CSC 411: Introduction to Machine Learning

CSC 411 Lecture 9: SVMs and Boosting

Mengye Ren and Matthew MacKay

University of Toronto

Overview

- Support Vector Machines
- Connection between Exponential Loss and AdaBoost

Binary Classification with a Linear Model

- Classification: Predict a discrete-valued target
- Binary classification: Targets $t \in \{-1, +1\}$
- Linear model:

$$z = \mathbf{w}^\top \mathbf{x} + b$$
$$y = sign(z)$$

- Question: How should we choose $\mathbf{w}$ and b?

Zero-One Loss

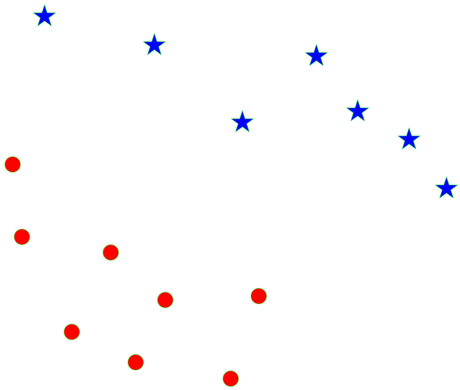
- We can use the $0 - 1$ loss function, and find the weights that minimize it over data points

$$\mathcal{L}_{0-1}(y, t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{if } y \neq t \end{cases}$$
$$= \mathbb{I}\{y \neq t\}.$$

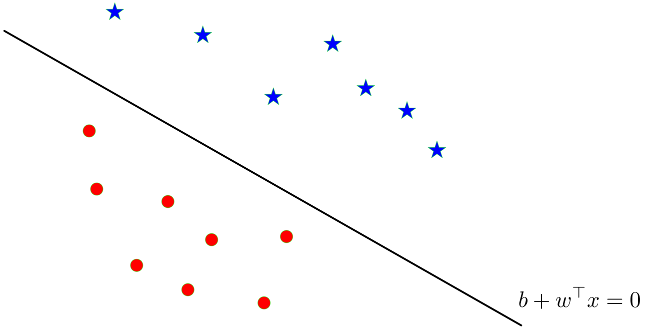
- But minimizing this loss is computationally difficult, and it can't distinguish different hypotheses that achieve the same accuracy.
- We investigated some other loss functions that are easier to minimize, e.g., logistic regression with the cross-entropy loss $\mathcal{L}_{\mathrm{CE}}$.
- Let's consider a different approach, starting from the geometry of binary classifiers.

Separating Hyperplanes

Suppose we are given these data points from two different classes and want to find a linear classifier that separates them.

