Tutorial 9 (along with ~~some~~ solutions)

Sunday 13th November, 2016

1 Dual of the Support Vector Classifier

Taking hints from the Support Vector Regression Problem, derive the Dual for SVM Primal Classification Problem stated below:

- $(w^*, b^*, \xi_i^*) = \underset{w,b,\xi_i}{argmin} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$
  s.t. $y_i(w^\top \phi(x_i) + b) \geq 1 - \xi_i$ and
  $\xi_i \geq 0$, $\forall i = 1, \ldots, n$
  - Recall that we derived this form by minimizing $\frac{1}{2}\|w\|^2$ instead of maximizing $\frac{2}{\|w\|}$ ($\frac{1}{2}\|w\|^2$ is monotonically decreasing with respect to $\frac{2}{\|w\|}$)
  - C determines the trade-off between the error $\sum \xi_i$ and the margin $\frac{2}{\|w\|}$

1.1 ANSWER

- Let $L^*(\alpha, \mu) = \min_{w,b,\xi} \underline{L(w, b, \xi, \alpha, \mu)}$ → First construct Lagrange fn.
- By weak duality theorem, we have:
  $L^*(\alpha, \mu) \leq \min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$
  s.t. $y_i(w^\top \phi(x_i) + b) \geq 1 - \xi_i$, and
  $\xi_i \geq 0$, $\forall i = 1, \ldots, n$
- The above is true for any $\alpha_i \geq 0$ and $\mu_i \geq 0$
- Thus,

$$\max_{\alpha,\mu} L^*(\alpha, \mu) \leq \min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

1.2 Dual objective

- In case of SVM, we have a strictly convex objective and linear constraints – therefore, strong duality holds:

$$\max_{\alpha,\mu} L^*(\alpha, \mu) = \min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

- This value is precisely obtained at the $(w^*, b^*, \xi^*, \alpha^*, \mu^*)$ that satisfies the necessary (and sufficient) optimality conditions
- Assuming that the necessary and sufficient conditions (KKT or Karush–Kuhn–Tucker conditions) hold, our objective becomes:

$$\max_{\alpha,\mu} L^*(\alpha, \mu)$$

- $L(w, b, \xi, \alpha, \mu) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^n \xi_i + \sum_{i=1}^n \alpha_i(1 - \xi_i - y_i(w^\top\phi(x_i) + b)) - \sum_{i=1}^n \mu_i\xi_i$
- We obtain w, b, ξ in terms of α and μ by setting $\nabla_{w,b,\xi}L = 0$:
  - **w.r.t.** w**:** $w = \sum_{i=1}^n \alpha_i y_i \phi(x_i)$
  - **w.r.t.** b**:** $-b\sum_{i=1}^n \alpha_i y_i = 0$
  - **w.r.t.** ξ_i**:** $\alpha_i + \mu_i = C$

very similar to SVR. but fewer variables & constraints

- Thus, we get:
  $L(w, b, \xi, \alpha, \mu)$
  $= \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \phi^\top(x_i)\phi(x_j) + C\sum_i \xi_i + \sum_i \alpha_i - \sum_i \alpha_i\xi_i - \sum_i \alpha_i y_i \sum_j \alpha_j y_j \phi^\top(x_j)\phi(x_i) - b\sum_i \alpha_i y_i - \sum_i \mu_i\xi_i$
  $= \underline{-\frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \underbrace{\phi^\top(x_i)\phi(x_j)}_{K(x_i, x_j)} + \sum_i \alpha_i}$

1.3 The dual optimization problem becomes

$f(x) = \sum_i \alpha_i K(x, x_i) + \underset{\sim}{b}$

- $$\max_\alpha -\frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \underset{\sim}{\phi^\top(x_i)\phi(x_j)} + \sum_i \alpha_i$$

  s.t. → by complementary slackness
  $\alpha_i \in [0, C]$, $\forall i$ and
  $\sum_i \alpha_i y_i = 0$ → $\nabla_b = 0$

