Naive Bayes with Gaussian but with the naive assumption done away with. Consider the set of class labels:

$$C_1, \dots C_j, \dots C_{|c|}$$

where each of them come from a normal distribution as follows:

$$\mathcal{N}(\mu_1, \Sigma_1) \dots \mathcal{N}(\mu_j, \Sigma_j) \dots \mathcal{N}(\mu_{|C|}, \Sigma_{|C|})$$

We can show that the ML estimators are as follows:

$$(\hat{\mu_j})_{ML} = \frac{\sum_{i=1}^{n} \phi(x_i)\delta(class(x_i), c_j)}{\sum_{i=1}^{n} \delta(class(x_i), c_j)}$$

$$(\hat{\Sigma_j})_{ML} = \frac{\sum_{i=1}^{n}(\phi(x_i) - (\hat{\mu}_j)_{ML})(\phi(x_i) - (\hat{\mu}_j)_{ML})^T \delta(class(x_i), c_j)}{\sum_{i=1}^{n} \delta(class(x_i), c_j)}$$

The decision surface of this classifier is quadratic in μ.

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 16 + 17: Gaussian Discriminant Analysis, Clustering

Instructor: *Ganesh Ramakrishnan* — Date: *20/09/2011, 23/09/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

11.1 Nature of separating surface

There are 2 types of mixture of Gaussians.

1. Different μ_j's and Σ_j's. This was the case we saw in the last class. We had written down the equation for $(\hat{\mu_j})_{ML}$ and $(\hat{\Sigma_j})_{ML}$. We had also briefly noted that the decision surface is quadratic. It can also be shown from the following equation for a mixture of 2 gaussians.

$$\frac{e^{(-1/2)(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)}}{(2\pi)^{m/2}|\Sigma_1|^{1/2}} = \frac{e^{(-1/2)(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)}}{(2\pi)^{m/2}|\Sigma_2|^{1/2}}$$

The decision surface looks as shown in Figure 17

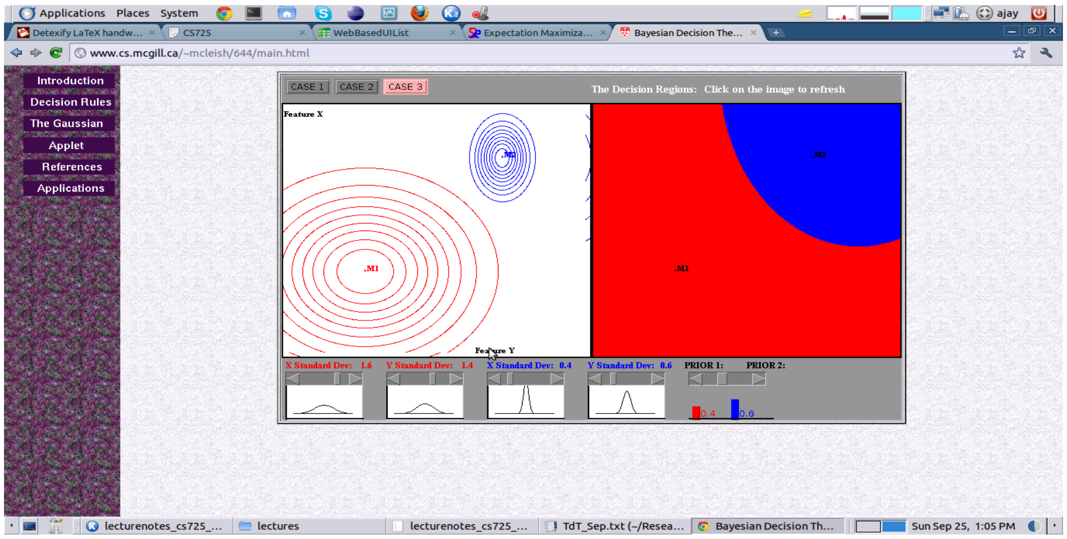


Figure 17: Decision Surface of Gaussian Mixture (Case 1)

2. Different μ_j's and shared Σ. In this case the estimators are as follows:

$$(\hat{\mu_j})_{ML} = \frac{\sum_{i=1}^{n}\phi(x_i)\delta(class(x_i),c_j)}{\sum_{i=1}^{n}\delta(class(x_i),c_j)} \qquad \text{(same as before)}$$

$$(\hat{\Sigma})_{ML} = \frac{\sum_{i=1}^{n}\left[(\phi(x_i)-(\hat{\mu_j})_{ML})^T(\phi(x_i)-(\hat{\mu_j})_{ML})\right]}{n}$$

The nature of the separating surface is linear. When we equate the following terms, $x^T\Sigma^{-1}x$ gets cancelled.

$$(1/2)(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) = (1/2)(x-\mu_2)^T\Sigma^{-1}(x-\mu_2)$$

The separating surface looks as shown in Figure 18

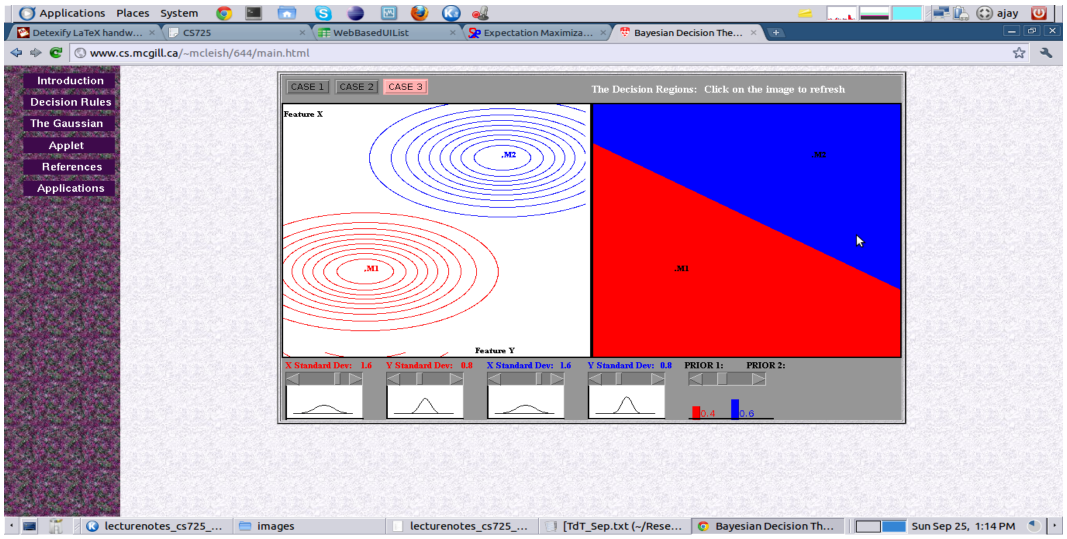


Figure 18: Decision Surface of Gaussian Mixture (Case 2)

11.2 Curse of dimensionality

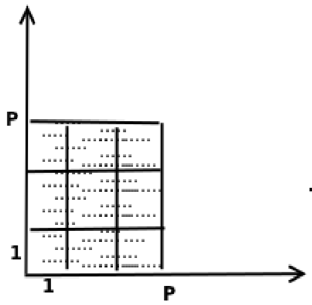
The number of parameters to be determined increase non-linearly as the number of dimensions. This is termed as the *curse of diminsionality.* For the cases from previous section, the curse of dimensionality is evidently seen in the first case. It is ameliorated in the second case. The number of parameters to be determined for the 2 cases are as follows:

1. No. of params $= \left(m + \frac{m(m+1)}{2}\right)|C|$
2. No. of params $= |C|m + \frac{m(m+1)}{2}$

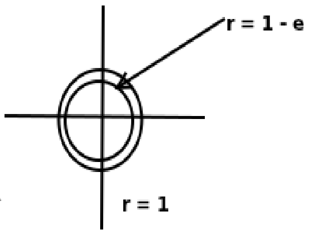
Other perspectives for curse of dimensionality

1. Consider a $p \times p$ grid of cells as shown in Figure 19a with unity being the length of each cell. In 2-dim there are p^2-cells. In m-dim there will be p^m unit cells. For a learning problem, we have to smear the data points over all the cells. Adding new diminsions (features) will make the smearing skewed (as the number of data points remain the same even in higher dimensions)

2. Consider a ring of radius $r = 1$ as shown in Figure 19b. Now fix a small ϵ value and consider a second ring of radius $1 - \epsilon$. In 2 dimensions, the ratio of area of the annulus to the entire



(a) p^2 cells



(b) $1-\epsilon$ annulus

Figure 19: Curse of Dimensionality

area is $\frac{1^2-(1-\epsilon)^2}{1^2} = 2\epsilon - \epsilon^2$. In m-dimensions it is $\frac{1^m-(1-\epsilon)^m}{1^m}$. As m grows, the volume of the annulus grows more than in lower dimensions.

For more information on curse of dimensionality, refer to *PRML, Chapter 2*

12 Unsupervised Learning

So far we have been studying *supervised learning* wherein during the training we are also provided with the class label of a particular example. The data is in the form $D(x_i, class(x_i))$. Also we assumed the generative model of classification i.e the class labels generate the examples. This is depicted as shown in Figure 20.

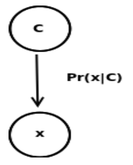


Figure 20: Generative Model

Now, we consider another type of learning in the same generative model setting. Here the class labels are not known during training. This is also called *unsupervised learning.* So the data is of the form $D(x_i)$. We will look at *clustering*, a type of unsupervised learning.

Here we will learn from a set of data points $x_1, x_2, \ldots, x_n$, whose class labels are not known

12.1 Objective function (Likelihood)

Let there be a total of $|C|$ no. of classes. The likelihood function is written as follows:

$$\begin{aligned}
L(x_1,\ldots,x_n) &= \prod_{i=1}^{n} p(x_i) \\
&= \prod_{i=1}^{n}\sum_{j=1}^{|C|} p(x_i,c_j) \\
&= \prod_{i=1}^{n}\sum_{j=1}^{|C|} p(x_i|c_j)p(c_j) \\
LL(x_1,\ldots,x_n) &= \sum_{i=1}^{n}\log\big(\sum_{j=1}^{|C|} p(x_i|c_j)p(c_j)\big) \quad \text{(log-likelihood)}
\end{aligned}$$

In the above equation, we are tempted to move the log inside the summation. But to achieve this, we have to find a bound on the log-likelihood. We try to find the lower bound, since our original intention is to maximize LL. If we maximize the lower bound on LL we are better off.

Let rewrite the LL function in terms of some reference distribution $qneq0$ as follows:

$$LL = \sum_{i=1}^{n}\log\sum_{j=1}^{|C|} q(c_j|x_i)\left[\frac{p(x_i,c_j)}{q(c_j|x_i)}\right] \geq \sum_{i=1}^{n}\sum_{j=1}^{|C|} q(c_j|x_i)\log\left[\frac{p(x_i,c_j)}{q(c_j|x_i)}\right] \tag{16}$$

log is strictly concave function. Also

$$\log(\sigma_i\alpha_i x_i) = \sum_i \alpha_i \log x_i \qquad \iff x_i = x_j, \quad \forall i \neq j$$

So equality hold in Equation 16 iff,

$$\begin{aligned}
& \frac{p(x_i,c_j)}{q(c_j|x_i)} = \lambda(x_i) \quad \text{(for some } \lambda \text{ which is independent of } j\text{)} \\
\Rightarrow \quad & \frac{p(c_j|x_i)p(x_i)}{q(c_j|x_i)} = \lambda(x_i) \quad \forall j \\
\Rightarrow \quad & \frac{p(c_j|x_i)}{q(c_j|x_i)} = \lambda^{'}(x_i) \quad \text{(absorb } p(x_i) \text{ into } \lambda(x_i)\text{)} \\
\Rightarrow \quad & p(c_j|x_i) = \lambda^{'}(x_i)q(c_j|x_i) \quad \forall j \\
\Rightarrow \quad & \sum_{j=1}^{|C|} p(c_j|x_i) = 1 = \lambda^{'}(x_i)\sum_{j=1}^{|C|} q(c_j|x_i) = \lambda^{'}(x_i)
\end{aligned}$$

So equality hold iff,

$$\boxed{p(c_j|x_i) = q(c_j|x_i) \quad ,\forall j} \tag{17}$$

So the lower bound on log-likelihood is written as

$$\boxed{LL \geq \sum_{i=1}^{n}\sum_{j=1}^{|C|} q(c_j|x_i)\log\left[\frac{p(x_i,c_j)}{q(c_j|x_i)}\right] = \sum_{i=1}^{n}\sum_{j=1}^{|C|} q(c_j|x_i)\log p(x_i,c_j) - \sum_{i=1}^{n}\sum_{j=1}^{|C|} q(c_j|x_i)\log q(c_j|x_i)} \tag{18}$$

The first term looks like $\log p(x_i,c_j)^{q(c_j|x_i)}$. It is the expected log-likelihood under the q distribution and is written as $E_{q(c_j|x_i)}[LL]$. The second term is the entropy and is written as H_q.

Some observation on log-likelihood and its lower bound.

1. LL has params $q(c_j|x_i) = \theta_j = \{\mu_j, \Sigma_j\}$ and $p(c_j)$. LL is not convex in either of them.
2. Lower bound of LL has params $q(c_j|x_i)$, $p(c_j)$ and params of $p(x_i|c_j)$. It is not convex in any of these parameters.

So neither the lower bound of LL nor LL is convex. So can only find a local optimum. We next look at the procedure for finding the local optimum of this objective.

12.2 Expectation-Maximization Algorithm

We use procedures like co-ordinate descent to find the local optimum. This is also called *Expectation-Maximization (EM) algorithm.* It can be looked upon as batch co-ordinate ascent over the lower-bound of LL.

In EM algorithm, we divide the set of parameters to be estimated into two sets. They are $\big[q(c_j|x_i)\big]$ and $\big[\theta_j, p(c_j)\big]$. The constraint is that $\sum_j q(c_j|x_i) = 1, \forall i$.

Broadly speaking, we hold one set of parameters constant and optimize for the other set alternatively. The two steps are as follows:

1. **E-step (Expectation Step)**: Here we obtain the distribution q for which expectation is maximum.
$$\hat{q}(c_j|x_i) = \underset{q(c_j|x_i)}{\arg\max}\Big[E_q(LL) + H_q\Big] = p(c_j|x_i) \quad \text{(by Eq. 17)}$$
We keep $p(c_j)$ and θ_j fixed during this step.