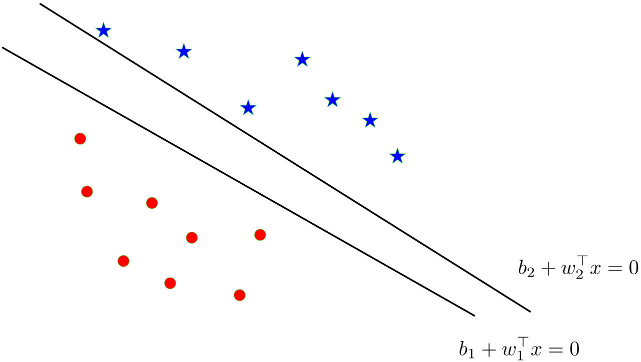


Separating Hyperplanes



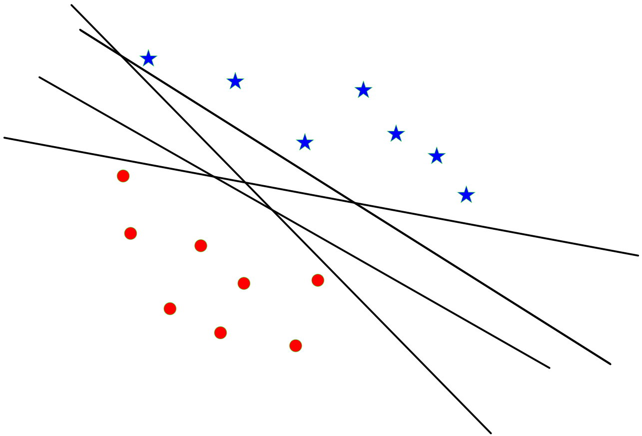
- The decision boundary looks like a line because $\mathbf{x} \in \mathbb{R}^2$, but think about it as a $D - 1$ dimensional hyperplane.
- Recall that a hyperplane is described by points $\mathbf{x} \in \mathbb{R}^D$ such that $f(\mathbf{x}) = \mathbf{w}^\top x + b = 0$.

Separating Hyperplanes



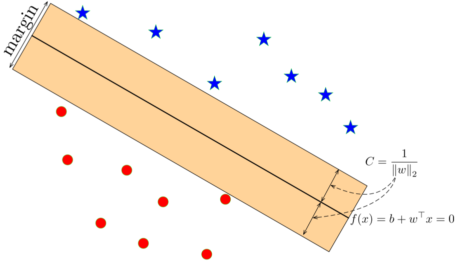
- There are multiple separating hyperplanes, described by different parameters $(\mathbf{w}, b)$.

Separating Hyperplanes



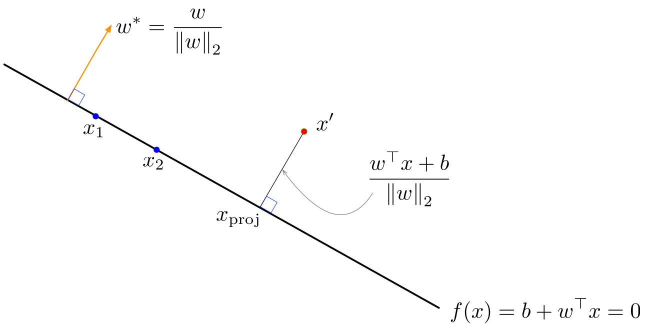
Optimal Separating Hyperplane

Optimal Separating Hyperplane: A hyperplane that separates two classes and maximizes the distance to the closest point from either class, i.e., maximize the **margin** of the classifier.



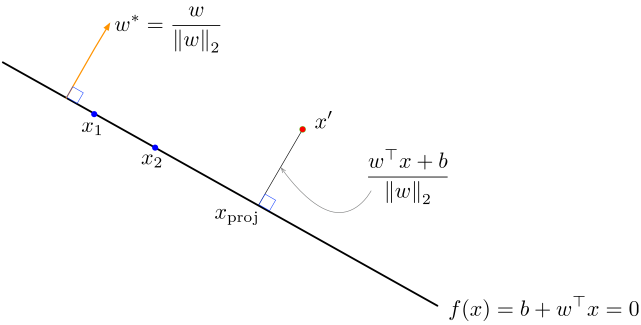
Intuitively, ensuring the decision boundary is not too close to any data points leads to better generalization on the test data.

Geometry of Points and Planes



- Recall that the decision hyperplane is orthogonal (perpendicular) to $\mathbf{w}$.
- The vector $\mathbf{w}^* = \frac{\mathbf{w}}{\|\mathbf{w}\|_2}$ is a unit vector pointing in the same direction as $\mathbf{w}$.
- The same hyperplane could equivalently be defined in terms of $\mathbf{w}^*$.