  Other constraints NOT involving α_i's ignored

- Deriving this did not require the complementary slackness conditions

  Refers to objective & NOT to constraints

- Conveniently, we also end up getting rid of $\underline{\mu}$

2 Transfer Learning

(eg: Transferring face detection CNN features for videos from one classroom to another)

In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization

Replace
typically numeric features → last layer
(maps)
CNN

or a fixed feature extractor for the task of interest. **Reflect on the following two major Transfer Learning[1] scenarios**:

1. ConvNet as fixed feature extractor. Take a ConvNet pretrained on ImageNet, remove the last fully-connected layer (this layers outputs are the 1000 class scores for a different task like ImageNet), then treat the rest of the ConvNet as a fixed feature extractor for the new dataset. In an AlexNet, this would compute a 4096-D vector for every image that contains the activations of the hidden layer immediately before the classifier. We call these features CNN codes. It is important for performance that these codes are ReLUd (i.e. thresholded at zero) if they were also thresholded during the training of the ConvNet on ImageNet (as is usually the case). Once you extract the 4096-D codes for all images, train a linear classifier (e.g. Linear SVM or Softmax classifier) or a nonlinear classifier (e.g. Decision tree) for the new dataset.

2. Fine-tuning the ConvNet. The second strategy is to not only replace and retrain the classifier on top of the ConvNet on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation. It is possible to fine-tune all the layers of the ConvNet, or it is possible to keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion of the network. This is motivated by the observation that the earlier features of a ConvNet contain more generic features (e.g. edge detectors or color blob detectors) that should be useful to many tasks, but later layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset. In case of ImageNet for example, which contains many dog breeds, a significant portion of the representational power of the ConvNet may be devoted to features that are specific to differentiating between dog breeds.

Also **explain how you could do effective use of CNN-based feature maps in decision tree learning.**

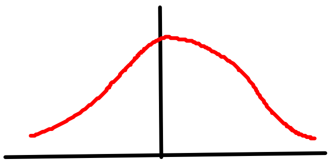
Solution: Most of the non-triviality will be about how to employ decision tree learning on numeric features.

One solution is to create a discrete attribute to test each continuous valued attribute. In the case of transfer learning with CNN, a feature (or heat[2]) map from CNN might have a continuous attribute, indicative of the any of the following numeric indices at `http://places.csail.mit.edu/demo.html` such as naturallight, openarea, ruggedscene, climbing, rockstone, directsunsunny, dry, vacationingtouring, natural, warm. We will illustrate a simple procedure to deal with numeric attributes with a very simple (non-CNN) problem - that of detecting whether or not a child is malnourished.

Let us say the continuous valued attribute was (Child) $Weight$ which assumes the values $\{15, 16, 19, 22, 27, 29\}$ in the training set.

- $Weight \in \{15, 16, 19, 22, 27, 29\}$ kg in the training data.
- $(Weight \geq 22) = f$

To improve efficiency, instead of considering threshold at every numeric value of the attribute in training data, you could look at pdf estimation (using, say Gaussian) & consider fewer thresholds



[1] Mostly from `http://cs231n.github.io/transfer-learning/`
[2] `https://en.wikipedia.org/wiki/Heat_map`

Weight:	15	16	19	22	27	29
Malnourished:	Yes	No	Yes	No	Yes	No

Age must be an additional attribute]

3 Decision Trees and Feature Selection

- We discussed the information gain criterion for feature splitting in Decision trees. Suggest two other criteria. Motivate each.

3.1 ANSWER

Solution[3]: 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:

$$\underset{V(\phi_i),\phi_i}{\operatorname{argmax}} \left(Imp(S) - \sum_{v_{ij} \in V(\phi_i)} \frac{|S_{v_{ij}}|}{|S|} Imp(S_{v_{ij}}) \right)$$

where $S_{ij} \subseteq \mathcal{D}$ is a subset of dataset such that each instance x has attribute value $\phi_i(x) = v_{ij}$.