2. **M-step (Maximization Step)**: Here we keep the $q(c_j|x_i)$ fixed and compute:
$$\hat{\theta}_j, \hat{p}(c_j) = \underset{\theta_j, p(c_j)}{\arg\max}\sum_{i=1}^{n}\sum_{j=1}^{|C|} q(c_j|x_i)\log p(x_i,c_j)$$

Application of EM to Mixture of Gaussians

This is also called the *probabilistic k-means clustering* algorithm. We have:

$$\begin{aligned} p(x_i) &= \sum_{j=1}^{|C|} p(x_i|c_j)p(c_j) \\ &= \sum_{j=1}^{|C|}\left[\frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}} e^{\frac{-(x_i-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j)}{2}}\right]p(c_j) \end{aligned}$$

The E and M steps are as follows:

1. E-step:

$$q(c_j|x_i) = \frac{p(x_j|c_j)p(c_j)}{\sum_j p(x_i|c_j)p(c_j)}$$

2. M-step:

$$\hat{p}(c_j), \hat{\mu}_j, \hat{\Sigma}_j = \underset{pc_j, \mu_j, \Sigma_j}{\arg\max} \left[\sum_{i=1}^{n} \sum_{j=1}^{|C|} q(c_j|x_i) \log \left(\frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}} e^{\frac{-(x_i-\mu_j)^T \Sigma_j^{-1} (x_i-\mu_j)}{2}} \right) \right]$$

Exercise

Prove the following parameter estimates obtained from the steps of EM algorithm for the mixture of gaussians.

1. $\hat{\mu}_j = \frac{\sum_{i=1}^{n} q(c_j|x_i) x_i}{n}$
2. $\hat{\Sigma}_j = \frac{\sum_{i=1}^{n} q(c_j|x_i)(x_i-\hat{\mu}_j)^T (x_i-\hat{\mu}_j)}{n}$
3. $\hat{p}(c_j) = \frac{1}{n} \sum_{i=1}^{n} q(c_j|x_i)$

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 18: Clustering: Similarity / Distance measures

Instructor: *Ganesh Ramakrishnan* — Date: *27/09/2011*
Computer Science & Engineering — Indian Institute of Technology, Bombay

EM for mixture of gaussians : Issues

1. It is not clear how to decide on the value of k (No. of clusters or mixture components)
2. Only locally optimal
3. No *hard* partitioning of points (will return a probability distribution for cluster membership)
4. Assumption about cluster shape (ellipsoid and convex)
5. Does not scale well with higher dimensions (primarily due to Σ_j's)
6. Compute intensive (If we have n points in m dimensions and we are finding k clusters, then each step of clustering takes $\mathcal{O}(nkm^2)$)
7. Not perfectly robust to outliers

13 Clustering

Now we will study various types of clustering algorithms like *k-means*, *k-mediods* and so on. These are special variants of mixture of Gaussians. They also come under the generative model framework. ($Pr(x|C)$).

The notes will follow the material from Jiawei Han's texbook, *Data Mining: Concepts and techniques, Chapter 7*. Read this in conjunction with the slides present at `http://www.cse.iitb.ac.in/~cs725/notes/classNotes/JaiweiHanDataMining.ppt`.

The broad steps in clustering algorithms are as follows:

1. Feature pre-processing on the datapoints
2. Distance/similarity measure formulation
3. Formulation of the objective function
4. Solving the objective function for optimality
   (thereby deriving the clusters that the datapoints belong to.)

13.1 Similarity Measures for Clustering

Consider a set of n feature vectors that describe various aspects about customers in a supermarket. Let the features be denoted by

$$\phi = \{\phi_1 \dots \phi_{m_1} \phi_{m_1+1} \dots \phi_{m_2} \phi_{m_2+1} \dots \phi_m\}$$

Let there be n such datapoints denoting n customers. There can be 3 different types of features

1. Numeric features: Features such as *age* and *salary* which can take a whole range of values.

2. Discrete features: Features such as *gender* = male, female and *marital status* = {married, single}

3. Ordinal features: Features where there is some notion of partial/total order among the values. For instance *designation* can take values ceo, manager, superintendent, engineer, worker or *car driven* can take values {luxury, sports, sedan, compact, mini} or *customer class* can be {gold, silver, bronze, none}

Feature preprocessing

Some kind of preprocessing is done on the features so that we dont have features which drastically vary in the values they take. Usually some kind of normalization is performed across all the values of a particular feature (column normalization). The other type of normalization across all the features of a particular datapoint (row normalization) is also useful sometimes, but is rare.

For numeric features, some normalizations are as follows:

1. $\dfrac{\phi_i - \phi_i^{min}}{\phi_i^{max} - \phi_i^{min}} \qquad \in [0,1]$
2. $\dfrac{\phi_i}{\phi_i^{max}}$
3. $\dfrac{\phi_i - \phi_i^{mean}}{\phi_i^{std}}$ (where ϕ_i^{mean} is the mean of values and $\phi_i^{std} = \dfrac{\sum_j |\phi_i(x_j) - \phi_i^{mean}|}{n}$)

For discrete features not much preprocessing is necessary. The ordinal features are mapped to numeric values and then preprocessing is applied on them.

A thing to note here is that feature processing is not unique to clustering and is widely used in other machine learning techniques. It significantly helps in classification as well.

Distance measures

The distance (or similarity) metric is denoted by d_{ij} (or s_{ij} respectively). It is the distance between any two datapoints i and j. Some examples of distance metrics are as follows:

1. Mahalanobis Distance: Given by the expression $||\phi(x) - \mu||_2^2 \Rightarrow (\phi(x) - \mu)^T \Sigma (\phi(x) - \mu)$. EM algorithm has this in some sense.

2. If $\phi(x)$ are numeric / ordinal, then a measure defined (after applying post-column normalization):

$$\Big(\sum_{l=1}^{m} \big(\phi_l(x_i) - \phi_l(x_j)\big)^p \Big)^{1/p}$$

   For different values of p, we get different distances:

   (a) $p = 1$: Manhattan distance
   (b) $p = 2$: Euclidean distance
   (c) $p > 2$: Minkowski distance

3. If $\phi(x)$ are binary, we define measures based on contingency matrix defined over any two features ϕ_i and ϕ_j.

$$M = \begin{bmatrix} \#(i=1, j=1) = p & \#(i=1, j=0) = q \\ \#(i=0, j=1) = r & \#(i=0, j=0) = s \end{bmatrix}$$

if $p+q+r+s=n$, some symmetric and asymmetric measures

(a) $d_{ij} = \frac{q+r}{n}$: symmetric
(b) $d_{ij} = \frac{q+r}{p+s}$: symmetric (odd's ratio)
(c) $d_{ij} = 1 - (p/n)$: asymmetric
(d) $d_{ij} = 1 - (s/n)$: asymmetric
(e) $d_{ij} = 1 - (\frac{p}{p+q+r})$: asymmetric
(Jaccard distance: refer: `http://en.wikipedia.org/wiki/Jaccard_index`)

4. If $\phi(x)$ are discrete then :

- $d_{ij} = 1 - \frac{\#(\phi_k(i)=\phi_k(j))}{n}$: Symmetric measure
- Expand ϕ to multiple binary features $\phi_1 \ldots \phi_k$, if the original ϕ, takes k values. Now we can have the various symmetric and asymmetric measures defined for binary features above.

5. If $\phi(x)$ is a combination of numeric/ordinal and discrete

$$tot\_d_{ij} = w_1 * d_{ij}^{discrete} + w_2 * d_{ij}^{num/ordinal} \quad \text{s.t. } w_1 + w_2 = 1, w_1, w_2 \in [0,1]$$

Similarity measures are usually metric measures. So $S_{ij} = 1 - d_{ij}$. Also sometimes they could be non-metric measures, for instance, cosine similarity given by:

$$\frac{\phi^T(x_i) \cdot \phi(x_j)}{||\phi(x_i)||^2 ||\phi(x_j)||^2}$$

Further Reading

1. *Jiawei Han, Micheline Kamber: Data Mining: Concepts and Techniques* , Chapter 7

2. *Hastie, Tibshirani, Friedman: The elements of Statistical Learning Springer Verlag*

3. *Ian H. Witten, Eibe Frank, Mark A. Hall: Data Mining: Practical Machine Learning Tools and Techniques*: for reference on data structures for efficient clustering / classification.

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 19: Clustering: Objective function, k-means algorithm and its variants

Instructor: *Ganesh Ramakrishnan* — Date: *30/09/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

13.2 Objective Function

In the case of mixture of Gaussians, the objective function to maximize was lower-bounded as follows:

$$LL \geq E_q(LL) + H_q$$

In the case of clustering using k-means algorithm, we can write the objective function as :

$$-\sum_{j=1}^{k} \sum_{x \in C_j} \left(\phi(x) - \mu_j\right)^2$$

It can be seen as a discrete optimization problem, given by:

$$O^* = max_{P_{ij},\mu_j}\left[-\sum_{j=1}^{k}\sum_{i=1}^{n} P_{ij}||\phi(x) - \mu_j||^2\right]$$
$$\text{s.t. } P_{ij} = 0 \text{ or } 1$$
$$\sum_{j=1}^{k} P_{ij} = 1$$

In this optimization program, the region specified by constraints is non-convex. So the entire program is not convex.

Suppose we have the objective function:

$$O^{'*} = max_{P_{ij},\mu_j}\left[-\sum_{j=1}^{k}\sum_{i=1}^{n} P_{ij}||\phi(x) - \mu_j||^2\right]$$
$$\text{s.t. } P_{ij} \in [0,1]$$
$$\sum_{j=1}^{k} P_{ij} = 1$$

We can show that O^* is a lower bound of $O^{'*}$ i.e $O^* \leq O^{'*}$. We can approximate the new objective ($O^{'*}$) function to the old (O^*) by

$$P^*_{ij} > \theta \rightarrow \hat{P}_{ij} = 1$$
$$P^*_{ij} \leq \theta \rightarrow \hat{P}_{ij} = 0$$

For some arbitrary threshold θ. (For e.g.: $\theta = 0.5$)

A brute force way of solvinf the problem is computationally infeasible. ($\mathcal{O}(k^n)$). One can use a branch and bound technique (early termination of some branches) to solve the problem. However, finding the bounds are hard.

13.3 k-means Algorithm

To solve O^* we can use an algorithm called k-means. It is a batch co-ordinate ascent algorithm on P_{ij}'s and μ_j's. At the beginning we have the assumption that the cluster centers are the average if the points belonging to the cluster. i.e

$$\mu_j = \frac{\sum_i P_{ij}\phi(x_i)}{\sum_i P_{ij}}$$

We will start with an arbitrary assignment of points to clusters. The steps of one iteration of the algorithm are:

Step 1:

$$\mu_j = \frac{\sum_i P_{ij}\phi(x_i)}{\sum_i P_{ij}}$$

Step 2:

$$P^*_{ij} \in \arg\max_{P_{ij}} \left[-\sum_{j=1}^{k}\sum_{i=1}^{n} P_{ij}\big(\phi(x_i) - \mu_j\big)^2 \right]$$

for each i

If

$$P_{ij} = 1 \text{ and } j \in \arg\max_{j'}\big[-\big(\phi(x_i) - \mu_j'\big)^2\big] \text{ then leave } P_{ij} \text{ unchanged for that } i$$

else

$$P_{ij} = \begin{cases} 1 & \text{if } j = \arg\max_{j'}\big[-\big(\phi(x_i) - \mu_j'\big)^2\big] \\ 0 & \text{otherwise} \end{cases}$$

Check after every iteration if any new P_{ij}' is different from old P_{ij}. If yes continue. If no halt.

Note: An iteration of k-means algorithm is of order $\mathcal{O}(nk)$. k-means algorithm is also called *k-centroids* algorithm.

Disadvantages of k-means

1. Fixed value of k: So determining the right value of k is very critical to the success of the approach.
2. Sometimes there is a problem owing to the wrong initialization of μ_j's.
3. Mean in "no-man's land": Lack of robustness to outliers.

13.4 Variants of k-means

Some of the variants of k-means are discussed in this section. For more details read Chapter 7 of Jiawei Han's book. The variants are *k-mediods*, *k-modes*, *CLARA* and *CLARANS*.

k-mediods

Here the assumption is $\mu_j = \phi(x_i)$ for some value of i. In otherwords, the cluster's centroid coincides with one of the actual points in the data. Each step of the k-mediod algorithm is $k(n-1)n \sim \mathcal{O}(kn^2)$

k-modes

For discrete valued attributes, in the k-modes algorithm we have:

$$\left[\mu_j\right] = \underset{v \in \{1, \ldots V_l\}}{\arg\max} \sum_{x_i \in C_j} \delta\big(\phi_l(x_i, v)\big) \quad \forall l = 1 \ldots m$$

For continuous valued attributes, we may need to fit some distribution and find the mode for each feature over the cluster.

CLARA and CLARANS

In CLARA the μ's should be $\phi(x_i)$ only for $x_i \in S$ where S is a fixed sample i.e. $\mu_{ij} = \phi(x_i)$ for some $x_i \in S$. Each step of this algorithm is $k * |S| * (n-1) \sim \mathcal{O}(k|S|(n-1))$ where $|S| > k$.

CLARANS is same in all respects as CLARA except that S is different in each iteration.

Exercise

1. Write the steps of the k-mediod algorithm.

2. Given the following objective function with the constraints:

$$max_{P_{ij}, \mu_j} \left[-\sum_{j=1}^{k} \sum_{i=1}^{n} P_{ij} ||\phi(x) - \mu_j||^2 \right]$$
$$\text{s.t. } P_{ij} \in [0, 1]$$
$$\sum_{j=1}^{k} P_{ij} = 1$$

Check if this is a convex optimization program and solve.

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 20: Clustering and MDL, Hierachical Clustering

Instructor: *Ganesh Ramakrishnan* — Date: *07/10/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

13.5 MDL principle and clustering

The principle of *minimum description length (MDL)* is a general one and not necessarily specific to clustering. We can see the clustering objective as one that tries to achieve the MDL objective. The explanation is as follows:

Consider a small change in the k-means clustering objective. Now the objective also optimizes over the number of clusters k. So we have,

$$\operatorname*{arg\,max}_{P_{ij},\mu_j,k} - \sum_{j=1}^{k}\sum_{i=1}^{n} P_{ij}||\phi(x_i) - \mu_j||^2$$

However, letting the algorithm choose the value of k will lead to a case where $k = n$ where each point is a cluster of its own. This kind of trivial clustering is definitely not desirable.

Now let us consider the problem of communicating the location of points to another person over a SMS. The more information we pass the more is the cost. So we want to minimize the amout of information set yet be precise enough so that the other side reconstruct the same set of points given the information.

