Supervised Learning: Additional Topics

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

February 2023

This Lecture

- Evaluation metrics for classification
- Decision trees
- Support Vector Machines
- Nearest-neighbour methods
- Supervised learning: summary

This Lecture

- Evaluation metrics for classification
- Decision trees
- Support Vector Machines
- Nearest-neighbour methods
- Supervised learning: summary

Confusion Matrix

True label	Predicted label			
	Poha	Paratha	Idli	Toast
Poha	95	0	2	3
Paratha	0	98	0	2
Idli	1	15	84	0
Toast	0	0	0	100

Useful device to identify weaknesses of model.

Confusion Matrix

True label	**Predicted label**			
	Poha	Paratha	Idli	Toast
Poha	95	0	2	3
Paratha	0	98	0	2
Idli	1	15	84	0
Toast	0	0	0	100

Useful device to identify weaknesses of model.

Confusion Matrix

True label	Predicted label			
	Poha	Paratha	Idli	Toast
Poha	95	0	2	3
Paratha	0	98	0	2
Idli	1	15	84	0
Toast	0	0	0	100

Useful device to identify weaknesses of model.

Confusion Matrix

True label	Predicted label			
	Poha	Paratha	Idli	Toast
Poha	95	0	2	3
Paratha	0	98	0	2
Idli	1	15	84	0
Toast	0	0	0	100

Useful device to identify weaknesses of model.

Confusion Matrix: 2 Classes

True label	**Predicted label**	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Confusion Matrix: 2 Classes

True label	Predicted label	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

- Accuracy = $\frac{TP+TN}{TP+FN+FP+TN}$.

Confusion Matrix: 2 Classes

True label	**Predicted label**	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

- Accuracy = $\frac{TP+TN}{TP+FN+FP+TN}$.
- $TPR = \frac{TP}{TP+FN}$. Also called Sensitivity, Recall.

Confusion Matrix: 2 Classes

True label	**Predicted label**	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

- Accuracy = $\frac{TP+TN}{TP+FN+FP+TN}$.
- $TPR = \frac{TP}{TP+FN}$. Also called Sensitivity, Recall.
- $TNR = \frac{TN}{FP+TN}$. Also called Specificity.

Confusion Matrix: 2 Classes

True label	**Predicted label**	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

- Accuracy = $\frac{TP+TN}{TP+FN+FP+TN}$.
- $TPR = \frac{TP}{TP+FN}$. Also called Sensitivity, Recall.
- $TNR = \frac{TN}{FP+TN}$. Also called Specificity.
  FPR = $\frac{FP}{FP+TN} = 1 - TNR$.

Confusion Matrix: 2 Classes

True label	**Predicted label**	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

- Accuracy = $\frac{TP+TN}{TP+FN+FP+TN}$.
- $TPR = \frac{TP}{TP+FN}$. Also called Sensitivity, Recall.
- $TNR = \frac{TN}{FP+TN}$. Also called Specificity.
  FPR = $\frac{FP}{FP+TN} = 1 - TNR$.
- Precision = $\frac{TP}{TP+FP}$.

Confusion Matrix: 2 Classes

True label	**Predicted label**	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

- Accuracy = $\frac{TP+TN}{TP+FN+FP+TN}$.
- $TPR = \frac{TP}{TP+FN}$. Also called Sensitivity, Recall.
- $TNR = \frac{TN}{FP+TN}$. Also called Specificity.

  FPR = $\frac{FP}{FP+TN} = 1 - TNR$.
- Precision = $\frac{TP}{TP+FP}$.
- F1 score = $\frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$.

This Lecture

- Evaluation metrics for classification
- Decision trees
- Support Vector Machines
- Nearest-neighbour methods
- Supervised learning: summary

Sales Data

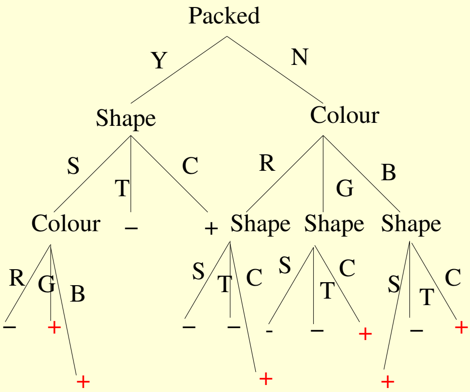
Colour	**Shape**	**Packed**	Sale
Red	Square	Yes	–
Red	Square	No	–
Red	Triangle	Yes	–
Red	Triangle	No	–
Red	Circle	Yes	+
Red	Circle	No	+
Green	Square	No	–
Green	Triangle	Yes	–
Green	Triangle	No	–
Green	Circle	No	+
Blue	Square	Yes	+
Blue	Square	No	+
Blue	Triangle	Yes	–
Blue	Triangle	No	–
Blue	Circle	Yes	+
Blue	Circle	No	+

Sales Data

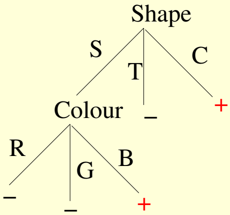
Colour	**Shape**	**Packed**	Sale
Red	Square	Yes	$-$
Red	Square	No	$-$
Red	Triangle	Yes	$-$
Red	Triangle	No	$-$
Red	Circle	Yes	$+$
Red	Circle	No	$+$
Green	Square	No	$-$
Green	Triangle	Yes	$-$
Green	Triangle	No	$-$
Green	Circle	No	$+$
Blue	Square	Yes	$+$
Blue	Square	No	$+$
Blue	Triangle	Yes	$-$
Blue	Triangle	No	$-$
Blue	Circle	Yes	$+$
Blue	Circle	No	$+$

Which products are selling? Which ones are not?