1. An example choice of $Imp(S)$ discussed in the class notes is the entropy[4] $Imp(S) = H(S) = -\sum_{i=1}^{K} Pr(C_i) \bullet \log(Pr(C_i))$.

2. Alternative impurity measures are shown in Table 1 and they all measure the extent of spread of the probabilities over the classes. In other words, as shown in Figure 1 each impurity measure indicates the extent to which the data is "confused" about the classes.

Name	Imp (D)
Entropy	$-\sum_{i=1}^{K} Pr(C_i) \bullet \log(Pr(C_i))$
Gini Index	$\sum_{i=1}^{K} Pr(C_i)(1 - Pr(C_i))$
Class (Min Prob) Error	$\underset{i}{\operatorname{argmin}}(1 - Pr(C_i))$

Table 1: Decision Tree: Impurity measurues

[3] Section 5.1 of https://www.cse.iitb.ac.in/~cs725/notes/classNotes/extra_lecturenotes_cs725.pdf

[4] See slide 5 of http://23.253.82.180/course/307/858/2217

super linear

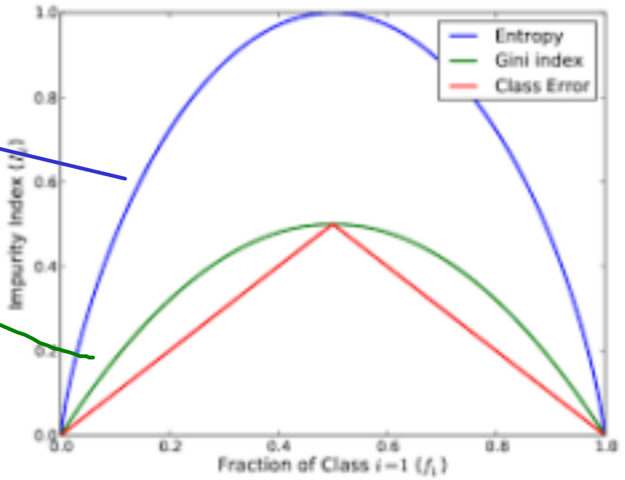


Figure 1: Plot of Entropy, Gini Index and Misclassification Accuracy. Source: https://inspirehep.net/record/1225852/files/TPZ_Figures_impurity.png

Favoured by default by impurity reduction

ϕ_j 10 values

ϕ_i 4 values

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 $\phi_i < 1,\ 1 \le \phi_i < 3,\ 3 \le \phi_i < 5,\ \phi_i \ge 5$

3.1.1 An empirical observation

Since smaller range of attribute values in 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. Recall that in overfitting, the system learns a model which is specific to the training collection and does not generalize well on unseen data.

3.1.2 Need to address this empirical observation

This could be achieved by the maximization of the following expression:

$$Imp_{skew}(S) = Imp(S) \; - \left(\frac{\sum_{v_{ij} \in V(\phi_i)} \frac{|S_v|}{|S|} Imp(S_{v_{ij}})}{-\sum_{v \in V(\phi_i)} \frac{|S_v|}{|S|} \log(S_v)} \right)$$

Rominiscent of IDF of TFIDF. but here you divide by "log"

The second term in the above expression is called $\triangle Imp(S)$. The intuition for using this term is that, more the skew, lower will be the denominator and is therefore better at countering lowered impurity. In other words, this new measure prefers a less skewed tree as shown in Figure 2. We will refer to the above modified measure $Imp_{skew}(S)$ as the Skew Adjusted Information Gain.