In the extreme case we send actual co-ordinates of the points. This is a very costly information transfer but allows the other person to accurately reconstruct the points.

Another option would be to communicate cluster information. So in this we send across the cluster centroids, the difference of points with its corresponding centroid and the magnitude of the difference. So in this case, the amount of information to be sent across is much less given that the magnitude of the points may be large but their difference will be small. Hence. information to be encoded is less. This is akin to the MDL principle.

Let us denote the data by D and theory about the data T (cluster information and so on). Let the information that is to be encoded be denoted by I bits. So we can approximate $I(D)$ as $I(D, D)$ and is given by the expression:

$$I(D) \approx I(D, D) > I(D|T) + I(T)$$

where $I(D)$ represents the co-ordinate of every point. $I(D|T)$ is cluster information of a point like membership of point to a cluster and relative co-ordinate (difference) of point w.r.t to the corresponding cluster. $I(T)$ is overall clustering information like number of clusters, order of magnitude of points in each cluster and means of every cluster.

$P(D, T)$ is similar in spirit to the Bayes rule. We have $P(D, T) = P(D|T)P(T)$ or
$\log P(D, T) = \log P(D|T) + \log P(T)$

So we can modify the original objective as follows:

$$\operatorname*{arg\,max}_{P_{ij},\mu_j,k} - \sum_{j=1}^{k}\sum_{i=1}^{n} P_{ij}||\phi(x_i) - \mu_j||^2 - \gamma k$$

The first term of the objective is I(D—T) and the second term is -I(T). The second term is also called the *regularizer*.

In essence, according to the MDL principle, we need to define $I(D|T)$ and $I(T)$ and choose T such that it minimizes $I(D|T) + I(T)$. This is also aligned with the *Occam Razor* principle.

13.6 Hierachical clustering

The choices to be made during this type of clustering are as follows:

1. Bottom-up (agglomerative) vs. Top-down (divisive).
2. Which two clusters to merge (single-link, complete-link, average distance): Merge clusters that have the least mutual distance. Altenatively, which clusters to break.
3. When to stop merging clusters (closely linked to the distance measure). Stop when the distance between two clusters is $> \theta$ (some threshold). Altenatively, when to stop splitting the clusters.

Some of the issues are :

1. Can't undo clustering decision.
2. Lack of flexibility in choice of clustering algorithm
3. Do not scale well.

However, the main advantage of hierarchical clustering is that it is easy to visualize. So a choosen k from hierarchical clustering can be used in k-means or any other clustering algorithm run from scratch.

Some of the algoritms studied here were *Birch clustering* and *Chameleon.*

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 21: Clustering: Miscellaneous topics, Apriori Algorithm

Instructor: *Ganesh Ramakrishnan* — Date: *08/10/2011*
Computer Science & Engineering — Indian Institute of Technology, Bombay

13.7 Miscellaneous topics

Observations / Issues with Density-based clustering (DBScan)

1. Distance between any 2 self-elected centers is $> \epsilon$.
2. Each cluster is an ϵ ball.

To address Issue 1 we could do the following:

1. A point x is a center if $ball(\epsilon, x)$ contains at least minimum set of points m.
2. Centres may give up individual cluster and merge with other cluster. (x is only a potential center)

Grid-based clustering

An algorithm called STING was discussed in class. Some of the salient steps of the algorithm (see Han's book for more details)

1. Partition the space recursively according to different levels of resolution.
2. Compute $(n, \sum X_i, \sum X_i^2, distrib\_type, min, max)$ for each lowest level cell directly from data.
3. Compute $(n, \sum X_i, \sum X_i^2, distrib\_type, min, max)$ for each higher cell successively from the statistics of its lower level cells (i.e. children)
4. Scan every point. For each point, compute significance tests in all leaves. Find the best fit and move up.

To find the type of distribution in step 2, we use the concept of goodness-of-fit. Step 3 of the algorithm is of order $\mathcal{O}(grid\_hierarchy\_size)$.

Exercise

1. Construct a case where DBScan will discover non-convex clusters.
2. Construct a case where DBScan will discover one cluster inside another.

Further Reading

Read more about goodness-of-fit, MDL, Hypothesis Testing and cross-validation

14 Association Rule Mining

All algorithms so far cluster points. Now we try to cluster features. Get all feature sets that have count (*support*) $> \mathcal{S}$. These are called *interesting features*. A brute force way of doing this is of order $\mathcal{O}(2^m)$ where m is the number of features.

14.1 A-priori Algorithm

This is a smarter way of achieving the same objective. It is in the following spirit: If $count(\phi_1) > \mathcal{S}$ and $count(\phi_2) < \mathcal{S}$ (not interesting), then obviously $count(\phi_1, \phi_2) < \mathcal{S}$ (not interesting). So we will refrain from looking at such feature pairs.

In general, we construct the following sets

$$S_1 = \big\{\phi_1 | count(\phi_1) > \mathcal{S}\big\}$$
$$S_2 = \big\{\{\phi_i, \phi_j\} | \phi_i, \phi_j \in S_1, i \neq j, count(\phi_i, \phi_j) > \mathcal{S}\big\}$$

This is count is calculated using implementations of efficient data-structures called inverted index, precomputed from data. Forward index contains a mapping from examples to features i.e $e_i \rightarrow \{\phi_j\}$ where as the inverted index contains the reverse mapping from features to examples that contain them i.e. $\phi_j \rightarrow \{e_i\}$.

In calculating S_3, we have two options :

$$\text{Option 1: } S_3 = \big\{S_1^i \cup S_2^j \big| S_1^i \in S_1, S_2^j \in S_2, count(S_1^i \cup S_2^j) > \mathcal{S}\big\}$$
$$\text{Option 2: } S_3 = \big\{S_2^i \cup S_2^j \big| |S_2^i \cap S_2^j| = 1, S_2^i \in S_2, S_2^j \in S_2, count(S_2^i \cup S_2^j) > \mathcal{S}\big\}$$

When we generalize Option 2, we get the following general expression:

$$S_n = \big\{S_{n-1}^i \cup S_{n-1}^j \big| |S_{n-1}^i \cap S_{n-1}^j| = n-1, S_{n-1}^i \in S_{n-1}, S_{n-1}^j \in S_{n-1}, count(S_{n-1}^i \cup S_{n-1}^j) > \mathcal{S}\big\}$$

The final set of all interesting sets $\boxed{S^* = \bigcup_{i=1}^{m} S_i}$

Excercise

Write the generalized expression for Option 1 above.

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 22: Association Rule Mining, Non-parametric density estimation

Instructor: *Ganesh Ramakrishnan* — Date: *11/10/2011*
Computer Science & Engineering — Indian Institute of Technology, Bombay

14.2 Association Rule Mining

[3] In data mining, association rule learning is a popular and well researched method for discovering interesting relations between variables in large databases [4]. Association rules have been introduced for discovering regularities between products in large scale transaction data recorded by point-of-sale (POS) systems in supermarkets.
Simple Association Rule: The problem of Association Rule can be defined as follows:
Given I=$\{i_1, i_2, \ldots, i_n\}$ be a set of n binary attributes called items. and $D = \{\phi_1, \phi_2, \ldots, \phi_m\}$ be a set of transactions called the database, where each phi_i contains one or many of the attributes in I. Now the goal is to find an association rule R:

$$\phi_{i_1} \wedge \phi_{i_2} \wedge \ldots \wedge \phi_{i_k} \Rightarrow \phi_{i_j}$$

such that $Sup(R) > s$ i.e., Support$(\phi_{i_1} \wedge \phi_{i_2} \wedge \ldots \wedge \phi_{i_k} \Rightarrow \phi_{i_j}) > s$ and
$Conf(R) > c$ where

$$Conf(\phi_{i_1} \wedge \phi_{i_2} \wedge \ldots \wedge \phi_{i_k} \Rightarrow \phi_{i_j}) = \frac{Sup(\phi_{i_1}, \phi_{i_2}, \ldots .\phi_{i_k}, \phi_{i_j})}{Sup(\phi_{i_1}, \phi_{i_2}, \ldots .\phi_{i_k})} \tag{19}$$

which can be otherwise stated as:

$$Conf(body \Rightarrow head) = \frac{Sup(body, head)}{Sup(body)} \tag{20}$$

Types of Association Rule Mining

Association Rule Mining also called Item Set Mining, has following subcategories as well.

1. **Constraint based mining**: In this type of rule mining, constraints can be placed on what should be discovered and what should be not.

2. **Handling Quantitaive attributes**:

3. **Multi-level Association Rule**: This type of association rule mining gets rules by placing higher threshold for smaller sets ensuring concrete rules and then in the next level decrease threshold to get ruels for next level and so on.

[3] **Lecture scribed by Kedhar**
[4] `http://en.wikipedia.org/wiki/Association_rule_learning`

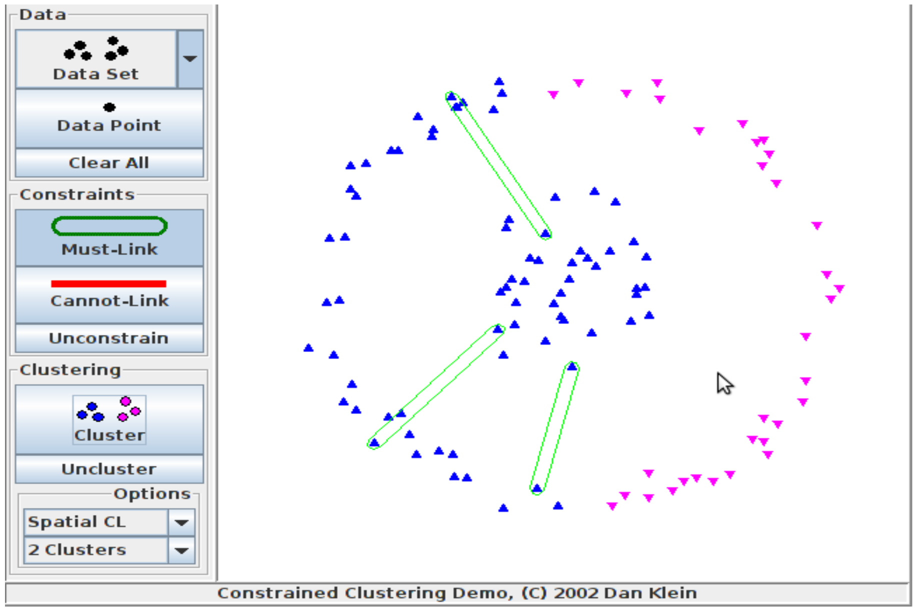


Figure 21: Figure showing constrained clustering

14.3 Classification Methods

Classification methods can be classified into three categories depending upon human intervention. They are:

- **Supervised methods**: These methods require human supervision. They are classified further as:
  - Non Metric Classifier learning
  - Probabilistic and generative approaches like Naive Bayes and Gradient Descent etc.
- **Unupervised methods**: These type of methods do not require any sort of human intervention. Some examples are:
  - Clustering
  - Association rule mining
- **Semi-supervised**: These type of methods require some sort of human intervention in some stages. Some examples are:
  - Active Learning: In this approach, there is some input from the user on the instance to be used to learn some rules or do some classification.
  - Active Labelling: User will supervise the system on what features should be used for the problem.
  - Several other graphical model learning techniques

15 Non parametric density estimation and classification

Consider $D = \{\phi(x), C(x)\}$ be the set of input, where $C(x)$ is the class to which x belong to. Now given the instance x, the problem is to classify it into one of the classes. Using the Bayes theorem, we have:

$$\Pr(C/x) = \frac{\Pr(x/C)*\Pr(C)}{\Pr(x)}$$

15.1 Histogram Method

: Let $x_1, x_2, \ldots, x_n$ be 1-dimensional data points. To classify these n points, in this method, construct 't' bins, each of same width. Populate the t bins with these instances (as in histogram completion).
Let $x \in B_i$, then

$$\Pr(x) = \frac{|B_i|}{\sum_{j=1}^{n} |B_j|} \tag{21}$$

Using the histogram computaion, we can do histogram seperately for everey class to compute $\Pr(x/C)$. Extending the concept to instances with m-dimensions, the number of bins would be t^m, which increases exponentially with the dimensions of the instances. Hence, this method do not seem to be a good one.

If $\Pr(x)$ ~true density ($\Pr(x)$ can be replaced by $\Pr(x/C)$).
Now consider area A given by P_A.

$$P_A = P(x \in A) = \int \Pr(x)dx = P_x * |A| \tag{22}$$

if A is small. Now considering k points from T, we have
$P(k$ points from $T \in$ A)= $\binom{n}{r} * (P_A)^k * (1 - P_A)^{n-k}$ (Binomial distribution)
Then the expectation,
E(number of points that lie in A) $= E(k) = nP_A$.
Its variance is given by $var(k) = n * P_A * (1 - P_A)$.
These deriviations are key fot K-Nearest Neighbours and Parzen Window Classifier.

15.2 K-Nearest Neighbour and Parzen Window Classifier

In the section 15.1, if $k \approx n * P_A$ and $P_A \approx P_x * |A|$, we have

$$P_x \approx \frac{k}{n * |A|} \tag{23}$$

In 23, We fix the value of k and determine A in which these k lie. This is K-Nearest Neighbour classifier. Otherwise, fix the region A and then determine k based on the training sample, this is Parzen window kernel density classifier.