Geometry of Points and Planes



- Let's compute the distance between a point $\mathbf{x}'$ and the hyperplane $H = \{\mathbf{x} : \mathbf{w}^T\mathbf{x} + b = 0\}$
- We can write: $\mathbf{x}' = \mathbf{x}_{\text{proj}} + \mathbf{x}_N$ where $\mathbf{x}_{\text{proj}} \in H$ and $\mathbf{x}_N \in \text{span}(\mathbf{w})$
- Note: $\|\mathbf{x}_N\|_2$ is the distance from $\mathbf{x}'$ to H

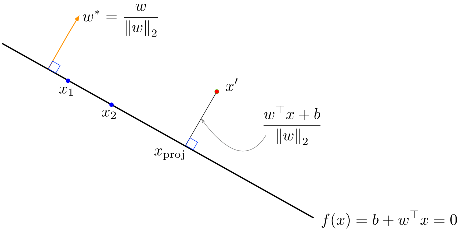
Geometry of Points and Planes

- Since $\mathbf{x}_N \in \text{span}(\mathbf{w})$, can write $\mathbf{x}_N = \lambda \mathbf{w}$ for some λ
- Then:

$$\begin{aligned} \mathbf{w}^T\mathbf{x}' + b &= \mathbf{w}^T(\mathbf{x}_{\text{proj}} + \mathbf{x}_N) + b \\ &= \mathbf{w}^T\mathbf{x}_{\text{proj}} + b + \mathbf{w}^T(\lambda\mathbf{w}) \\ &= 0 + \lambda \|\mathbf{w}\|_2^2 \\ &= \lambda \|\mathbf{w}\| \|\mathbf{w}\| \end{aligned}$$

- Hence, $\|\mathbf{x}_N\| = |\lambda| \|\mathbf{w}\| = \frac{|\mathbf{w}^T\mathbf{x}' + b|}{\|\mathbf{w}\|}$

Geometry of Points and Planes



The (signed) distance of a point $\mathbf{x}'$ to the hyperplane is

$$\frac{\mathbf{w}^\top \mathbf{x}' + b}{\|\mathbf{w}\|_2}$$

Maximizing Margin as an Optimization Problem

- Recall: the classification for the i-th data point is correct when

$$\text{sign}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) = t^{(i)}$$

- This can be rewritten as

$$t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) > 0$$

- Enforcing a margin of C:

$$t^{(i)} \cdot \underbrace{\frac{(\mathbf{w}^\top \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|_2}}_{\text{signed distance}} \geq C$$

Maximizing Margin as an Optimization Problem

Max-margin objective:

$$\max_{\mathbf{w},b} C$$

$$\text{s.t.} \; \frac{t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|_2} \geq C \qquad i = 1, \ldots, N$$

- Note: can scale $\mathbf{w}$ and b by any positive value and get the same decision boundary
- This means we can choose to enforce $\|\mathbf{w}\|_2 = r$ for any $r > 0$ without changing the original solution
- Let's add the constraint $\|\mathbf{w}\|_2 = \frac{1}{C}$

$$\max_{\mathbf{w},b} C$$

$$\text{s.t.} \; \frac{t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|_2} \geq C \qquad i = 1, \ldots, N$$

$$\|\mathbf{w}\|_2 = \frac{1}{C}$$

Maximizing Margin as an Optimization Problem

$$\max_{\mathbf{w},b} C$$

$$\text{s.t. } \frac{t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|_2} \geq C \qquad i = 1, \ldots, N$$

$$\|\mathbf{w}\|_2 = \frac{1}{C}$$

Note that if $\|\mathbf{w}\|_2 = \frac{1}{C}$ then:

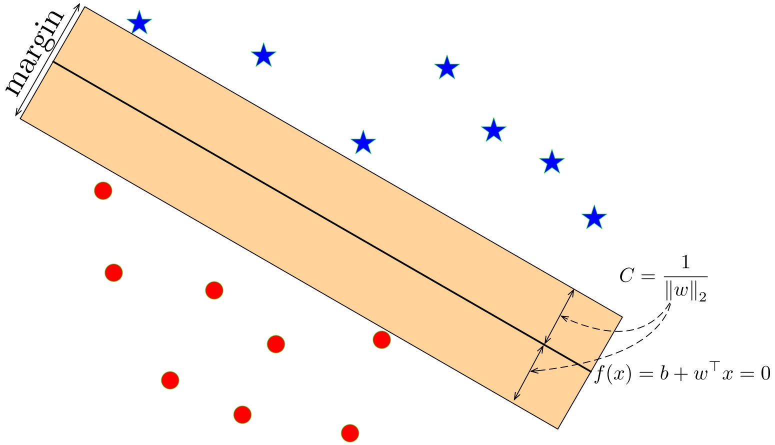
$$\underbrace{\frac{t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|_2} \geq \frac{1}{\|\mathbf{w}\|_2}}_{\textbf{geometric margin constraint}} \qquad \iff \qquad \underbrace{t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1}_{\textbf{algebraic margin constraint}}$$