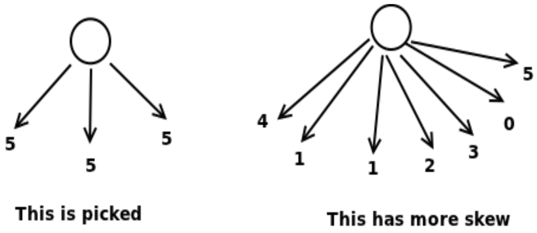


Figure 2: Skewness and empirical observation

3.1.3 Summing it all up

Below we present the overall decision tree learning algorithm with stopping criteria as described above. Ideally, this algorithm goes on until all the data points at the leaf are of the same single class and the two stopping criterion added to the algorithm make it terminate even if such a condition does not occur.

```
Algorithm 1 T = dtree $(S, \phi_i, V)$
  if $\phi$ = empty then
    return a tree with only one branch $C_j$, where $C_j$ is the majority class in $S$ {Stopping criterion}
  end if
  if all instances in S have label = $c_i$ then
    return a tree with only one branch $c_i$ {Stopping criterion}
  end if
  $\phi_j = \text{argmax}_{V(\phi_i),\phi_i} \left(\triangle Imp(S)\right) \quad \forall v \in V(\phi_i)$
  $T_v = dtree(S_v, \phi - \phi_i, V)$
  return a tree rooted at $\phi_i$ and having all the $T_v$ branches
```

- We also discussed how the information gain criterion can be used for feature selection in general. Suggest two other criteria. You can build on your answer to the question above.

3.2 ANSWER

The answer is similar to the one above. All the three impurity functions, *viz.*, entropy, gini-index and misclassification error could be used for feature selection. Moreover, the Skew Adjusted Information Gain measure $Imp_{skew}(S)$ could also be used for feature selection.

- **OPTIONAL:** Suggest how you could used hypothesis testing for pruning or stopping decision tree construction.

3.3 ANSWER

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.

3.3.1 Alternatives in Pruning

There are various strategies/heuristics to decrease the complexity of the tree learnt. Some of the options are as follows:

Discussed in class

1. Early termination. Stop if $\triangle Imp(S) < \theta$, 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 Hypothesis testing procedures could be used to achieve this. (For instance, the binomial and χ^2-tests).

   Explicitly prune

   Evaluate / test significance of every node as an afterthought
4. Use an objective function like

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

   The complexity is characterized by the description length principle (MDL)

Please note that the section that follows is a completely optional reading. However, it could enhance your understanding significantly.

3.3.2 (OPTIONAL) Hypothesis Testing for Decision Tree Pruning

The 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 3. The numbers n_1, n_2 etc. indicate the number of instances of the particular class.

If the class ratios of instances remain similar to that before the split, then we might not gain much by splitting that node. To quantify this, we employ Hypothesis testing. The idea is to compare 2 probability distributions.

We will illustrate with a 2-class classification problem. If p is the probability of taking the left branch, the probability of taking the right branch is $1 - p$. Then we obtain the following:

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

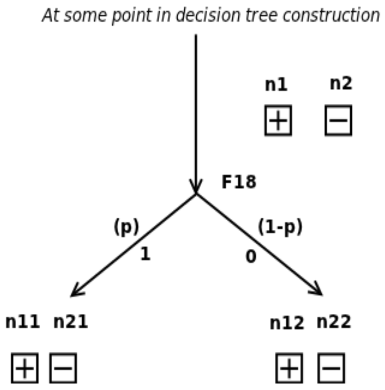


Figure 3: Splitting criterion

significance tests built around n_{11}, n_{21}, n_{12} & n_{22}

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

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

Why? Because these splits only add to the complexity of the tree and do not convey any meaningful information not already present. Suppose we are interested not only in equal distributions but also in 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 employed.

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

$$p(C_i) = \mu_i = \frac{n_i}{\sum_{i=1}^{K} n_i} \qquad \text{for a given class } i \text{ (pre-splitting distribution)}$$

3.3.3 Hypothesis testing: problem

Note: The text presented here in red refers specifically to the decision tree problem. The text in black is for the general Hypothesis testing problem and can be read up in Section 7 of https://www.cse.iitb.ac.in/~cs725/notes/classNotes/extra_lecturenotes_cs725_aut11.pdf

Let $X_1, \ldots X_n$ be i.i.d random samples that correspond to class labels of instances that have gone into the left branch.