Decision Trees

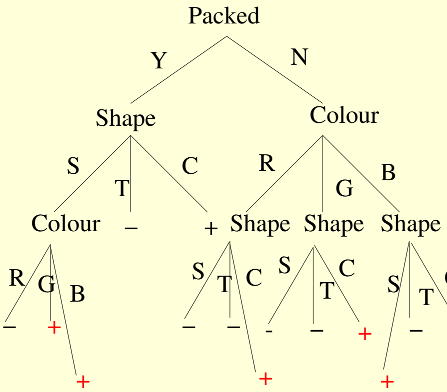


Decision Tree 1

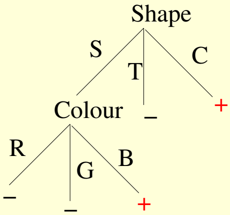


Decision Tree 2

Decision Trees



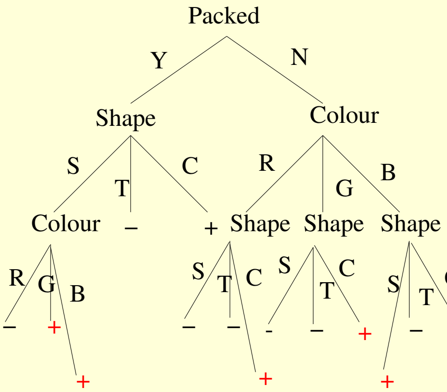
Decision Tree 1



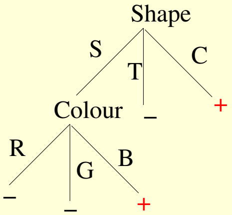
Decision Tree 2

- How many possible trees can we draw to represent our data set?

Decision Trees



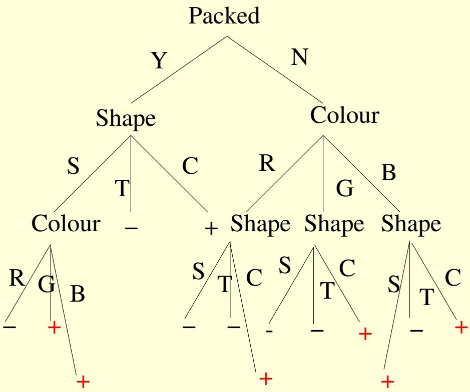
Decision Tree 1



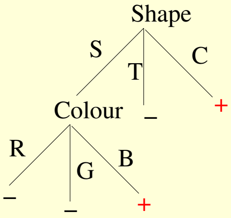
Decision Tree 2

- How many possible trees can we draw to represent our data set? A lot!—more than exponential in the number of features.

Decision Trees



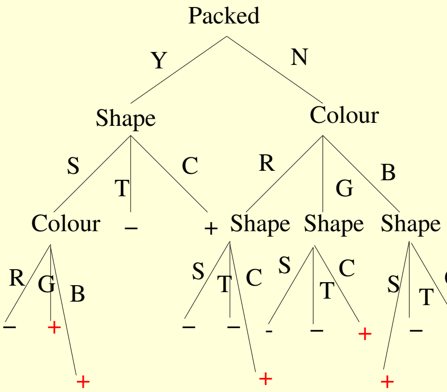
Decision Tree 1



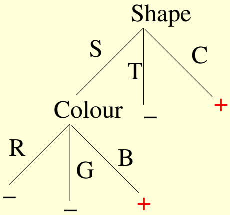
Decision Tree 2

- How many possible trees can we draw to represent our data set?
  A lot!—more than exponential in the number of features.
- Which tree among 1 and 2 is preferable? Why?

Decision Trees



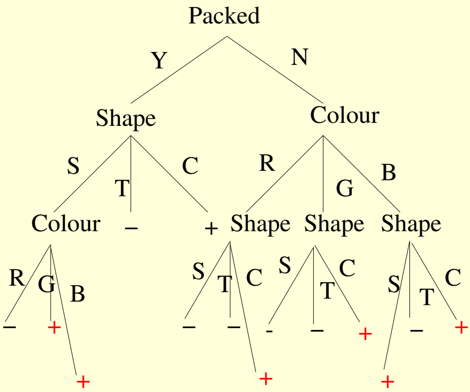
Decision Tree 1



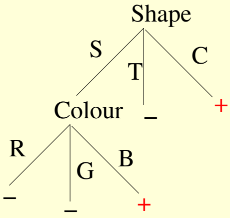
Decision Tree 2

- How many possible trees can we draw to represent our data set?
  A lot!—more than exponential in the number of features.
- Which tree among 1 and 2 is preferable? Why?
  A smaller tree is likely to generalise better.