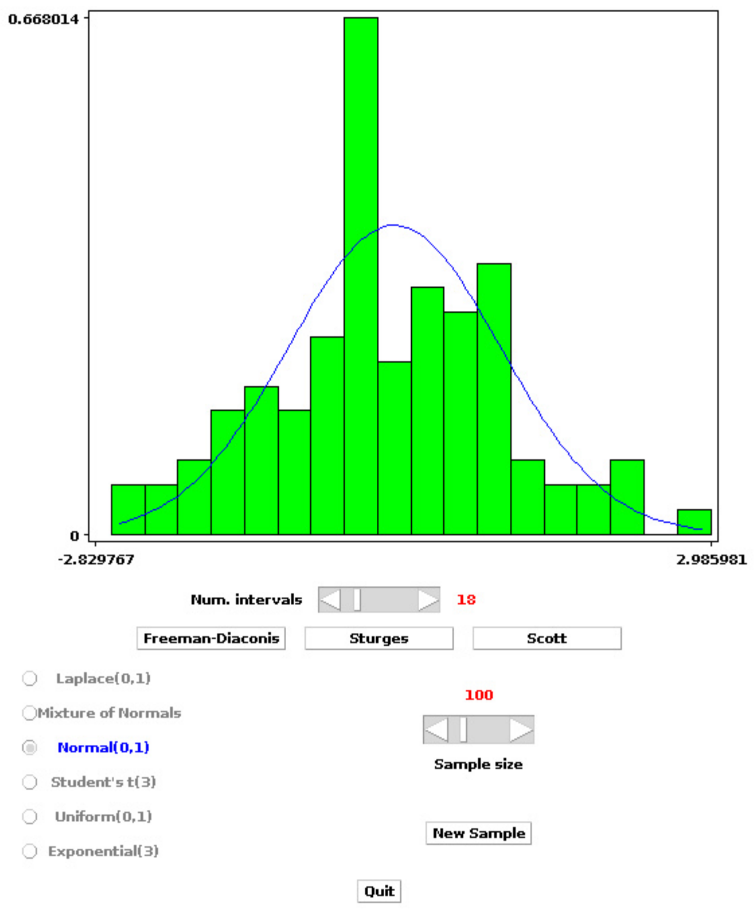


Figure 22: Figure showing clustering using histogram method

K-Nearest Neighbour classification

In this method, the goal is to find out the K-nearest neighbours to the given instance x^*.
Now, $\Pr(x) = \frac{k}{n*|A|}$.
Let $x_1^1, x_2, \ldots .x_k$ be the K-nearest neighbours and let k_j be the number of points from K-NN that belong to class C_j.

$$\Pr(x/C_j) = \frac{k_j}{n*|A|} \tag{24}$$

and

$$\Pr(C_j/x) = \frac{\frac{k_j}{n*|A|} * \Pr(C_j)}{\frac{k}{n*|A|}} = \frac{k_j}{k} * \Pr(C_j) \tag{25}$$

From the above equation, it can be understood that Standard KNN takes uniform $\Pr(C-j)$.

Now the class(C_*) to which x_* belong to can obtained as:

$$C_* = argmax_{C_j} \frac{k_j}{k} * \Pr(C_j) \tag{26}$$

Notes:K-Nearest Neighbour classifier is also called memory based classifier. Its a lazy classifier and non-smooth one.

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 23: Non-parametric density estimation, Probabilistic discriminative classification models

Instructor: *Ganesh Ramakrishnan* — Date: *14/10/2011*
Computer Science & Engineering — Indian Institute of Technology, Bombay

15.3 Kernel density estimators

Let us consider a hypercube of dimension h. Assume that the volume is fixed (alternately h or σ (in case of smoother version - see below)). $p_h(x)$ is given by:

$$\begin{aligned} p_h(x) &= \frac{K_h(x)}{nV} = \frac{K_h(x)}{nh^m} \\ &= \sum_{i=1}^{n} \frac{K(x, x_i}{nh^m} \end{aligned}$$

In the case of *Rectangular kernel*, K is given by:

$$K(x, x_i) = \begin{cases} 1 & \text{iff } |x - x_i| \leq [h/2 \; h/2 \ldots h/2]^T_{m_{dim}} \\ 0 & \text{otherwise} \end{cases}$$

In the smoother version called the *Gaussian kernel*, we have $p_\sigma(x)$ as

$$p_\sigma(x) = \frac{1}{n} \sum_{i=1}^{n} K_\sigma(x, x_i)$$

where K_σ is given by:

$$K_\sigma = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{||x-x_i||^2}{2\sigma^2}}$$

Generally, we can absorb h^m in the K function and we have,

$$\begin{aligned} p(x) &= \frac{1}{n} \sum_{i=1}^{n} K(x, x_i) \\ p(x|c_j) &= \frac{1}{n_j} \sum_{x_i \in c_j} K(x, x_i) \\ p(c_j|x) &= \frac{p(x|c_j)p(c_j)}{p(x)} \quad \text{(by Bayes's rule)} \\ \arg\max_j p(c_j|x) &= \arg\max_j p(c_j) \times (1/n_j) \sum_{x_i \in c_j} K(x, x_i) \\ &= \arg\max_j \frac{1}{n} \sum_{x_i \in c_j} K(x, x_i) \quad (\text{since } p(c_j) = n_j/n) \end{aligned}$$

The equation above is of the *parzen window classifier.*

Unfortunately, this requires the kernel to be evaluated on too many points, which translates to high memory and computation requirements.

Note: Kernel density estimation on a sparse set of points (instead of all points like before; can we do kernel density estimation on a "good" set of points ?). This is one way of looking at *Support Vector Machines*(to be seen later). This can also be seen as a weighted kernel density estimation

$$p(c_j|x) \propto \sum_{i=1}^{n} \alpha_i K(x, x_i)$$

where α_i's are the parameters to be discovered and are in some range. Also most α_i's are 0. This gives the sparsity that we discussed before.

16 Discriminative Classification

Most models seen so far try to model the likelihood of the data i.e $p(x|c_j)$. So we indirectly obtain $p(c_j|x)$ using the Bayes' rule:

$$p(c_j|x) = \frac{p(x|c_j)p(c_j)}{p(x)}$$

What if we model $p(c_j|x)$ directly ? This is called *probabilistic discriminative classification model.*

16.1 Maximum Entropy models

Given no other information, if we have to model $p(c_j|x)$, we have to consider a uniform distribution. When there is a uniform distribution, the entropy has the maximum value. Therefore, in this paradigm, we try to maximize the entropy. Hence the name Maximum entropy models.

$$max_{p(c_j|x_i)} - \sum_{i=1}^{n} \sum_{j=1}^{|c|} p(c_j|x_i) \log p(c_j|x_i)$$

$$\text{s.t. } (1) \sum_{i=1}^{n} \sum_{j=1}^{|c|} p(x_i, c_j)\phi_l(x_i, c_j) = \frac{1}{n|c|} \sum_{i=1}^{n} \phi_l(x, c(x_i)) \quad , \forall l$$

i.e. $\mathcal{E}_{p(x_i,c_j)}(\phi_l) = Avg(\phi_l)$ (expected entropy w.r.t distribution = actual entropy seen in data)

$$(2) \sum_{j} p(c_j|x_i) = 1, \forall i \quad \text{and} \quad p(c_j|x_i) \in [0, 1]$$

The above formaluation is a *concave optimization program.* Optimality occurs at (see practice h/w 2, problem 3):

$$-1 - \log p(c_j|x_i) + \sum_{l=1}^{m} \lambda_l \phi_l(x_i, c_j) + \eta_i + \theta_{ij}^{'} - \theta_{ij}^{2} = 0$$

$$\Rightarrow p(c_j|x_i) = \frac{e^{\sum_{i=1}^{m} \lambda_l \phi_l(x_i, c_j)}}{\sum_{j'=1}^{|c|} e^{\sum_{i=1}^{m} \lambda_l \phi_l(x_i, c_{j'})}}$$

Where λ_l parameters are obtained from the 1st constraint and η and θ from the 2nd constraint. We can show that the equivalent dual problem (using $p(c_j|x_i) \propto e^{\sum_{i=1}^{m} \lambda_l \phi_l(x_i,c_j)}$) as:

$$max_{\lambda_l} LL = max_{\lambda_l} \sum_{i=1}^{n} \log p(c(x_i)|x_i) \tag{27}$$

This form of $p(c_j|x_i)$ is an important element of the exponential family of distributions. (See further reading below for a detailed description of exponential family)

e.g.: Logistic Regression

Assume $\{\phi_{l,c_j}(x,c)\}$ is a set of features indexed by pair (l, c_j), $\forall c_j$ and $\forall l$. The corresponding parameters are λ_{l,c_j}. Also assume that:

$$\phi_{l,c_j}(x,c) = \begin{cases} \phi_l(x) & \text{iff } c = c_j \\ 0 & \text{otherwise} \end{cases}$$

Now $p(c_j|x)$ is given by:

$$p(c_j|x) \propto e^{\sum_{l,c_k} \lambda_{l,c_k} \phi_{l,c_k}(x,c_j)} = e^{\sum_l \lambda_{l,c_j} \phi_l(x)}$$

And the classifier C^* is given by:

$$C^* = \arg\max_{c_j} \sum_{l,c_j} e^{\lambda_{l,c_j} \phi_l(x)}$$

this is also called the *logistic regression classifier*. The logistic regression classifier's decision surface is linear in the ϕ space.

Note: How to learn λ ?

1. Gradient descent: Updates are given by (for the dual problem in Eq. 27):

$$\lambda^{new} = \lambda^{old} - \alpha \nabla LL$$

. Where the jth component of ∇LL is given by:

$$\sum_{i=1}^{n} \phi_l(x_i, c(x_i)) - \sum_{i=1}^{n} \sum_{j=1}^{|c|} p^{old}(c_j|x_i)\phi_l(x_i, c_j)$$

Here, $p^{old}(c_j|x_i)\phi_l(x_i, c_j)$ is computed using λ^{old}

2. Newton method: Updates are given by

$$\lambda^{new} = \lambda^{old} - (\nabla^2 LL)^{-1} \nabla LL$$

. $\nabla^2 LL$ is given by: $\nabla^2 LL = \phi^T M \phi$

Further Reading

Exponential family : Read Section 7.4 of `http://www.cse.iitb.ac.in/~cs725/notes/classNotes/misc/CaseStudyWithProbabilisticModels.pdf`

Exercise

Write the expression for M in the Newton updates expression of $\nabla^2 LL$.
See *Convex optimization notes section 4.5.2.*(`http://www.cse.iitb.ac.in/~cs725/notes/classNotes/BasicsOfConvexOptimization.pdf`)

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 24: Graphical models primer

Instructor: *Ganesh Ramakrishnan* — Date: *18/10/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

17 Graphical Models

In this lecture a general introduction to *graphical models* was presented. The two broad categories of graphical models namely undirected and directed models were discussed. Properties of conditional independence and how a graph is factored based on this was also discussed. It was stressed upon that the *absence of an edge is more important than the presence of an edge*.

Some ways of inferencing in graphical models were briefly touched upon.

Further Reading

1. Graphical Models slides presented in class : `http://www.cse.iitb.ac.in/~cs725/notes/classNotes/graphicalModels.ppt`
2. Detailed graphical models notes : `http://www.cse.iitb.ac.in/~cs725/notes/classNotes/misc/CaseStudyWithProbabilisticModels.pdf`

CS 725 : Foundations of Machine Learning Autumn 2011

Lecture 25: Linear Models, Regression, Least Squared Error

Instructor: *Ganesh Ramakrishnan* Date: *21/10/2011*
Computer Science & Engineering Indian Institute of Technology, Bombay

18 Generalized Models for Classification

The goal in classification is to take an input vector $\mathbf{x}$ and assign it to one of K discrete classes $\mathcal{C}_k$ where $k = 1, ..., K$. In most cases, the classes are taken to be disjoint, so that each input is assigned to one and only one class. The input space is thus divided into *decision regions* whose boundaries are called *decision boundaries* or *decision surfaces.* We consider only linear models for classification in this lecture, which means that the decision surfaces are linear functions of the input vector $\mathbf{x}$ and hence are defined by $(D-1)$-dimensional hyperplanes within the D-dimensional input space.

The simplest method of classification (for 2 classes) is to design a function f such that

$$f(\mathbf{x}_i) = \begin{cases} v_{c_+} & \text{if } \mathbf{x}_i \in \mathcal{C}_+ \\ v_{c_-} & \text{if } \mathbf{x}_i \in \mathcal{C}_- \end{cases}$$

Three broad types of classifiers

1. The first method involves explicit construction of $\mathbf{w}$ for $\mathbf{w}^T\phi(\mathbf{x}) = 0$ as the decision surface.
2. The second method is to model $P(\mathbf{x}|\mathcal{C}_+)$ and $P(\mathbf{x}|\mathcal{C}_-)$ together with the prior probabilities $P(\mathcal{C}_k)$ for the classes, from which we can compute the posterior probabilities using Bayes' theorem
$$P(\mathcal{C}_k|\mathbf{x}) = \frac{P(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{P(\mathbf{x})}$$
These types of models are called *generative models.*
3. The third method is to model $P(\mathcal{C}_+|\mathbf{x})$ and $P(\mathcal{C}_-|\mathbf{x})$ directly. These types of models are called *discriminative models.* In this case $P(\mathcal{C}_+|\mathbf{x}) = P(\mathcal{C}_-|\mathbf{x})$ gives the required decision boundary.

18.1 Generalized linear models

In these models we adopt linear regression to model the classification problem. This is done by modeling a function f as follows:
$$f(\mathbf{x}) = g(\mathbf{w}^T\phi(\mathbf{x}))$$
where g is known as *activation function* and ϕ the vector of basis functions. Classification is achieved by:
$$g(\theta) = \begin{cases} v_{c_+} & \text{if } \theta > 0 \\ v_{c_-} & \text{if } \theta < 0 \end{cases}$$
The decision surface in this case is given by $\mathbf{w}^T\phi(\mathbf{x}) = 0$

Examples

An example of generative model is as follows:

$$P(\mathbf{x}|\mathcal{C}_+) = \mathcal{N}(\mu_+, \Sigma)$$

$$P(\mathbf{x}|\mathcal{C}_-) = \mathcal{N}(\mu_-, \Sigma)$$

With prior probabilities $P(\mathcal{C}_+)$ and $P(\mathcal{C}_-)$ known, we can derive $P(\mathcal{C}_+|\mathbf{x})$ and $P(\mathcal{C}_+|\mathbf{x})$. In this case it can be shown that the decision boundary $P(\mathcal{C}_+|\mathbf{x}) = P(\mathcal{C}_-|\mathbf{x})$ is a hyperplane.

An example of discriminative model is

$$P(\mathcal{C}_+|\mathbf{x}) = \frac{e^{\mathbf{w}^T\phi(\mathbf{x})}}{1 + e^{\mathbf{w}^T\phi(\mathbf{x})}}$$

$$P(\mathcal{C}_-|\mathbf{x}) = \frac{1}{1 + e^{\mathbf{w}^T\phi(\mathbf{x})}}$$

Examples of first model (which directly construct the classifier) include

- Linear Regression
- Perceptron
- Fisher's Discriminant
- Support Vector Machines

18.2 Handling Multiclasses

We now consider the extension of linear discriminants to $K > 2$ classes. One solution is to buid a K-class discriminant by combining a number of two-class discriminant functions.

- *one-versus-the-rest*: In this approach, $K-1$ classifiers are constructed, each of which separtes the points in a particular class $\mathcal{C}_k$ from points not in that classes

- *one-versus-one*: In this method, ${}^KC_2$ binary discriminant functions are introduced, one for every possible pair of classes.