The null hypothesis is H_0 and the alternative hypothesis is H_1:

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

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

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

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,

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

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

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

3.3.4 Goodness-of-fit test for DTrees

Consider the test statistic $S_i = \sum_{j=1}^{n} \delta(X_j, C_i)$.

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

$$\begin{aligned} \mathcal{E}_{H_0}[Y_i] &= \sum_{j=1}^{n} \mathcal{E}_{H_0}[\delta(X_j, C_i)] \\ &= \sum_{j=1}^{n} \mu_i * 1 + (1 - \mu_i) * 0 \\ &= n\mu_i \end{aligned}$$

$$\begin{aligned} C &= \left\{ (X_1, \ldots X_n) \middle| \sum_{i=1}^{K} \frac{(Y_i - \mathcal{E}_{H_0}(Y_i))^2}{\mathcal{E}_{H_0}(Y_i)} \leq c \right\} \qquad \text{where } c \text{ is some constant} \\ &= \left\{ (X_1, \ldots X_n) \middle| \sum_{i=1}^{K} \frac{(Y_i - n\mu_i)^2}{n\mu_i} \leq c \right\} \end{aligned}$$

As we might have seen in a basic course on statistics[5], the above expression $\sim \chi^2_{K-1}$. We then use the chi-square tables to find c given the value of α.

[5] Section 7 of https://www.cse.iitb.ac.in/~cs725/notes/classNotes/extra_lecturenotes_cs725.pdf

3.3.5 Final Heuristic used for DTree construction

- Compute $\sum_{i=1}^{K} \frac{(Y_i - n\mu_i)^2}{n\mu_i} \sim t_s \quad \forall\, splits$, 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 $Pr_{\chi^2_{K-1}}(x \geq c_\alpha) = \alpha$

Egs misclassified by confident models considered more imp subsequently

4 Adaboost (Advanced and Optional)

Recall the main idea behind Adaboost[6]. Let $C_t(\mathbf{x}) = \sum_{j=1}^{t} \alpha_j T_j(\mathbf{x})$ be the boosted linear combination of classifiers until t^{th} iteration. Let the error to be minimized over α_t be the sum of its exponential loss on each data point,

$$\mathbf{E}_t = \underline{\sum_{i=1}^{m} \exp\left(-y^{(i)} \underline{C_t\left(\mathbf{x}^{(i)}\right)}\right)} = \sum_{i=1}^{m} \exp\left(-\left(y^{(i)} \underline{\sum_{j=1}^{t} \alpha_j T_j(\mathbf{x}^{(i)})}\right)\right)$$

Prove the following claims for Adaboost

1. Claim1: The error that is the sum of exponential loss on each data point is an upper bound on the simple sum of training errors on each data point

   Solution:

   The simple sum of training errors on each data point is

   $$\underline{\sum_{i=1}^{m} \delta\left(y^{(i)} \neq sign\left(C_t\left(\mathbf{x}^{(i)}\right)\right)\right)} = \sum_{i=1}^{m} \exp\left(\delta\left(y^{(i)} \neq sign\left(\sum_{i=1}^{m} \alpha_i T_i(\mathbf{x}^{(i)})\right)\right)\right)$$

   $\delta(\text{assertion}) = 1$ iff assertion is true & $= 0$ o/w

   We next note that the exponential function[7] is a convex function. That is,

   $$\exp\left(\sum_{i=1}^{m} \beta_i r_i\right) \leq \sum_{i=1}^{m} \beta_i \exp(r_i)$$

   for each $\beta_i \in [0, 1]$ such that $\sum_{i=1}^{m} \beta_i = 1$. By this convexity, one can derive (see Figure 4 on the next page:

   $$\mathbf{E}_t = \sum_{i=1}^{m} \delta\left(y^{(i)} \neq sign\left(C_t\left(\mathbf{x}^{(i)}\right)\right)\right) \leq \sum_{i=1}^{m} \exp\left(-y^{(i)} C_t\left(\mathbf{x}^{(i)}\right)\right)$$