Plugging in $C = \frac{1}{\|\mathbf{w}\|_2}$, equivalent optimization objective:

$$\min \|\mathbf{w}\|_2^2$$

$$\text{s.t. } t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1 \qquad i = 1, \ldots, N$$

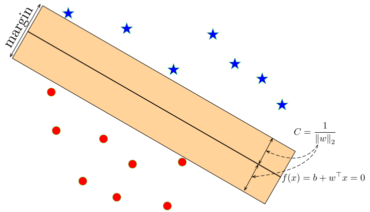
Maximizing Margin as an Optimization Problem



Maximizing Margin as an Optimization Problem

Algebraic max-margin objective:

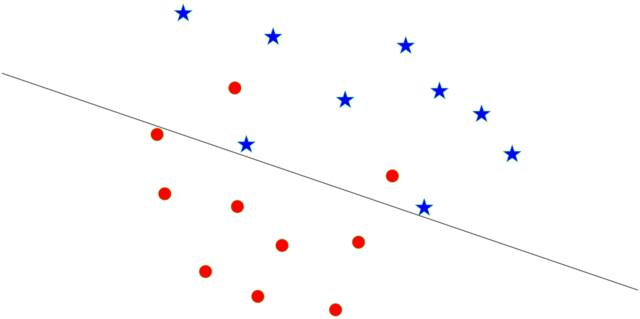
$$\min_{\mathbf{w},b} \|\mathbf{w}\|_2^2$$
$$\text{s.t. } t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1 \qquad i = 1, \ldots, N$$



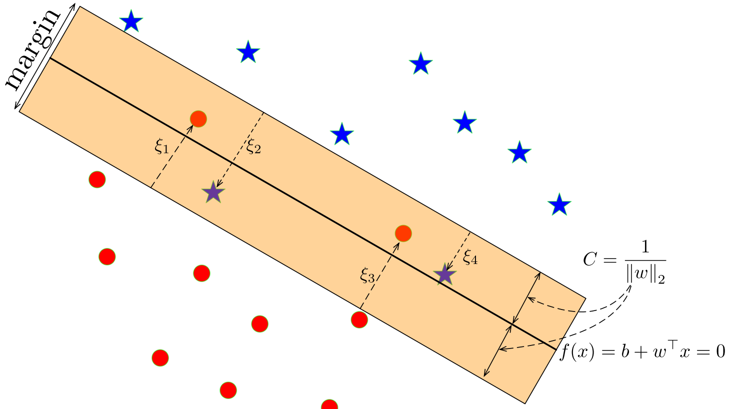
- Observe: if the margin constraint is not tight for $\mathbf{x}^{(i)}$, we could remove it from the training set and the optimal $\mathbf{w}$ would be the same.
- The important training examples are the ones with algebraic margin 1, and are called **support vectors**.
- Hence, this algorithm is called the (hard) **Support Vector Machine (SVM)** (or Support Vector Classifier).
- SVM-like algorithms are often called **max-margin** or **large-margin**.

Non-Separable Data Points

How can we apply the max-margin principle if the data are **not** linearly separable?



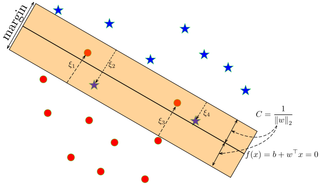
Maximizing Margin for Non-Separable Data Points



Main Idea:

- Allow some points to be within the margin or even be misclassified; we represent this with **slack variables** ξ_i.
- But constrain or penalize the total amount of slack.