Decision Trees



Decision Tree 1



Decision Tree 2

- How many possible trees can we draw to represent our data set?
  A lot!—more than exponential in the number of features.
- Which tree among 1 and 2 is preferable? Why?
  A smaller tree is likely to generalise better.
  Occam's Razor: a simpler solution is usually better.

Learning (Compact) Decision Trees

1. If all data have the same class C, create a leaf with prediction C. Return.
2. Identify a feature f that divides data into “homogeneous” (“pure”) sets.
3. Create a node with a split based on f; divide training data based on f and move it into corresponding branches.
4. Grow a sub-tree for each branch (recursively).

Learning (Compact) Decision Trees

1. If all data have the same class C, create a leaf with prediction C. Return.
2. Identify a feature f that divides data into “homogeneous” (“pure”) sets.
3. Create a node with a split based on f; divide training data based on f and move it into corresponding branches.
4. Grow a sub-tree for each branch (recursively).

- Many implementations, variations available.

Learning (Compact) Decision Trees

1. If all data have the same class C, create a leaf with prediction C. Return.
2. Identify a feature f that divides data into "homogeneous" ("pure") sets.
3. Create a node with a split based on f; divide training data based on f and move it into corresponding branches.
4. Grow a sub-tree for each branch (recursively).

- Many implementations, variations available.
- Procedure described above is greedy; won't necessarily find the most compact decision tree.

Learning (Compact) Decision Trees

1. If all data have the same class C, create a leaf with prediction C. Return.
2. Identify a feature f that divides data into "homogeneous" ("pure") sets.
3. Create a node with a split based on f; divide training data based on f and move it into corresponding branches.
4. Grow a sub-tree for each branch (recursively).

- Many implementations, variations available.
- Procedure described above is greedy; won't necessarily find the most compact decision tree.
- What are some advantages of decision trees?

Learning (Compact) Decision Trees

1. If all data have the same class C, create a leaf with prediction C. Return.
2. Identify a feature f that divides data into "homogeneous" ("pure") sets.
3. Create a node with a split based on f; divide training data based on f and move it into corresponding branches.
4. Grow a sub-tree for each branch (recursively).

- Many implementations, variations available.
- Procedure described above is greedy; won't necessarily find the most compact decision tree.
- What are some advantages of decision trees?

  Often human-readable/interpretable.

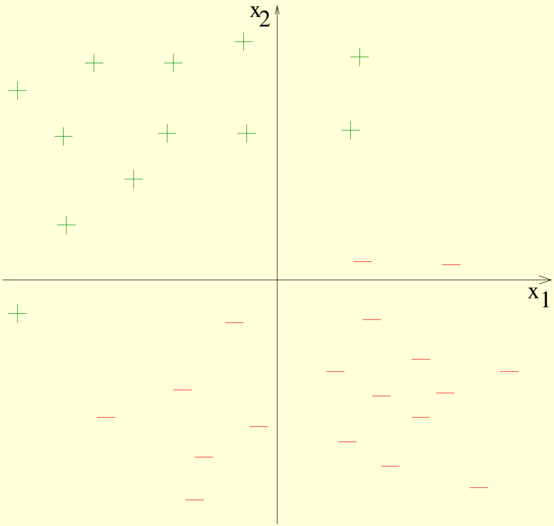
  Very fast to train (parallelisable).

  Suited to work with categorical features.

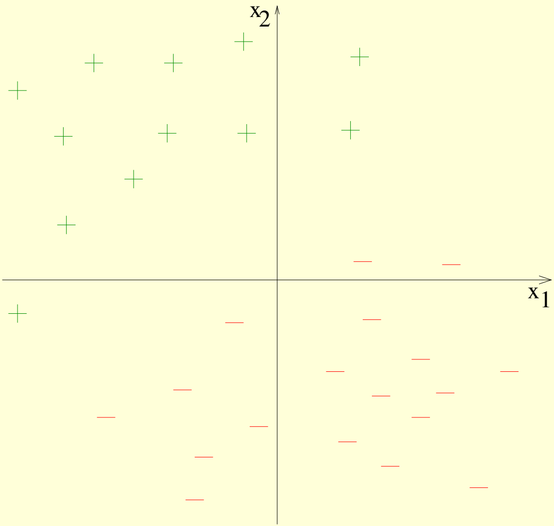
This Lecture

- Evaluation metrics for classification
- Decision trees
- Support Vector Machines
- Nearest-neighbour methods
- Supervised learning: summary