Attempting to construct a K class discriminant from a set of two class discriminants leads to ambiguous regions. The problems with the first two approaches are illustrated in Figures 23 and 24, where there are ambiguous regions marked with '?'.

Avoiding ambiguities

We can avoid above mentioned difficulties by considering a single K-class discriminant comprising K functions $g_{\mathcal{C}_k}(\mathbf{x})$. Then $\mathbf{x}$ is assigned to a class $\mathcal{C}_k$ that has the maximum value for $g_{\mathcal{C}_k}(\mathbf{x})$

If $g_{\mathcal{C}_k}(\mathbf{x}) = \mathbf{w}_{\mathcal{C}_k}^T\phi(\mathbf{x})$ the decision boundary between class $\mathcal{C}_j$ and class $\mathcal{C}_k$ is given by $g_{\mathcal{C}_k}(\mathbf{x}) = g_{\mathcal{C}_j}(\mathbf{x})$ and hence corresponds to

$$(\mathbf{w}_{\mathcal{C}_k}^T - \mathbf{w}_{\mathcal{C}_j}^T)\phi(\mathbf{x}) = 0$$

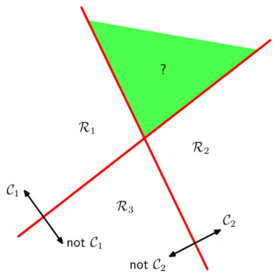


Figure 23: Illustrates the ambiguity in *one-versus-rest* case

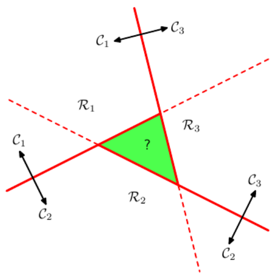


Figure 24: Illustrates the ambiguity in *one-versus-one* case

18.3 Least Squares approach for classification

We now apply the Least squares method to the classification problem. Consider a classification problem with K classes. Then the target values are represented by a K component target vector $\mathbf{t}$. Each class is described by its own model

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \phi(\mathbf{x})$$

where $k \in \{1, ...K\}$. We can conveniently group these together using vector notation so that

$$\mathbf{y}(\mathbf{x}) = \mathbf{W}^T \phi(\mathbf{x})$$

where $\mathbf{W}$ is a matrix whose k^{th} column comprises the unknown parameters $\mathbf{w}_k$ and $\phi(\mathbf{x})$ is the vector of basis function values evaluated at the input vector $\mathbf{x}$. The procedure for classification is then to assign a new input vector $\mathbf{x}$ to the class for which the output $y_k = \mathbf{w}_k^T \phi(\mathbf{x})$ is largest.

We now determine the parameter matrix $\mathbf{W}$ by minimizing a sum-of-squares error function. Consider a training data set $\{\mathbf{x}_n, \mathbf{t}_n\}$ where $n \in \{1, .., N\}$, where $\mathbf{x}_n$ is input and $\mathbf{t}_n$ is corresponding target vector. We now define a matrix $\mathbf{\Phi}$ whose n^{th} row is given by $\phi(\mathbf{x}_n)$.

$$\mathbf{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \dots & \phi_{K-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \dots & \phi_{K-1}(\mathbf{x}_2) \\ \dots & \dots & \dots & \dots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \dots & \phi_{K-1}(\mathbf{x}_N) \end{pmatrix}$$

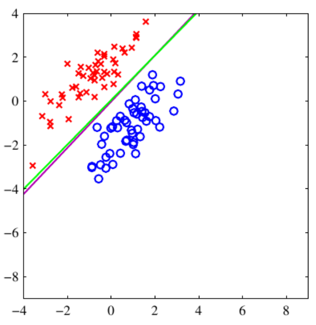


Figure 25: Data from two classes classified by least squares (magenta) and logistic (green)

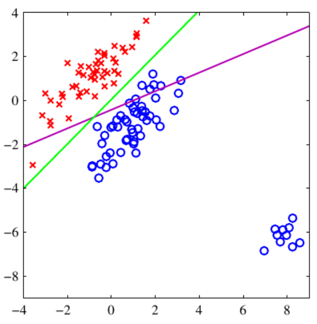


Figure 26: Response of the classifiers to addition of outlier points

We further define a matrix $\mathbf{T}$ whose n^{th} row is given by the vector $\mathbf{t}_n^T$. Now, the sum-of-squares error function can then be written as

$$err(\mathbf{W}) = \frac{1}{2} Tr\{(\mathbf{\Phi W} - \mathbf{T})^T (\mathbf{\Phi W} - \mathbf{T})\}$$

We can now minimize the error by setting the derivative with respect to $\mathbf{W}$ to zero. The solution we obtain for $\mathbf{W}$ is then of the form

$$\mathbf{W} = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{T}$$

Limitations of Least Squares

Even as the least-squares approach gives a closed form solution for the discriminant function parameters, it suffers from problems such as lack of robustness to outliers. This is illustrated in Figures 25 and 26 where we see that introduction of additional data points in the Figure 26 produce a significant change in the location of the decision boundary, even though these points would be correctly classified by the original boundary in Figure 25. For comparison, least squares approach is contrasted with logisitc regression, which remains unaffected due to the additional points.

19 Regression

Suppose there are two sets of variables $\mathbf{x} \in \Re^n$ and $\mathbf{y} \in \Re^k$ such that $\mathbf{x}$ is independent and y is dependant. The regression problem is concerned with determining y in terms of $\mathbf{x}$. Let us assume that we are given m data points $\mathcal{D} = \langle \mathbf{x}_1, \mathbf{y}_1 \rangle, \langle \mathbf{x}_2, \mathbf{y}_2 \rangle, .., \langle \mathbf{x}_m, \mathbf{y}_m \rangle$. Then the problem is to determine a function f^* such that $f^*(\mathbf{x})$ is the best predictor for $\mathbf{y}$, with respect to $\mathcal{D}$. Suppose $\varepsilon(f, \mathcal{D})$ is an error function, designed to reflect the discrepancy between the predicted value $f(\mathbf{x}')$ of $\mathbf{y}'$ and the actual value $\mathbf{y}'$ for any $\langle \mathbf{x}', \mathbf{y}' \rangle \in \mathcal{D}$, then

$$f^* = \operatorname*{arg\,min}_{f \in \mathcal{F}} \varepsilon(f, \mathcal{D}) \tag{28}$$

where, $\mathcal{F}$ denotes the class of functions over which the optimization is performed.

19.1 Motivating Example : Curve Fitting

Learn $f : X \rightarrow Y$ such that $E(f, X, Y_1)$ is minimized. Here the error function E and form of the function to learn f is chosen by the modeler.

Consider one such form of f,

$$f(x) = w_0 + w_1 x + w_2 x^2 + ... + w_t x^t$$

The sum of squares error is given by,

$$E = \frac{1}{2} \sum_{i=1}^{m} (f(x_i) - y_i)^2$$

So the expression is,

$$\operatorname*{arg\,min}_{w=[w_1,w_2,...w_t]} \frac{1}{2} \sum_{i=1}^{K} [(w_0 + w_1 x + w_2 x^2 + ... + w_t x^t) - y_1(i)]^2$$

If there are m data points, then a polynomial of degree $m-1$ can exactly fit the data, since the polynomial has m degrees of freedom (where degrees of freedom=no. of coefficients)

As the degree of the polynomial increases beyond m, the curve becomes more and more wobbly, while still passing through the points. Contrast the degree 10 fit in Figure 28 against the degree 5 fit in Figure 27. This is due to the problem of overfitting (overspecification)

Now E is a convex function. To optimize it, we need to set $\nabla_w E = 0$. The ∇ operator is also called gradient.

Solution is given by,

$$X = (\phi^t \phi)^{-1} \phi^t Y$$

If $m << t$ then

- ϕ becomes singular and the solution cannot be found OR
- The column vectors in ϕ become nearly linearly dependent

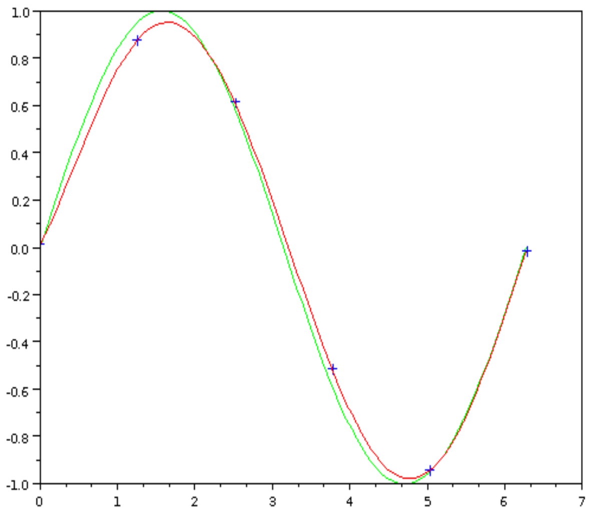


Figure 27: Fit for degree 5 polynomial.

RMS (root mean sqare) error is given by :

$$RMS = \sqrt{\frac{2E}{k}}$$

Generally, some test data (which potentially could have been part of the training data) is held out for evaluating the generalized performance of the model. Another held out fraction of the training data, called the validation dataset is typically used to find the most appropriate degree t_{best} for f.

19.2 Linear regression and method of least squares error

Depending on the function class we consider, there are many types of regression problems. In Linear regression we consider only linear functions, functions that are linear in the basis function. Here $\mathcal{F}$ is of the form $\{\sum_{i=1}^{p} w_i\phi_i(\mathbf{x})\}$. $\phi_i : \mathbb{R}^n \to \mathbb{R}^k$ Here, the ϕ_i's are called the **basis functions** (for example, we can consider $\phi_i(x) = x^i$, *i.e.*, polynomial basis functions) .

Any function in $\mathcal{F}$ is characterized by its parameters, the w_i's. Thus, in (28) we have to find $f(\mathbf{w}^*)$ where

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \varepsilon(\mathbf{w}, \mathcal{D})$$

Least square solution

The error function ε plays a major role in the accuracy and tractability of the optimization problem. The error function is also called the **loss function**. The squared loss is a commonly used loss function. It is the sum of squares of the differences between the actual value and the predicted

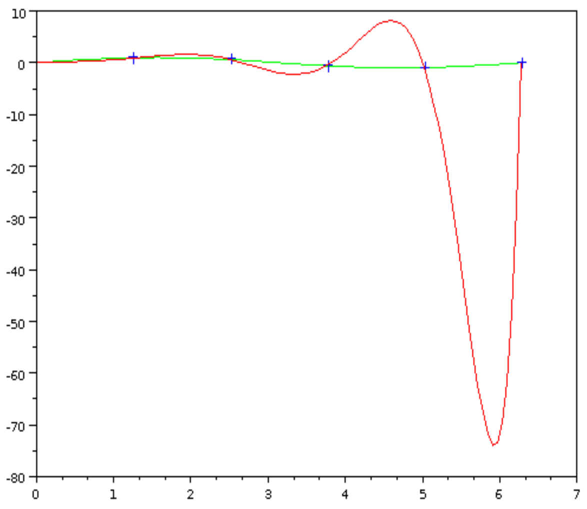


Figure 28: Fit for degree 10 polynomial. Note how wobbly this fit is.

value.

$$\varepsilon(f, \mathcal{D}) = \sum_{\langle \mathbf{x}_i, y_i \rangle \in \mathcal{D}} (f(\mathbf{x}_i) - y_i)^2$$

So the least square solution for linear regression is given by

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{j=1}^{m} \Big(\sum_{i=1}^{p} (w_i \phi_i(x_j) - y_j \Big)^2$$

The minimum value of the squared loss is zero. Is it possible to achieve this value ? In other words is $\forall j, \ \sum_{i=1}^{p} w_i \phi_i(x_j) = y_j$ possible ?

The above equality can be written as $\forall u, \ \phi^T(x_u)\mathbf{w} = y_u$
or equivalently $\phi\mathbf{w} = \mathbf{y}$ where

$$\phi = \begin{bmatrix} \phi_1(\mathbf{x}_1) & \cdots & \phi_p(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ \phi_1(\mathbf{x}_m) & \cdots & \phi_p(\mathbf{x}_m) \end{bmatrix} \text{ and } \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

It has a solution if $\mathbf{y}$ is in the column space (the subspace of $\mathbb{R}^n$ formed by the column vectors) of ϕ. It is possible that there exists no $\mathbf{w}$ which satisfies the conditions? In such situations we can solve the least square problem.

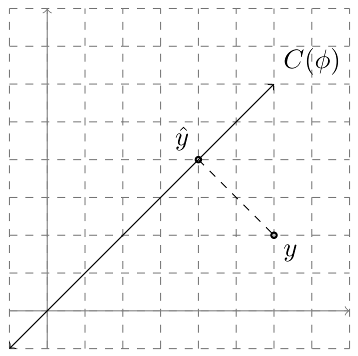


Figure 29: Least square solution $\hat{y}$ is the orthogonal projection of y onto column space of ϕ

Geometrical interpretation of least squares

Let $\hat{\mathbf{y}}$ be a solution in the column space of ϕ. The least squares solution is such that the distance between $\hat{\mathbf{y}}$ and $\mathbf{y}$ is minimized. From the diagram it is clear that for the distance to be minimized, the line joining $\hat{\mathbf{y}}$ to $\mathbf{y}$ should be orthogonal to the column space. This can be summarized as

1. $\phi\mathbf{w} = \hat{\mathbf{y}}$
2. $\forall v \in \{1, ..p\},\ (\mathbf{y} - \hat{\mathbf{y}})^T\phi_v = 0$ or $(\hat{\mathbf{y}} - \mathbf{y})^T\phi = 0$

$$\begin{aligned}
\hat{\mathbf{y}}^T\phi &= \mathbf{y}^T\phi \\
ie,\ (\phi\mathbf{w})^T\phi &= \mathbf{y}^T\phi \\
ie,\ \mathbf{w}^T\phi^T\phi &= \mathbf{y}^T\phi \\
ie,\ \phi^T\phi\mathbf{w} &= \phi^T\mathbf{y} \\
\therefore\ \mathbf{w} &= (\phi^T\phi)^{-1}\mathbf{y}
\end{aligned}$$

In the last step, please note that, $\phi^T\phi$ is invertible only if ϕ has full column rank.

Theorem: If ϕ has full column rank, $\phi^T\phi$ is invertible. A matrix is said to have full column rank if all its column vectors are linearly independent. A set of vectors $\mathbf{v}_i$ is said to be linearly independent if $\sum_i \alpha_i\mathbf{v}_i = 0 \Rightarrow \alpha_i = 0$.