Maximizing Margin for Non-Separable Data Points



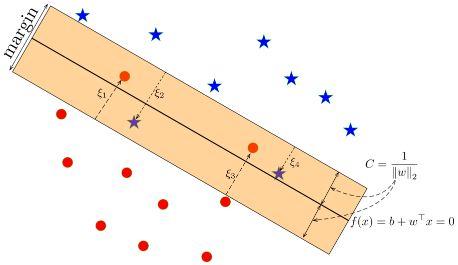
- **Soft margin constraint**:

$$\frac{t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|_2} \geq C(1 - \xi_i),$$

  for $\xi_i \geq 0$.

- Penalize $\sum_i \xi_i$

Maximizing Margin for Non-Separable Data Points



$$\max_{\mathbf{w},b,\xi} C + \gamma \sum_{i=1}^{N} \xi_i$$

$$\text{s.t. } \frac{t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|_2} \geq C(1 - \xi_i) \qquad i = 1, \ldots, N$$

$$\xi_i \geq 0 \qquad i = 1, \ldots, N$$

Maximizing Margin for Non-Separable Data Points

Do the same $\|\mathbf{w}\|_2 = \frac{1}{C}$ trick to derive the **Soft-margin SVM** objective:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2} \|\mathbf{w}\|_2^2 + \gamma \sum_{i=1}^{N} \xi_i$$

$$\text{s.t.} \quad t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1 - \xi_i \qquad i = 1, \ldots, N$$

$$\xi_i \geq 0 \qquad i = 1, \ldots, N$$

- γ is a hyperparameter that trades off the margin with the amount of slack.
  - For $\gamma = 0$, we'll get $\mathbf{w} = 0$. (Why?)
  - As $\gamma \to \infty$ we get the hard-margin objective.
- Note: it is also possible to constrain $\sum_i \xi_i$ instead of penalizing it.

From Margin Violation to Hinge Loss

Let's simplify the soft margin constraint by eliminating ξ_i. Recall:

$$\begin{aligned} t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) &\geq 1 - \xi_i \qquad & i = 1, \ldots, N \\ \xi_i &\geq 0 & i = 1, \ldots, N \end{aligned}$$

- Rewrite as $\xi_i \geq 1 - t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b)$.
- **Case 1:** $1 - t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \leq 0$
  - The smallest non-negative ξ_i that satisfies the constraint is $\xi_i = 0$.
- **Case 2:** $1 - t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) > 0$
  - The smallest ξ_i that satisfies the constraint is $\xi_i = 1 - t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b)$.
- Hence, $\xi_i = \max\{0, 1 - t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b)\}$.
- Therefore, the slack penalty can be written as

$$\sum_{i=1}^{N} \xi_i = \sum_{i=1}^{N} \max\{0, 1 - t^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b)\}.$$

From Margin Violation to Hinge Loss

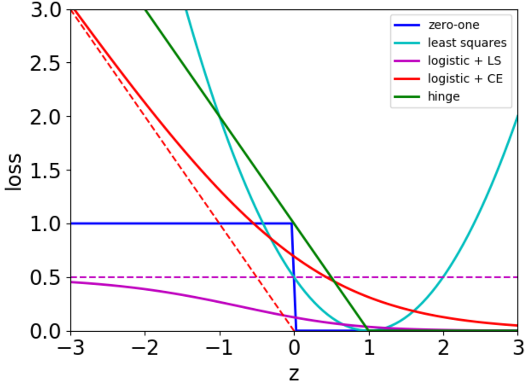
If we write $y^{(i)}(\mathbf{w}, b) = \mathbf{w}^\top \mathbf{x} + b$, then the optimization problem can be written as

$$\min_{\mathbf{w},b,\xi} \sum_{i=1}^{N} \max\{0, 1 - t^{(i)} y^{(i)}(\mathbf{w}, b)\} + \frac{1}{2\gamma} \|\mathbf{w}\|_2^2$$

- The loss function $\mathcal{L}_{\mathrm{H}}(y, t) = \max\{0, 1 - ty\}$ is called the **hinge** loss.
- The second term is the L_2-norm of the weights.
- Hence, the soft-margin SVM can be seen as a linear classifier with hinge loss and an L_2 regularizer.