Maximum-margin Hyperplane

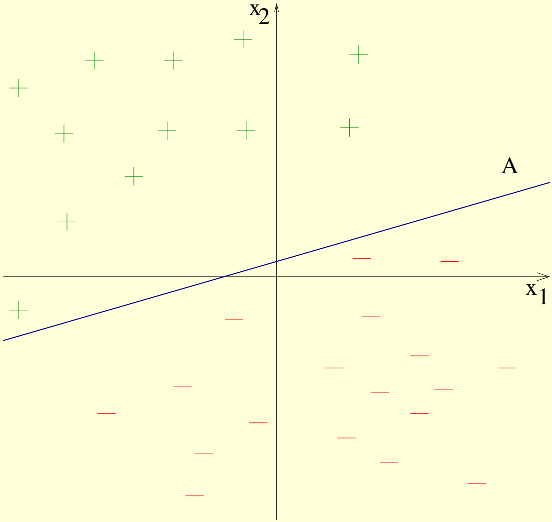


Maximum-margin Hyperplane



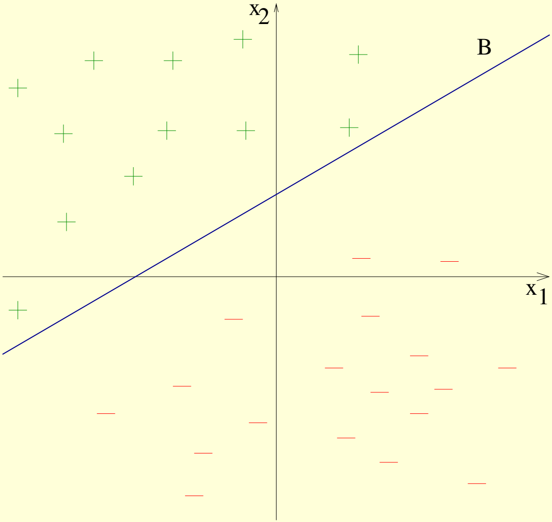
Which line among A, B, C, D is the most preferable?

Maximum-margin Hyperplane



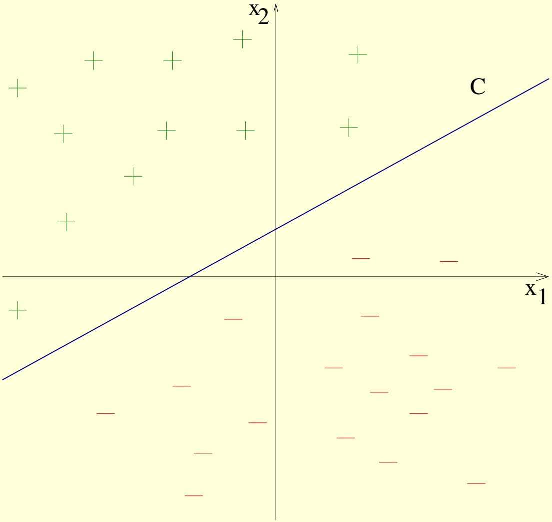
Which line among A, B, C, D is the most preferable?

Maximum-margin Hyperplane



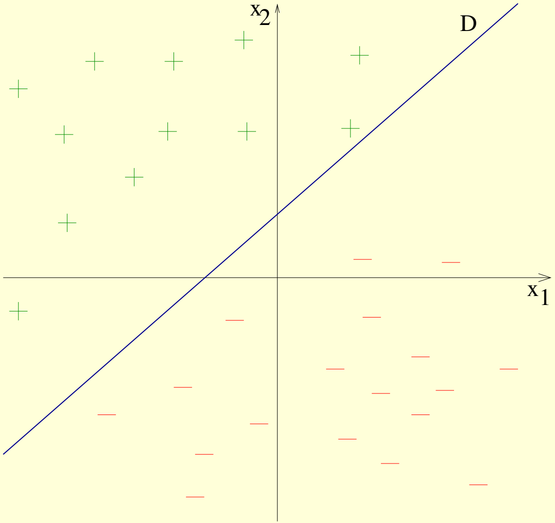
Which line among A, B, C, D is the most preferable?

Maximum-margin Hyperplane



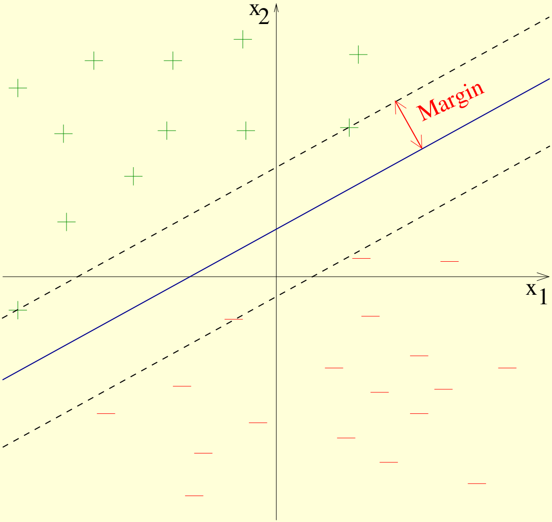
Which line among A, B, C, D is the most preferable?

Maximum-margin Hyperplane



Which line among A, B, C, D is the most preferable?

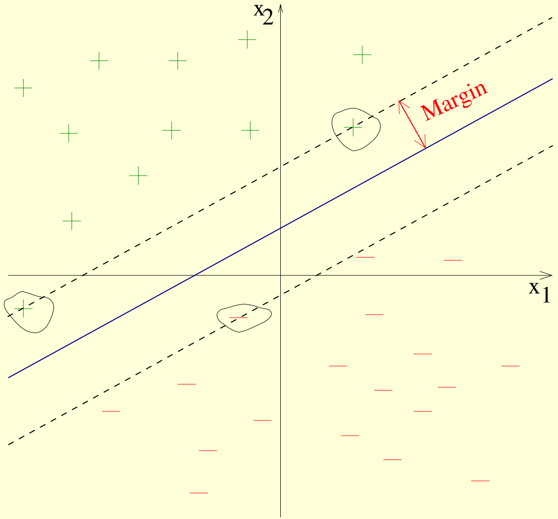
Maximum-margin Hyperplane



Which line among A, B, C, D is the most preferable?