Proof: Given that ϕ has full column rank and hence columns are linearly independent, we have that $\phi\mathbf{x} = 0 \Rightarrow \mathbf{x} = \mathbf{0}$.

Assume on the contrary that $\phi^T\phi$ is non invertible. Then $\exists\mathbf{x} \neq \mathbf{0} \ni \phi^T\phi\mathbf{x} = \mathbf{0}$.

$\Rightarrow \mathbf{x}^T\phi^T\phi\mathbf{x} = 0$

$\Rightarrow (\phi\mathbf{x})^T\phi\mathbf{x} = ||\phi\mathbf{x}||^2 = 0$

$\Rightarrow \phi\mathbf{x} = \mathbf{0}$. This is a contradiction. Hence the theorem is proved.

19.3 Regularised solution to regression

In last lecture, we derived solution for the regression problem formulated in least-squares sense which was aimed at minimizing rms error over observed data points. We also analysed conditions under which the obtained solution was guaranteed to be a global minima. However, as we observed, increasing the order of the model yielded larger rms error over test data, which was due to large fluctuations in model learnt and consequently due to very high values of model coefficients (weights). In this lecture, we discuss how the optimization problem can be modified to counter very large magnitudes of coefficients. Subsequently, solution of this problem is provided through lagrange dual formulation followed by discussion over obtained solution and impact over test data. Towards the end of the lecture, a very gentle introduction to axiomatic probability is provided.

Problem formulation

In order to cease coefficients from becoming too large in magnitude, we may modify the problem to be a constrained optimization problem. Intuitively, for achieving this criterion, we may impose constraint on magnitude of coefficients. Any norm for this purpose might give good working solution. However, for mathematical convenience, we start with the euclidean (L_2) norm. The overall problem with objective function and constraint goes as follows:

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & (\Phi\mathbf{w} - Y)^T(\Phi\mathbf{w} - Y) \\ \text{such that} \quad & ||\mathbf{w}||_2^2 \leq \xi \end{aligned} \tag{29}$$

As observed in last lecture, the objective function, namely $f(\mathbf{w}) = (\Phi\mathbf{w} - Y)^T(\Phi\mathbf{w} - Y)$ is strictly convex. Further to this, the constraint function, $g(\mathbf{w}) = \| \mathbf{w} \|_2^2 - \xi$, is also a convex function. For convex $g(\mathbf{w})$, the set $S = \{\mathbf{w} | g(\mathbf{w}) \leq 0\}$, can be proved to be a convex set by taking two elements $w_1 \in S$ and $w_2 \in S$ such that $g(w_1) \leq 0$ and $g(w_2) \leq 0$. Since $g(\mathbf{w})$ is a convex function, we have the following inequality:

$$\begin{aligned} g(\theta w_1 + (1-\theta)w_2) &\leq \theta g(w_1) + (1-\theta)g(w_2) \\ &\leq 0; \forall \theta \in [0,1], w_1, w_2 \in S \end{aligned} \tag{30}$$

As $g(\theta w_1 + (1-\theta)w_2) \leq 0$; $\forall \theta \in S$, $\forall w_1, w_2 \in S$, $\theta w_1 + (1-\theta)w_2 \in S$, which is both sufficient and necessary for S to be a convex set. Hence, function $g(\mathbf{w})$ imposes a convex constraint over the solution space.

Bound on λ in the regularized least square solution

As discussed earlier, we need to minimize the error function subject to constraint $\|\mathbf{w}\| \leq \xi$. Applying KKT conditions to this problem, if $\mathbf{w}^*$ is a global optimum then from the first KKT condition we get,

$$\nabla_{\mathbf{w}^*}(f(\mathbf{w}) + \lambda g(\mathbf{w})) = 0 \tag{31}$$

where, $f(\mathbf{w}) = (\Phi\mathbf{w} - Y)^T(\Phi\mathbf{w} - Y)$ and $g(\mathbf{w}) = \|\mathbf{w}\|^2 - \xi$
Solving we get,

$$2(\Phi^T\Phi)\mathbf{w}^* - 2\Phi^T - 2\lambda\mathbf{w}^* = 0$$

i.e.

$$\mathbf{w}^* = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T\mathbf{y} \tag{32}$$

From the second KKT condition we get,

$$\|\mathbf{w}^*\|^2 \leq \xi \tag{33}$$

From the third KKT condition,

$$\lambda \geq 0 \tag{34}$$

From the fourth condition

$$\lambda\|\mathbf{w}^*\|^2 = \lambda\xi \tag{35}$$

Thus values of $\mathbf{w}^*$ and λ which satisfy all these equations would yield an optimum solution. Consider equation (32),

$$\mathbf{w}^* = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T\mathbf{y}$$

Premultiplying with $(\Phi^T\Phi + \lambda I)$ on both sides we have,

$$(\Phi^T\Phi + \lambda I)\mathbf{w}^* = \Phi^T\mathbf{y}$$

$$\therefore (\Phi^T\Phi)\mathbf{w}^* + (\lambda I)\mathbf{w}^* = \Phi^T\mathbf{y}$$

$$\therefore \|(\Phi^T\Phi)\mathbf{w}^* + (\lambda I)\mathbf{w}^*\| = \|\Phi^T\mathbf{y}\|$$

By triangle inequality,

$$\|(\Phi^T\Phi)\mathbf{w}^*\| + (\lambda)\|\mathbf{w}^*\| \geq \|(\Phi^T\Phi)\mathbf{w}^* + (\lambda I)\mathbf{w}^*\| = \|\Phi^T\mathbf{y}\| \tag{36}$$

Now , $(\Phi^T\Phi)$ is a nxn matrix which can be determined as Φ is known .
$\|(\Phi^T\Phi)\mathbf{w}^*\| \leq \alpha\|\mathbf{w}^*\|$ for some α for finite $|(\Phi^T\Phi)\mathbf{w}^*\|$. Substituting in previous equation,

$$(\alpha + \lambda)\|\mathbf{w}^*\| \geq \|\Phi^T\mathbf{y}\|$$

i.e.

$$\lambda \geq \frac{\|\Phi^T\mathbf{y}\|}{\|\mathbf{w}^*\|} - \alpha \tag{37}$$

Note that when $\|\mathbf{w}^*\| \to 0, \lambda \to \infty$. This is obvious as higher value of λ would focus more on reducing value of $\|\mathbf{w}^*\|$ than on minimizing the error function.

$$\|\mathbf{w}^*\|^2 \leq \xi$$

Eliminating $\|\mathbf{w}^*\|$ from the equation (14) we get,

$$\therefore \lambda \geq \frac{\|\Phi^T\mathbf{y}\|}{\sqrt{\xi}} - \alpha \tag{38}$$

This is not the exact solution of λ but the bound (15) proves the existance of λ for some ξ and Φ.

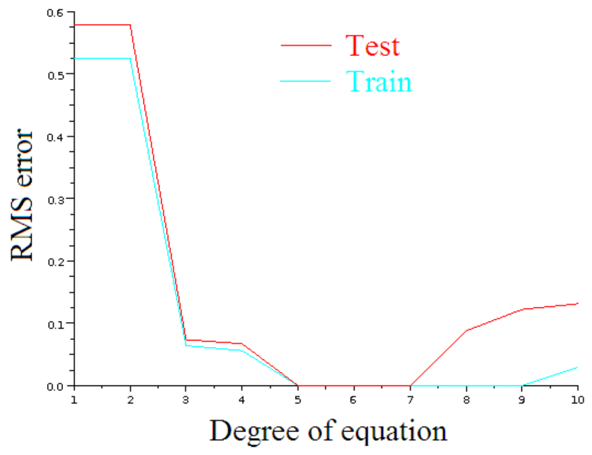


Figure 30: RMS error vs. degree of polynomial for test and train data

RMS Error variation

Recall the polynomial curve fitting problem we considered in earlier lectures. Figure 30 shows RMS error variation as the degree of polynomial (assumed to fit the points) is increased. We observe that as the degree of polynomial is increased till 5 both train and test errors decrease. For degree > 7, test error shoots up. This is attributed to the overfitting problem (The datasize for train set is 8 points.)
Now see Figure 31 where variation in RMS error and Lagrange multiplier λ has been explored (keeping the polynomial degree constant at 6). Given this analysis, what is the optimum value of λ that must be chosen? We have to choose that value for which the test error is minimum (Identified as optimum in the figure.).

Alternative objective function

Consider equation (31). If we substitute $g(\mathbf{w}) = \|\mathbf{w}\|^2 - \xi$, we get

$$\nabla_{\mathbf{w}^*}(f(\mathbf{w}) + \lambda \cdot (\|\mathbf{w}\|^2 - \xi)) = 0 \tag{39}$$

This is equivalent to finding

$$\min(\| \Phi\mathbf{w} - \mathbf{y} \|^2 + \lambda \| \mathbf{w} \|^2) \tag{40}$$

For same λ these two solutions are the same.This form or regression is known as Ridge regression. If we use L_1 norm then it's called as 'Lasso'. Note that $\mathbf{w}^*$ form that we had derived is valid only for L_2 norm.

19.4 Linear Regression : Drawbacks

The following are the problems with linear regression model for classification:

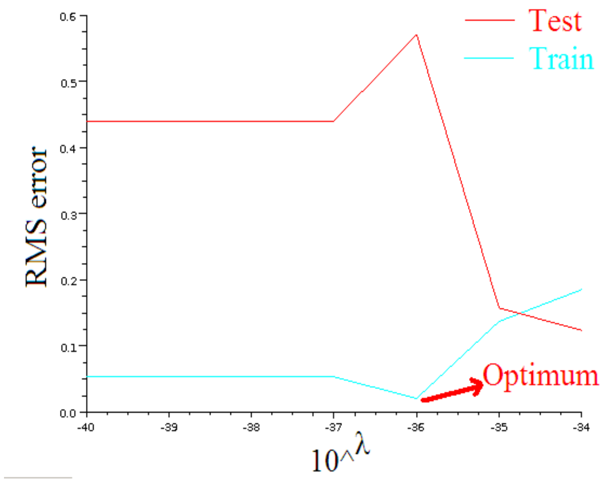


Figure 31: RMS error vs. $10^{\lambda}$for test and train data (at Polynomial degree = 6)

1. Sensitivity to outliers
2. Masking

Sensitivity to outliers

Outliers : They are points which have noise and adversely affect the classification.
In the right hand figure , the separating hyperplane has changed because of the outliers.

Masking

It is seen empirically that linear regression classifier may mask a given class. This is shown in the left hand figure. We had 3 classes one in between the other two. The between class points are not classified.
The right hand figure is the desirable classification.

The equation of the classifier between class C1(red dots) and class C2(green dots) is
$(\omega_1 - \omega_2)^T\phi(x) = 0$
and the equation of the classifier between the classes C2(green dots) and C3(blue dots) is
$(\omega_2 - \omega_3)^T\phi(x) = 0$

19.5 Possible solutions

1. **Mapping to new space**

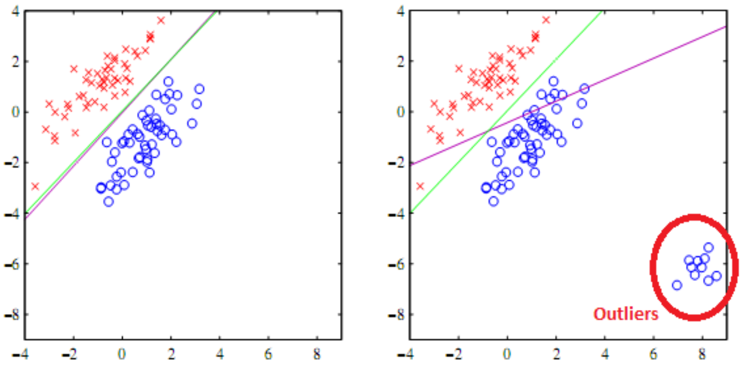


Figure 32: Outliers

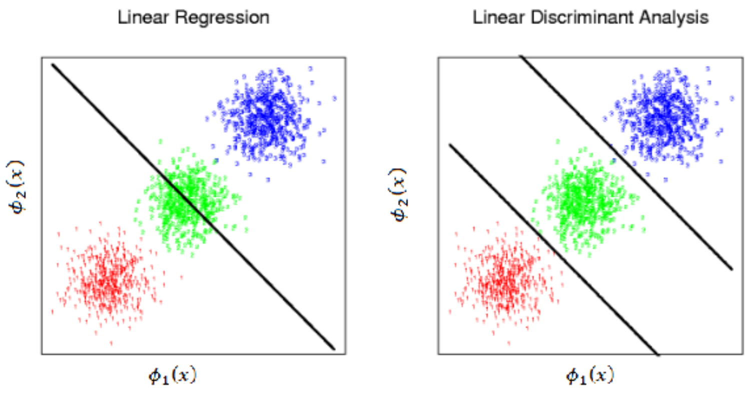


Figure 33: Masking

We will transform the original dimensions to new dimensions. New dimensions are function of original dimensions. This is a work around solution.

$\phi_1'(x) = \sigma_1(\phi_1, \phi_2)$
$\phi_2'(x) = \sigma_2(\phi_1, \phi_2)$

Here we try to determine the transformations ϕ_1' and ϕ_2' such that we can get a linear classifier in this new space. When we map back to the original dimensions , the separators may not remain linear.

Problem : Exponential blowup of number of parameters $(w's)$ in order $O(n^{k-1})$.

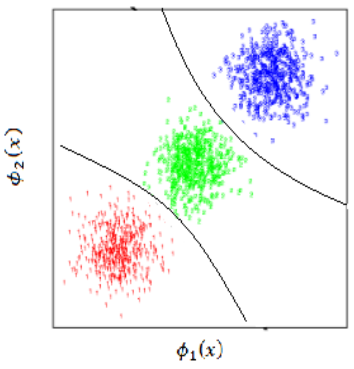


Figure 34: Mapping back to original dimension class separator not linear

2. **Decision surface perpendicular bisector to the mean connector.**

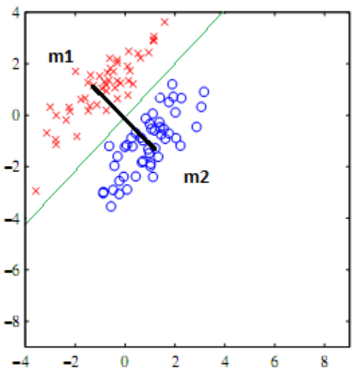


Figure 35: Class separator perpendicular to the line joining mean

Decision surface is the perpendicular bisector of the line joining mean of class $C_1(m_1)$ and mean of class $C_2(m_2)$.