2. Claim2: $\boldsymbol{\alpha_t} = (1/2)\ln\left((1 - err_t)/err_t\right)$ actually minimizes this upper bound.

3. (Advanced and Optional) Claim3: If each classifier is slightly better than random, that is if $err_t < 1/K$, Adaboost achieves zero training error exponentially fast

[6] https://www.cse.iitb.ac.in/~cs725/notes/lecture-slides/lecture-25-annotated.pdf

[7] https://en.wikipedia.org/wiki/Exponential_function

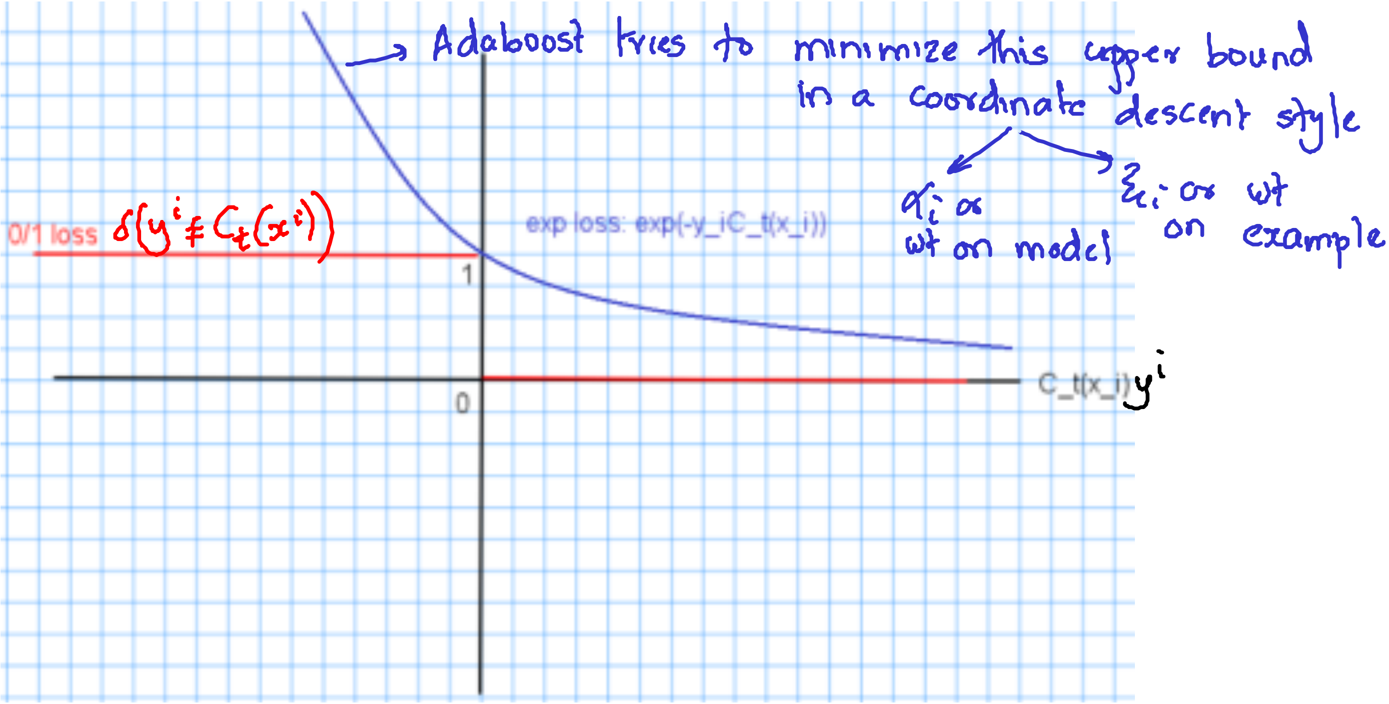


Figure 4: Figure Illustrating how the **exponential loss** is an upper bound for the **0/1 loss**