C is the maximum-margin line.

Maximum-margin Hyperplane



Which line among A, B, C, D is the most preferable?

C is the maximum-margin line.

The circled points are support vectors; the model is a Support Vector Machine.

Learning Support Vector Machines

- The maximum-margin hyperplane can be obtained by solving a quadratic program.

Learning Support Vector Machines

- The maximum-margin hyperplane can be obtained by solving a quadratic program.
- The problem is convex and efficient to solve.

Learning Support Vector Machines

- The maximum-margin hyperplane can be obtained by solving a quadratic program.
- The problem is convex and efficient to solve.
- The formulation can be tweaked to accommodate small violations of linear separability.

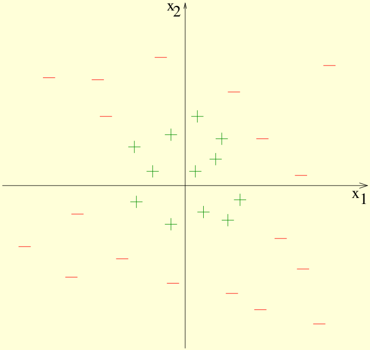
Learning Support Vector Machines

- The maximum-margin hyperplane can be obtained by solving a quadratic program.
- The problem is convex and efficient to solve.
- The formulation can be tweaked to accommodate small violations of linear separability.
- SVMs were very popular in the 2000's.

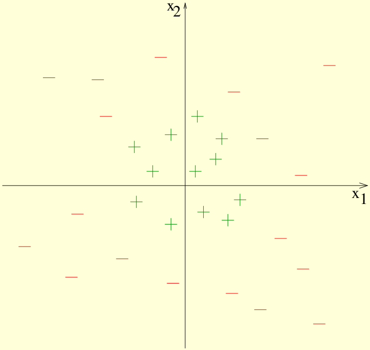
Learning Support Vector Machines

- The maximum-margin hyperplane can be obtained by solving a quadratic program.
- The problem is convex and efficient to solve.
- The formulation can be tweaked to accommodate small violations of linear separability.
- SVMs were very popular in the 2000's.
- Why all the fuss? The model is still linear, isn't it?

Kernels

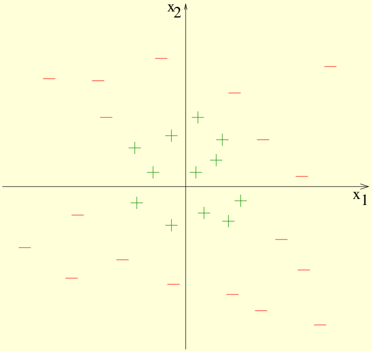


Kernels



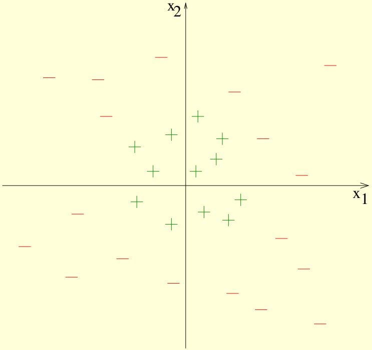
- Can we add a feature x_3 to make this data linearly separable?

Kernels



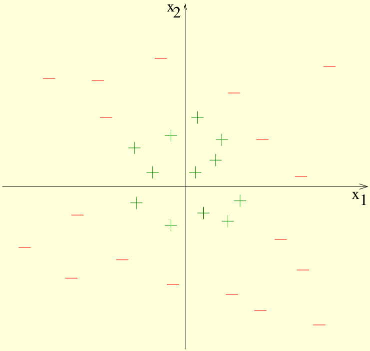
- Can we add a feature x_3 to make this data linearly separable?
  Yes: set $x_3 = (x_1)^2 + (x_2)^2$. Predict $+$ if $x_3 - C \leq 0$ and $-$ otherwise.

Kernels



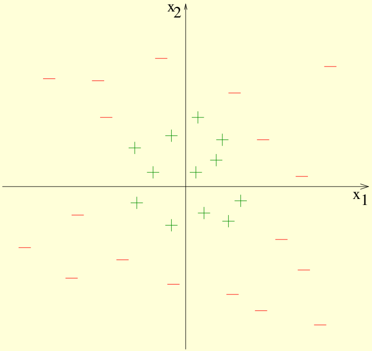
- Can we add a feature x_3 to make this data linearly separable?
  Yes: set $x_3 = (x_1)^2 + (x_2)^2$. Predict $+$ if $x_3 - C \leq 0$ and $-$ otherwise.
- Kernels provide a convenient mechanism to transform points in input space X to a higher dimensional space X', hoping that the points will be separable in X'.