$m_1 = (1/N_1)\sum_{n \in C_1} x_n$ where m_1 is the mean of class C_1 and N_1 is the number of points in class C_1.

$m_2 = (1/N_2)\sum_{n \in C_2} x_n$ where m_2 is the mean of class C_2 and N_2 is the number of points in class C_2.

$||\phi(x) - m_1|| < ||\phi(x) - m_2|| => x \in C_1$

$||\phi(x) - m_2|| < ||\phi(x) - m_1|| => x \in C_2$

Comment : This is solving the masking problem but not the sensitivity problem as this does not capture the orientation(eg: spread of the data points) of the classes.

3. **Fisher Discrimant Analysis.**

Here we consider the mean of the classes , within class covariance and global covariance.
Aim : To increase the separation between the class means and to minimize within class variance. Considering two classes.
S_B is Inter class covariance and S_W is Intra class covariance.

$m_1 = (1/N_1)\sum_{n \in C_1} x_n$ where m_1 is the mean of class C_1 and N_1 is the number of points in class C_1.

$m_2 = (1/N_2)\sum_{n \in C_2} x_n$ where m_2 is the mean of class C_2 and N_2 is the number of points in class C_2.

$N_1 + N_2 = N$ where N is the total number of training points.

$S_B = (m_2 - m_1)(m_2 - m_1)^T$

$S_W = \sum_{n \in C_1}(x_n - m_1)(x_n - m_1)^T + \sum_{n \in C_2}(x_n - m_2)(x_n - m_2)^T$

$J(w) = (w^T S_B w)/(w^T S_w w)$

By maximizing $J(w)$ we get the following:

$w \alpha S_w^{-1}(m_2 - m_1)$

Summary

	Sensitivity to outliers	**Masking**
Perpendicular Bisector of means connector	Does not solve	Solves
Fischer Discriminant	Does not solve	Solves

We have seen that the fisher discriminant analysis is better compared to the other two possible solutions.

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 26: Perceptron Algorithm, Support Vector Machines

Instructor: *Ganesh Ramakrishnan* — Date: *02/11/2011*
Computer Science & Engineering — Indian Institute of Technology, Bombay

20 Perceptron Algorithm

We saw a class of classifiers that model $w^T\phi(x)$ directly. Among them were *least squared error classification*, *perpendicular bisector of line connecting the mean points* and *fisher discriminant analysis*. All these models have the problem that they are not robust to outliers. They are extremely sensitive to them, in the sense that a few outlier points can drastically change the position and orientation of the decision surface.

20.1 Introduction

Desirables for avoiding sensitivity to outliers

1. Few points properly classified and far away from the separating surface (decision boundary) should not influence the decision boundary much.
2. (Possibly) few misclassified points far away from the separating surface should also not influence the decision boundary.

In Perceptron algorithm the main idea is to *learn w (for $w^T\phi(x) = 0$) only from misclassified examples weighing them by their distance from the separating hyperplane.*

A misclassified example is defined as

$$w^T\phi(x_i) > 0 \text{ for } y_i = -1$$
$$w^T\phi(x_i) < 0 \text{ for } y_i = +1$$

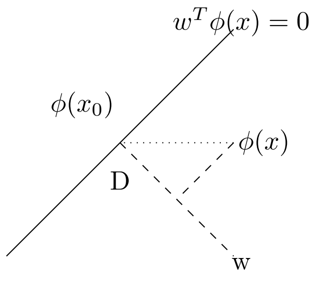
Combining the above two equations we get,

$$y_i w^T\phi(x_i) < 0$$

20.2 Training algorithm

As said earlier, perceptron explicitly accounts for the signed distribution of misclassified points from hyperplane. $w^T\phi(x) = 0$.

Distance from hyperplane can be calculated as follows



$$D = w^T(\phi(x) - \phi(x_0))$$

Since $w^T(\phi(x_0)) = 0$, we get distance $= w^T(\phi(x))$. **Note:** We study perceptron (and later SVM) for 2-class classification problems only. We label them as y=1 and y=-1. A point is misclassified if $y_i w^T(\phi(x)) < 0$

Algorithm 2 Perceptron Algorithm

```
w=ones() {Initialization}
loop
  if given $< x, y >, w^T\Phi(x).y \leq 0$ then
    $w = w + \Phi(x).y$
  end if
end loop
```

Intuition

$$\begin{aligned} y w_{k+1}^T \phi(x) &= y(w_k + y\phi(x)^T\phi(x) \\ &= y w_k^T \phi(x) + y^2 \|\phi(w)\|^2 \\ &> y w_k^T \phi(x) \end{aligned}$$

Note: We applied the update for this point, (since $y w_k^T \phi(x) \leq 0$) We have $y w_k^T \phi(x) \; > \; y w_k^T \phi(x)$. So we have more hope that this point is classified correctly now. More formally, perceptron tries to minimize the error function

$$E = -\sum_{x \in M} y \phi^T(x) \omega$$

where M is the set of misclassified examples.

Stochastic Gradient Descent algorithm

Perceptron algorithm is **similar** (Its not exactly equivalent) to a gradient descent algorithm, which can be shown as follows: Since ∇E is given by $\nabla E = -\sum\limits_{x \in M} y\phi(x)$. So,

$$\begin{aligned} w_{k+1} &= w_k - \eta \nabla E \\ &= w_k + \eta \sum_{x \in M} y\phi(x) \quad \text{(This takes all misclassified points at a time)} \end{aligned}$$

But what we are doing in standard Perceptron Algorithm, is basically *Stochastic Gradient Descent*:

$$\nabla E = -\sum_{x \in M} y\phi(x) = -\sum_{x \in M} \nabla E(x) \quad , \text{ where } E(x) = y\phi(x)$$

$$\begin{aligned} w_{k+1} &= w_k - \eta \nabla E(x) \\ &= w_k + \eta y \phi(x) \qquad\qquad \text{(for any } x \in M) \end{aligned}$$

If $\exists$ an optimal separating hyperplane with parameters w^* such that,

$$\phi^T(x) w^* = 0$$

then perceptron algorithm converges.

Proof:-

$$\lim_{k \to \infty} \|w_{k+1} - \rho w^*\|^2 = 0 \quad \text{(If this happens for some constant } \rho\text{, we are fine.)}$$
$$\|w_{k+1} - \rho w^*\|^2 = \|w_k - \rho w^*\|^2 + \|y\phi(x)\|^2 + 2y(w_k - \rho w^*)^T \phi(x)$$

Now, we want L.H.S. to be less than R.H.S. at every step, although by some small value, so that perceptron will converge overtime.
So, if we can obtain an expression of the form:

$$\|w_{k+1} - \rho w^*\|^2 < \|w_k - \rho w^*\|^2 - \theta^2$$

Then, $\|w_{k+1} - \rho w^*\|^2$ is reducing by atleast θ^2 at every iteration. So, from the above expressions, we need to find θ such that,

$$\|\phi(x)\|^2 + 2y(w_k - \rho w^*)^T \phi(x) < -\theta^2$$

(Here, $\|y\phi(x)\|^2 = \|\phi(x)\|^2$ because $\|y\| = 1, y$ is either $+1$ or -1)

So, the no. of iterations would be: $O\left(\frac{\|w_0 - \rho w^*\|^2}{\theta^2}\right)$

Some observations

1. $y w_k^T \phi(x) \leq 0$ ($\because x$ was misclassified)

2. $\Gamma^2 = \max_{x\in\mathcal{D}} \|\phi(x)\|^2$
3. $\delta = \max_{x\in\mathcal{D}} -2y{w^*}^T\phi(x)$

Here, margin $= {w^*}^T\phi(x) =$ dist. of closest point from hyperplane and, $\mathcal{D}$ is the set of all points, **not** just misclassifed ones.

$$\begin{aligned}\delta &= \max_{x\in\mathcal{D}} -2y{w^*}^T\phi(x)\\ &= \min_{x\in\mathcal{D}} y{w^*}^T\phi(x)\end{aligned}$$

Since, ${w^*}^T\phi(x) \geq 0$, so, $\delta \leq 0$. So, what we are interested in, is the 'least negative' value of δ

From the observations, and eq.(2), we have:

$$0 \leq \|w_{k+1} - \rho w^*\|^2 \leq \|w_k - \rho w^*\|^2 + \Gamma^2 + \rho\delta$$

Taking, $\rho = \frac{2\Gamma^2}{-\delta}$, then,

$$0 \leq \|w_{k+1} - \rho w^*\|^2 \leq \|w_k - \rho w^*\|^2 - \Gamma^2$$

Hence, we have, $\Gamma^2 = \theta^2$, that we were looking for in eq.(3). $\therefore$ $\|w_{k+1} - \rho w^*\|^2$ decreases by atleast Γ^2 at every iteration.

Here the notion of convergence is that w_k converges to ρw^* by making atleast some decrement at each step. Thus, for $k \to \infty$, $\|w_k - \rho w^*\| \to 0$. Hence, the proof of convergence.

20.3 Issues with Perceptron

1. In the non-separable case it may oscillate a lot and is super-sensitive to initialization of non-separable cases.
2. Can tend to overfit. (when we have very less bias)

21 Support Vector Machines

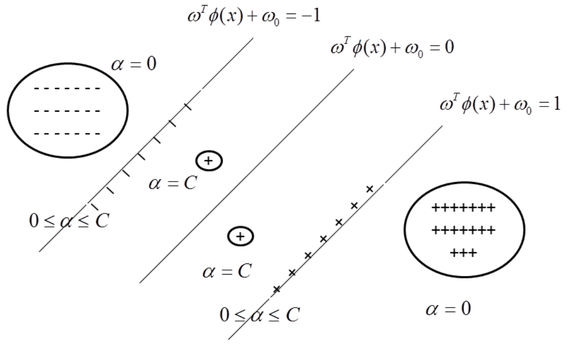
The main idea behind *support vector machines* is to find a w that mazimises the unsigned distance of the closest point to the separating surface.

So we have the formulation as follows:

$$\underset{w}{max} \quad \underset{i}{min} \quad \frac{(w^T\phi(x_i) + w_0)y_i}{||w||} \tag{41}$$

$$\underset{w}{max}\frac{1}{||w||} \quad \underset{i}{min} \quad (w^T\phi(x_i) + w_0)y_i \tag{42}$$

However, the catch is : if w^*, w_0^* is a solution, then $\lambda w^*, \lambda w_0^*$ is also a solution.



$\omega^T\phi(x)+\omega_0=-1$
$\omega^T\phi(x)+\omega_0=0$
$\omega^T\phi(x)+\omega_0=1$
$\alpha=0$
$\alpha=C$
$0\le\alpha\le C$
$\alpha=C$
$0\le\alpha\le C$
$\alpha=0$

Figure 36: Different types of points

21.1 Canonical SVM problem

Since many solutions are possible by scaling w and w_0 as noted above, we restricy our attention to a canonical (w, w_0) for which, $\underset{i}{min}(w^T\phi(x_i)+w_0)y_i = 1$

So we get,

$$\boxed{\underset{w}{max}\frac{1}{||w||} \quad \text{, s.t. } \forall i, (w^T\phi(x_i)+w_0)y_i \geq 1} \tag{43}$$

Any scaled solution (w^*, w_0^*) to Equation 43 is a solution to Equations 41/42. Equivalent to Equation 43 is also the following equation (By monotonic transformations)

$$\boxed{\underset{w}{min}||w||^2 \quad \text{, s.t. } \forall i, (w^T\phi(x_i)+w_0)y_i \geq 1}$$

The above objective $||w||^2$ is called the *regularizer / bias.*

Slack: to handle non-separability

$$\min_{w,w_0} ||w||^2 + c\sum_i \xi_i \tag{44}$$
$$s.t.\forall i \tag{45}$$
$$y_i(\phi^T(x_i)w + w_0') \geq 1 - \xi_i \tag{46}$$
$$where, \tag{47}$$
$$\forall i \xi_i \geq 0 \tag{48}$$

In soft margin we account for the the errors. The above formulation is one of the many formulation of soft SVMs. In the above formulation, large value of c means overfitting.

Three types of g points

In Figure 36 we can see three types of points. They are:

1. Correctly classified but $\xi_i > 0$ or violates margin
2. Correctly classified but $\xi_i = 0$ or on the margin
3. Inorrectly classified but $\xi_i > 1$

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 27: SVM: Dual Formulation, Notion of Kernel

Instructor: *Ganesh Ramakrishnan* — Date: *05/11/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

21.2 SVM : Dual Formulation

Primal formulation

$$
\begin{aligned}
p^* &= \min f(x) && (49)\\
x &\in D && (50)\\
\text{s.t. } g_i(x) &\leq 0 && (51)\\
i &= 1, \ldots, m && (52)
\end{aligned}
$$

Dual Formulation

$$
d^* = \max_{\lambda \in \mathbb{R}} \min_{x \in D} \left(f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \right) \tag{53}
$$

$$
\text{s.t. } \lambda_i \geq 0 \tag{54}
$$

Equation 53 is and convex optimization problem. Also, $d^* \leq p^*$ and $(p^* - d^*)$ is called the duality gap.

If for some (x^*, λ^*) where x^* is primal feasible and λ^* is dual feasible and we see the KKT conditions are satisfied and f is and all g_i are convex then x^* is optimal solution to primal and λ^* to dual.

Also, the dual optimization problem becomes,

$$
\begin{aligned}
d^* &= \max_{\lambda \in \mathbb{R}^m} \mathcal{L}(x^*, \lambda) && (55)\\
\text{s.t. } \lambda_i &\geq 0 \forall i && (56)\\
\text{where } \mathcal{L}(x, \lambda) &= f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) && (57)\\
\mathcal{L}^*(\lambda) &= \min_{x \in D} \mathcal{L}(x, \lambda) && (58)\\
&= \min_{x \in KKT} \mathcal{L}(x, \lambda) && (59)\\
\lambda_i &\geq 0 \forall i && (60)
\end{aligned}
$$

It happens to be,

$$
p^* = d^* \tag{61}
$$

21.3 Duality theory applied to KKT

$$\mathcal{L}(\bar{w}, \bar{\xi}, w_0, \bar{\alpha}, \bar{\lambda}) = \frac{1}{2} ||w||^2 + c \sum_{i=1}^{m} \xi_i + \sum_{i=1}^{m} \alpha_i \left[1 - \xi_i - y_i \left(\phi^T(x_i) w + w_0\right)\right] - \sum_{i=1}^{m} \lambda_i \xi_i \tag{62}$$

Now we check for KKT conditions at the point of optimality,

KKT 1.a

$$\nabla_w \mathcal{L} = 0 \tag{63}$$

$$\implies w - \sum_{j=1}^{n} \alpha_j y_j \phi^T(x_j) = 0 \tag{64}$$

KKT 1.b

$$\nabla_{xi_i} \mathcal{L} = 0 \tag{65}$$

$$\implies c - \alpha_i - \lambda_i = 0 \tag{66}$$

KKT 1.c

$$\nabla_{w_0} \mathcal{L} = 0 \tag{67}$$

$$\implies \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{68}$$

KKT 2

$$\forall i \tag{69}$$

$$y_i \left(\phi^T(x_i) w + w_0\right) \geq 1 - \xi_i \tag{70}$$

$$\xi_i \geq 0 \tag{71}$$

KKT 3

$$\alpha_j \geq 0 \text{ and } \lambda_k \geq 0 \tag{72}$$

$$\forall j, k = 1, \ldots, n \tag{73}$$

KKT 4

$$\alpha_j \left[y_i \left(\phi^T(x_j) w + w_0\right) - 1 + \xi_j\right] = 0 \tag{74}$$

$$\lambda_k \xi_k = 0 \tag{75}$$

(a)

$$w^* = \sum_{j=1}^{m} \alpha_j y_i \phi(x_j) \tag{76}$$

w^* is weighted linear combination of points $\phi(x)$s.

(b)

If $0 < \alpha_j < c$ then, by Equation 66
$0 < \lambda_j < c$ and by Equation 75, $\xi_j = 0$ and $y_i \left(\phi^T(x_j)w + w_0\right) = 1$

If however, $\alpha_j = c$ then $\lambda_j = 0$ and $y_i \left(\phi^T(x_j)w + w_0\right) \leq 1$.

If α_0 then $\lambda_j = c$ and $\xi_j = 0$, we get $y_i \left(\phi^T(x_j)w + w_0\right) \geq 1$. Then $\alpha_j = 0$

21.4 SVM dual

SVM can be formulated as the following optimization problem,

$$\min_w \{\frac{1}{2} \|w\|^2 + C \sum_{i=0}^{m} \xi_i\}$$

subject to constraint,

$$\forall i : y_i(\phi^T(x_i)w + w_0) \geq 1 - \xi_i$$

The dual of the SVM optimization problem can be stated as,

$$\max\{-\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \phi^T(x_i)\phi(x_j) + \sum_{j=1}^{m} \alpha_j\}$$

subject to constraints,

$$\forall i : \sum_i \alpha_i y_i = 0$$
$$\forall i : 0 \leq \alpha_i \leq c$$

The duality gap $= f(x^*) - L^*(\lambda^*) = 0$, as shown in last lecture. Thus, as is evident from the solution of the dual problem,

$$w^* = \sum_{i=1}^{m} \alpha_i^* y_i \phi(x_i)$$

To obtain w_o^*, we can use the fact (as shown in last lecture) that, if $\alpha_i \in (0, C)$, $y_i(\phi^T(x_i)w + w_0) = 1$. Thus, for any point x_i such that, $\alpha_i \in (0, C)$, that is, α_i is a point on the margin,

$$\begin{aligned} w_o^* &= \frac{1 - y_i(\phi^T(x_i)w^*)}{y_i} \\ &= y_i - \phi^T(x_i)w^* \end{aligned}$$

The decision function,

$$\begin{aligned} g(x) &= \phi^T(x)w^* + w_0^* \\ &= \sum_{i=0}^{m} \alpha_i y_i \phi^T(x)\phi(x_i) + w_0^* \end{aligned}$$

21.5 Kernel Matrix

A kernel matrix

$$K = \begin{bmatrix} \phi^T(x_1)\phi(x_1) & \phi^T(x_1)\phi(x_2) & \dots & \dots & \phi^T(x_1)\phi(x_n) \\ \phi^T(x_2)\phi(x_1) & \phi^T(x_2)\phi(x_2) & \dots & \dots & \phi^T(x_2)\phi(x_n) \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \phi^T(x_n)\phi(x_1) & \phi^T(x_n)\phi(x_2) & \dots & \dots & \phi^T(x_n)\phi(x_n) \end{bmatrix}$$

In other words, $K_{ij} = \phi^T(x_i)\phi(x_j)$. The SVM dual can now be re-written as,

$$\max\{-\frac{1}{2}\alpha^T K_y \alpha + \alpha^T ones(m, 1)\}$$

subject to constraints,

$$\begin{aligned} \sum_i \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq c \end{aligned}$$

Thus, for $\alpha_i \in (0, C)$

$$\begin{aligned} w_0^* &= y_i - \phi^T(x_i)w \\ &= y_i - \sum_{j=0}^{m} \alpha_j^* y_j \phi^T(x_i)\phi(x_j) \\ &= y_i - \sum_{j=0}^{m} \alpha_j^* y_j K_{ij} \end{aligned}$$

Generation of ϕ space

For a given $\mathbf{x} = [x_1, x_2, \dots, x_n] \to \phi(x) = [x_1^d, x_2^d, x_3^d, \dots, x_1^{d-1}x_2, \dots]$.
For $n = 2, d = 2$, $\phi(x) = [x_1^2, x_1x_2, x_2x_1, x_2^2]$, thus,

$$\begin{aligned} \phi^T(x).\phi(\bar{x}) &= \sum_{i=1}^{m}\sum_{j=1}^{m} x_i x_j.\bar{x}_i\bar{x}_j \\ &= (\sum_{i=1}^{m} x_i\bar{x}_i).(\sum_{j=1}^{m} x_j\bar{x}_j) \\ &= (\sum_{i=1}^{m} x_i\bar{x}_i)^2 \\ &= (x^T\bar{x})^2 \end{aligned}$$

In general, for $n \geq 1$ and $d \geq 1$, $\phi^T(x).\phi(\bar{x}) = (x^T\bar{x})^d$.
A polynomial kernel, in general, is defined as $K_{ij} = (x_i^T x_j)^d$.

CS 725 : FOUNDATIONS OF MACHINE LEARNING — AUTUMN 2011

Lecture 28: SVM: Kernel Methods, Algorithms for solving Dual

Instructor: *Ganesh Ramakrishnan* — Date: *07/11/2011*
COMPUTER SCIENCE & ENGINEERING — INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY

21.6 Requirements of Kernel

1. Since

$$\begin{aligned} K_{ij} &= \phi^T(x_i)\phi(x_j) \\ &= \phi^T(x_j)\phi(x_i) \end{aligned}$$

   Hence K should be a Symmetric Matrix.

2. The Cauchy Schwarz Inequality

$$(\phi^T(x)\phi(\bar{x}))^2 \leq \|\phi^T(x)\|^2\|\phi(\bar{x})\|^2$$

$$\Rightarrow {K_{ij}}^2 \leq K_{ii}K_{jj}$$

3. Positivity of Diagonal

$$K = V\Lambda V^T$$

   Where V is the eigen vector matrix (an orthogonal matrix), and Λ is the Diagonal matrix of eigen values.

Goal is to construct a ϕ. Which can be constructed as

$$\phi(x_i) = \sqrt{\lambda_i}V_i \quad (\lambda_i \geq 0)$$
$$K_{ii} = \lambda_i\|V_i\|^2$$

Hence K must be

1. Symmetric.
2. Positive Semi Definite.
3. Having non-negative Diagonal Entries.

Examples of Kernels

1. $K_{ij} = ({x_i}^T x_j)^d$
2. $K_{ij} = ({x_i}^T x_j + 1)^d$
3. Gaussian or Radial basis Function (RBF)
   $K_{ij} = e^{-\frac{\|x_i - x_j\|}{2\sigma^2}}$ $(\sigma \in R,\ \sigma \neq 0)$
4. The Hyperbolic Tangent function
   $K_{ij} = \tanh(\sigma x_i^T x_j + c)$

Properties of Kernel Functions

If K' and K'' are Kernels then K is also a Kernel if either of the following holds

1. $K_{ij} = {K'}_{ij} + {K''}_{ij}$
2. $K_{ij} = \alpha {K'}_{ij}$ $(\alpha \geq 0)$
3. $K_{ij} = {K'}_{ij} {K''}_{ij}$

Proof : (1) and (2) are left as an exercise.
(3)

$$\begin{aligned} K_{ij} &= K'_{ij} K''_{ij} \\ &= \phi'^T(x'_i)\phi'(x'_j) * \phi''^T(x''_i)\phi''(x''_j) \end{aligned}$$

Define $\phi(x_i) = \phi'^T(x'_i)\phi''^T(x''_i)$. Thus, $K_{ij} = \phi(x_i)\phi(x_j)$.
Hence, K is a valid kernel.

21.7 Algorithms for solving the dual

Duality offers multiple alternative check points to see if the solution is optimal. They are

1. KKT conditions satisfied $\forall i$
2. Primal objective $\approx$ Dual objective

We prefer solving the dual since we have the kernel and can avoid computing complex ϕ.
($K = x^T \bar{x}$ i.e $\phi(x) = x$ .. However, linear kernel has simple ϕ and could be solved in primal form)

Sequential Minimal Optimization Algorithm (SMO)

It turns out that for most solutions, most $\alpha_i = 0$. So general (LCQP) solvers are an overkill. To explot this, we use batch co-ordinate wise ascent. One of the best performers is the *sequential minimal optimization (SMO)* algorithm.

This optimizes for 2 α's at a time. The steps of the algorithm are:

1. Start with all $\alpha_i = 0$

2. Seclect any 2 αs, say α_1 and α_2 that violate the KKT
3. Solve for α_1 and α_2

$$\begin{aligned}
\min_{\alpha_1,\alpha_2} \quad & -\alpha_1 - \alpha_2 - \sum_{i\neq 1,2} \alpha_i + \frac{1}{2}\alpha_1^2 K_{11} + \alpha_2^2 K_{22} + \alpha_1\alpha_2 K_{12} y_1 y_2 \\
& + \alpha_1 y_1 \sum_{i\neq 1,2} K_{1i}\alpha_i y_i + \alpha_2 y_2 \sum_{i\neq 1,2} K_{2i}\alpha_i y_i \qquad (77) \\
\text{s.t.} \quad & \alpha_1 y_1 + \alpha_2 y_2 = -\sum_{j\neq 1,2} \alpha_j y_j = \alpha_1^{old} + \alpha_2^{old} \\
& \alpha_1, \alpha_2 \in [0, c]
\end{aligned}$$

4. From the second last constraint, we can write α_1 in terms of α_2.

$$\alpha_1 = -\alpha_2 \frac{y_2}{y_1} + \alpha_1^{old} + \alpha_2^{old} \frac{y_2}{y_1}$$

Then the objective is just a function of α_2, let the objective is $-D(\alpha_2)$. Now the program reduces to

$$\begin{aligned}
\min_{\alpha_2} \quad & -D(\alpha_2) \\
\text{s.t.} \quad & \alpha_2 \in [0, c]
\end{aligned}$$

Find α_2^* such that $\frac{\partial D(\alpha_2)}{\partial \alpha_2} = 0$. We have to ensure that $\alpha_1 \in [0, c]$. So based on that we will have to clipp α_2 , ie, shift it to certain interval. The condition is as follows

$$0 <= -\alpha_2 \frac{y_2}{y_1} + \alpha_1^{old} + \alpha_2^{old} \frac{y_2}{y_1} <= c$$

5.
   - case 1: $y_1 = y_2$

   $$\alpha_2 \in [max(0, -c + \alpha_1^{old} + \alpha_2^{old}), min(c, \alpha_1^{old} + \alpha_2^{old})]$$

   - case 2: $y_1 = -y_2$

   $$\alpha_2 \in [max(0, \alpha_2^{old} - \alpha_1^{old}), min(c, c - \alpha_1^{old} + \alpha_2^{old})]$$

   If α_2 is already in the interval then there is no problem. If it is more than the maximum limit then reset it to the maximum limit. This will ensure the optimum value of the objective constrained to this codition. Similarly if α_2 goes below the lower limit then reset it to the lower limit.

Chunking and Decomposition Methods

We are interested in solving dual of the objective because we have already seen that most of the dual variable will be zero in the solution and hence it will give a sparse solution (based on the KKT conidtion).

$$\begin{aligned} \text{Dual:} \quad & \min_{\alpha} && -\sum \alpha_i + \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j K_{ij} \\ & \text{s.t.} && \sum_i \alpha_i y_i = 0 \\ & && \alpha_i \in [0, c] \end{aligned} \tag{78}$$

The above program is a quadratic program. Any quadratic solvers can be used for solving (78), but a generic solver will not take consider speciality of the solution and may not be efficient. One way to solve (78) is by using projection methods(also called Kernel adatron). You can solve the above one using two ways - chunking methods and decomposition methods.

The chunking method is as follows

1. Initialize α_is arbitrarily
2. Choose points(I mean the components α_i) that violate KKT condition
3. Consider only K working set and solve the dual for the variables in working set

$\forall \alpha \in$ working set

$$\begin{aligned} & \min_{\alpha} && -\sum_{\alpha_i inWS} \alpha_i + \frac{1}{2}\sum_{i \in WS} \sum_{j \in WS} \alpha_i \alpha_j y_i y_j K_{ij} \\ & \text{s.t.} && \sum_{i \in WS} \alpha_i y_i = -\sum_{j \notin WS} \alpha_j y_j \\ & && \alpha_i \in [0, c] \end{aligned} \tag{79}$$

4. set $\alpha^{new} = [\alpha^{new}_{WS}, \alpha^{old}_{nonWS}]$

Decompsition methods follow almost the same procedure except that in step 2 we always take a fixed number of points which violate the KKT conditions the most.

Further Reading

For SVMs in general and kernel method in particular read the SVM book *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods by Nello Cristianini and John Shawe-Taylor* uploaded on moodle.

CS 725 : Foundations of Machine Learning — Autumn 2011

Lecture 29: Support Vector Regression, Attribute Selection

Instructor: *Ganesh Ramakrishnan* — Date: *11/11/2011*

Computer Science & Engineering — Indian Institute of Technology, Bombay

22 Support Vector Regression

Please refer to previous years' notes (`http://www.cse.iitb.ac.in/~cs725/notes/classNotes/lecturenote_2010.pdf`) *Section 22.2* for this topic.

23 Attribute Selection and Transformation

Please refer to the following material for this topic:

1. Chapter 7 of book *Data Mining* by I.H. Witten and E. Frank
2. Slides at `http://www.cse.iitb.ac.in/~cs725/notes/classNotes/dataprocessing.pdf`