Kernels



- Can we add a feature x_3 to make this data linearly separable?
  Yes: set $x_3 = (x_1)^2 + (x_2)^2$. Predict $+$ if $x_3 - C \leq 0$ and $-$ otherwise.
- Kernels provide a convenient mechanism to transform points in input space X to a higher dimensional space X', hoping that the points will be separable in X'.
- Kernels do not entail much computational overhead.

Kernels

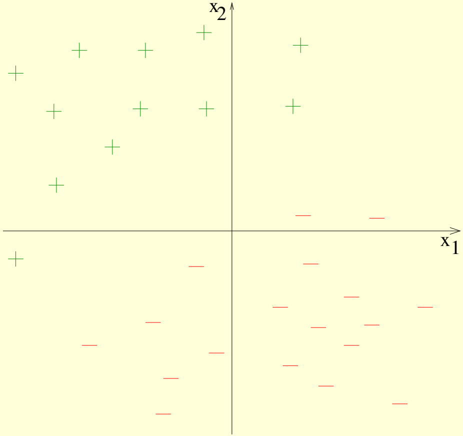


- Can we add a feature x_3 to make this data linearly separable?
  Yes: set $x_3 = (x_1)^2 + (x_2)^2$. Predict $+$ if $x_3 - C \leq 0$ and $-$ otherwise.
- Kernels provide a convenient mechanism to transform points in input space X to a higher dimensional space X', hoping that the points will be separable in X'.
- Kernels do not entail much computational overhead.
- While there is no guarantee of linear separability in X', kernelised SVMs have registered many empirical successes.

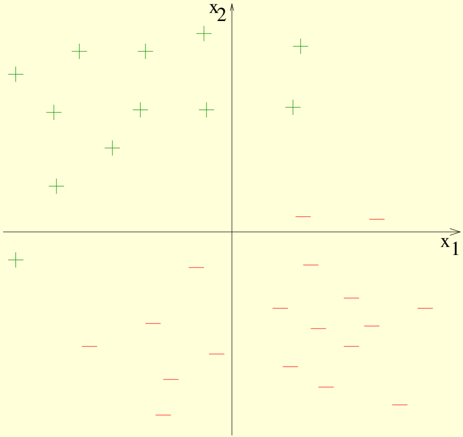
This Lecture

- Evaluation metrics for classification
- Decision trees
- Support Vector Machines
- Nearest-neighbour methods
- Supervised learning: summary

k-Nearest Neighbour (kNN)

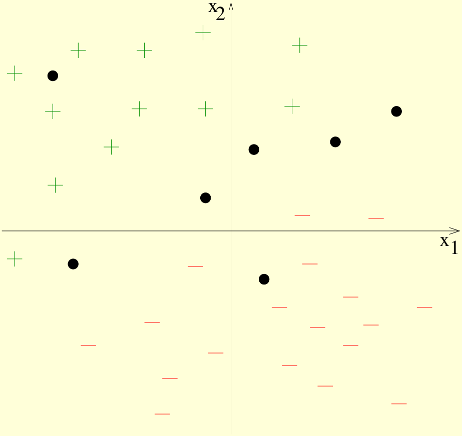


k-Nearest Neighbour (kNN)



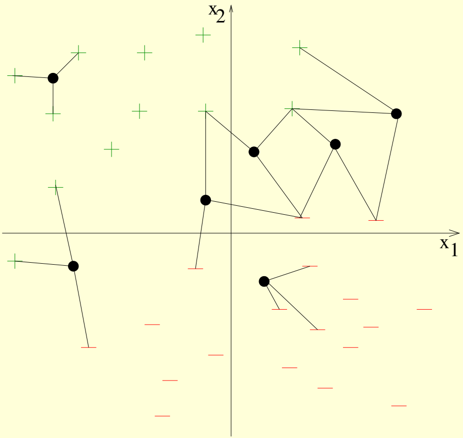
- Given query point x, predict as its label the majority label of x's k nearest neighbours in the training data.

k-Nearest Neighbour (kNN)



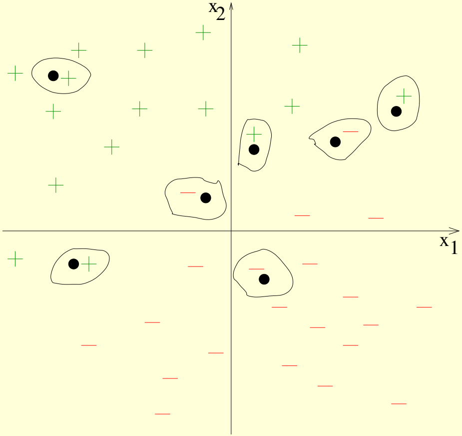
- Given query point x, predict as its label the majority label of x's k nearest neighbours in the training data.
- Illustration with $k = 3$.

k-Nearest Neighbour (kNN)



- Given query point x, predict as its label the majority label of x's k nearest neighbours in the training data.
- Illustration with $k = 3$.

k-Nearest Neighbour (kNN)



- Given query point x, predict as its label the majority label of x's k nearest neighbours in the training data.
- Illustration with $k = 3$.

Nearest Neighbour Methods

- What did we learn here? What is the model?

Nearest Neighbour Methods

- What did we learn here? What is the model?
  The model is the data set itself! There is nothing to "learn".

Nearest Neighbour Methods

- What did we learn here? What is the model?
  The model is the data set itself! There is nothing to "learn".
  However, sometimes only a subset of the training data is stored, so the "model" size is reduced.

Nearest Neighbour Methods

- What did we learn here? What is the model?
  The model is the data set itself! There is nothing to “learn”.
  However, sometimes only a subset of the training data is stored, so the “model” size is reduced.

- What are the main advantages and disadvantages of this approach?

Nearest Neighbour Methods

- What did we learn here? What is the model?
  The model is the data set itself! There is nothing to “learn”.
  However, sometimes only a subset of the training data is stored, so the “model” size is reduced.

- What are the main advantages and disadvantages of this approach?
  Simple, but only tends to work in relatively low dimensions. In high dimensions all neighbours are far away!

Nearest Neighbour Methods

- What did we learn here? What is the model?
  The model is the data set itself! There is nothing to "learn".
  However, sometimes only a subset of the training data is stored, so the "model" size is reduced.

- What are the main advantages and disadvantages of this approach?
  Simple, but only tends to work in relatively low dimensions. In high dimensions all neighbours are far away!
  Query time (finding k nearest neighbours) is high (linear in the model size), although finding k "approximate" nearest neighbours can be done quickly.

Nearest Neighbour Methods

- What did we learn here? What is the model?
  The model is the data set itself! There is nothing to “learn”.
  However, sometimes only a subset of the training data is stored, so the “model” size is reduced.

- What are the main advantages and disadvantages of this approach?
  Simple, but only tends to work in relatively low dimensions. In high dimensions all neighbours are far away!
  Query time (finding k nearest neighbours) is high (linear in the model size), although finding k “approximate” nearest neighbours can be done quickly.

- What is the effect of the parameter k?

Nearest Neighbour Methods

- What did we learn here? What is the model?
  The model is the data set itself! There is nothing to "learn".
  However, sometimes only a subset of the training data is stored, so the "model" size is reduced.

- What are the main advantages and disadvantages of this approach?
  Simple, but only tends to work in relatively low dimensions. In high dimensions all neighbours are far away!
  Query time (finding k nearest neighbours) is high (linear in the model size), although finding k "approximate" nearest neighbours can be done quickly.

- What is the effect of the parameter k?
  Small $k \implies$ complex model; large $k \implies$ simple model.

Nearest Neighbour Methods

- What did we learn here? What is the model?
  The model is the data set itself! There is nothing to "learn".
  However, sometimes only a subset of the training data is stored, so the "model" size is reduced.

- What are the main advantages and disadvantages of this approach?
  Simple, but only tends to work in relatively low dimensions. In high dimensions all neighbours are far away!
  Query time (finding k nearest neighbours) is high (linear in the model size), although finding k "approximate" nearest neighbours can be done quickly.

- What is the effect of the parameter k?
  Small $k \implies$ complex model; large $k \implies$ simple model.

- How to find the majority label if k is even?

Nearest Neighbour Methods

- What did we learn here? What is the model?
  The model is the data set itself! There is nothing to "learn".
  However, sometimes only a subset of the training data is stored, so the "model" size is reduced.

- What are the main advantages and disadvantages of this approach?
  Simple, but only tends to work in relatively low dimensions. In high dimensions all neighbours are far away!
  Query time (finding k nearest neighbours) is high (linear in the model size), although finding k "approximate" nearest neighbours can be done quickly.

- What is the effect of the parameter k?
  Small $k \implies$ complex model; large $k \implies$ simple model.

- How to find the majority label if k is even?
  If there are equal $+$'s and $-$'s among the neighbours, pick a label uniformly at random (that is, even tie-breaking).

Nearest Neighbour Methods

- What did we learn here? What is the model?
  The model is the data set itself! There is nothing to "learn".
  However, sometimes only a subset of the training data is stored, so the "model" size is reduced.

- What are the main advantages and disadvantages of this approach?
  Simple, but only tends to work in relatively low dimensions. In high dimensions all neighbours are far away!
  Query time (finding k nearest neighbours) is high (linear in the model size), although finding k "approximate" nearest neighbours can be done quickly.

- What is the effect of the parameter k?
  Small $k \implies$ complex model; large $k \implies$ simple model.

- How to find the majority label if k is even?
  If there are equal $+$'s and $-$'s among the neighbours, pick a label uniformly at random (that is, even tie-breaking).

- Can this approach be used for regression?

Nearest Neighbour Methods

- What did we learn here? What is the model?
  The model is the data set itself! There is nothing to "learn".
  However, sometimes only a subset of the training data is stored, so the "model" size is reduced.

- What are the main advantages and disadvantages of this approach?
  Simple, but only tends to work in relatively low dimensions. In high dimensions all neighbours are far away!
  Query time (finding k nearest neighbours) is high (linear in the model size), although finding k "approximate" nearest neighbours can be done quickly.

- What is the effect of the parameter k?
  Small $k \implies$ complex model; large $k \implies$ simple model.

- How to find the majority label if k is even?
  If there are equal $+$'s and $-$'s among the neighbours, pick a label uniformly at random (that is, even tie-breaking).

- Can this approach be used for regression?
  Of course. Predict the average value of the k nearest neighbours. (Sometimes each neighbour is given a weight inversely proportional to its distance.)

This Lecture

- Evaluation metrics for classification
- Decision trees
- Support Vector Machines
- Nearest-neighbour methods
- Supervised learning: summary

Summary of Supervised Learning

- Given labeled training data $D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)\}$, to learn a model M, which, given a new point x, can make a good prediction $M(x)$.

Summary of Supervised Learning

- Given labeled training data $D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)\}$, to learn a model M, which, given a new point x, can make a good prediction $M(x)$.
- Most well-known problem in machine learning. Large number of applications across different domains.

Summary of Supervised Learning

- Given labeled training data $D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)\}$, to learn a model M, which, given a new point x, can make a good prediction $M(x)$.
- Most well-known problem in machine learning. Large number of applications across different domains.
- Linear models are easy to learn, but their performance depends heavily on the features used.

Summary of Supervised Learning

- Given labeled training data $D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)\}$, to learn a model M, which, given a new point x, can make a good prediction $M(x)$.
- Most well-known problem in machine learning. Large number of applications across different domains.
- Linear models are easy to learn, but their performance depends heavily on the features used.
- Most methods will have parameters determining the model complexity (regularisation coefficient in linear models, nodes/layers in a neural network, depth of a decision tree, etc.)

Summary of Supervised Learning

- Given labeled training data $D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)\}$, to learn a model M, which, given a new point x, can make a good prediction $M(x)$.
- Most well-known problem in machine learning. Large number of applications across different domains.
- Linear models are easy to learn, but their performance depends heavily on the features used.
- Most methods will have parameters determining the model complexity (regularisation coefficient in linear models, nodes/layers in a neural network, depth of a decision tree, etc.)
- Model complexity can be tuned using cross-validation—to guard from overfitting as well as underfitting.

Summary of Supervised Learning

- Given labeled training data $D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)\}$, to learn a model M, which, given a new point x, can make a good prediction $M(x)$.
- Most well-known problem in machine learning. Large number of applications across different domains.
- Linear models are easy to learn, but their performance depends heavily on the features used.
- Most methods will have parameters determining the model complexity (regularisation coefficient in linear models, nodes/layers in a neural network, depth of a decision tree, etc.)
- Model complexity can be tuned using cross-validation—to guard from overfitting as well as underfitting.
- Non-linear models usually have less analytical justification, but find extensive empirical validation.

Summary of Supervised Learning

- Given labeled training data $D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)\}$, to learn a model M, which, given a new point x, can make a good prediction $M(x)$.
- Most well-known problem in machine learning. Large number of applications across different domains.
- Linear models are easy to learn, but their performance depends heavily on the features used.
- Most methods will have parameters determining the model complexity (regularisation coefficient in linear models, nodes/layers in a neural network, depth of a decision tree, etc.)
- Model complexity can be tuned using cross-validation—to guard from overfitting as well as underfitting.
- Non-linear models usually have less analytical justification, but find extensive empirical validation.
- Deep neural networks have revolutionised AI/ML by their performance on vision, speech, NLP applications. Other methods might be better-suited to other applications.

Summary of Supervised Learning

- Given labeled training data $D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)\}$, to learn a model M, which, given a new point x, can make a good prediction $M(x)$.
- Most well-known problem in machine learning. Large number of applications across different domains.
- Linear models are easy to learn, but their performance depends heavily on the features used.
- Most methods will have parameters determining the model complexity (regularisation coefficient in linear models, nodes/layers in a neural network, depth of a decision tree, etc.)
- Model complexity can be tuned using cross-validation—to guard from overfitting as well as underfitting.
- Non-linear models usually have less analytical justification, but find extensive empirical validation.
- Deep neural networks have revolutionised AI/ML by their performance on vision, speech, NLP applications. Other methods might be better-suited to other applications.
- There are many off-the-shelf implementations available for most supervised learning methods.

Summary of Supervised Learning

- Given labeled training data $D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)\}$, to learn a model M, which, given a new point x, can make a good prediction $M(x)$.
- Most well-known problem in machine learning. Large number of applications across different domains.
- Linear models are easy to learn, but their performance depends heavily on the features used.
- Most methods will have parameters determining the model complexity (regularisation coefficient in linear models, nodes/layers in a neural network, depth of a decision tree, etc.)
- Model complexity can be tuned using cross-validation—to guard from overfitting as well as underfitting.
- Non-linear models usually have less analytical justification, but find extensive empirical validation.
- Deep neural networks have revolutionised AI/ML by their performance on vision, speech, NLP applications. Other methods might be better-suited to other applications.
- There are many off-the-shelf implementations available for most supervised learning methods.
- Topics we did not cover: dimensionality reduction, logistic regression, ensemble methods, error analysis, ...

References

- Wikipedia page on confusion matrix:
  https://en.wikipedia.org/wiki/Confusion_matrix.

- Chapter 1, sections 3.2, 3.3, 7.7, **A Course in Machine Learning**, Hal Daumé III. Available on-line at http://ciml.info/.