Revisiting Loss Functions for Classification

Hinge loss compared with other loss functions



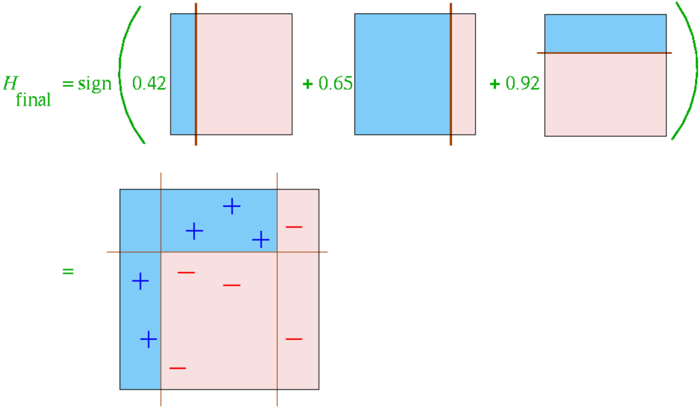
SVMs: What we Left Out

What we left out:

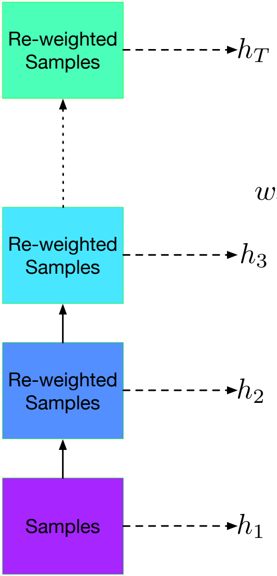
- How to fit $\mathbf{w}$:
  - One option: gradient descent
  - Can reformulate with the Lagrange dual
- The "kernel trick" converts it into a powerful nonlinear classifier. We'll cover this later in the course.
- Classic results from learning theory show that a large margin implies good generalization.

AdaBoost Revisited

Part 2: reinterpreting AdaBoost in terms of what we've learned about loss functions.



AdaBoost Revisited



$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

$$w_i \leftarrow w_i \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(\mathbf{i})}) \neq \mathbf{t}^{(\mathbf{i})}\}\right)$$

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\text{err}_t}{\text{err}_t}\right)$$

$$\text{err}_t = \frac{\sum_{i=1}^{N} w_i \mathbb{I}\{h_t(\mathbf{x}^{(\mathbf{i})} \neq \mathbf{t}^{(\mathbf{i})}\}}{\sum_{i=1}^{N} w_i}$$

Additive Models

- Consider a hypothesis class $\mathcal{H}$ with each $h_i : \mathbf{x} \mapsto \{-1, +1\}$ within $\mathcal{H}$, i.e., $h_i \in \mathcal{H}$. These are the "weak learners", and in this context they're also called **bases**.
- An **additive model** with m terms is given by

$$H_m(x) = \sum_{i=1}^{m} \alpha_i h_i(\mathbf{x}),$$

  where $(\alpha_1, \cdots, \alpha_m) \in \mathbb{R}^m$.
- Observe that we're taking a linear combination of base classifiers, just like in boosting.
- We'll now interpret AdaBoost as a way of fitting an additive model.

Stagewise Training of Additive Models

A greedy approach to fitting additive models, known as **stagewise training**:

1. Initialize $H_0(x) = 0$
2. For $m = 1$ to T:
   - Compute the m-th hypothesis and its coefficient, assuming previous additive model H_{m-1} is fixed:

$$(h_m, \alpha_m) \leftarrow \underset{h \in \mathcal{H}, \alpha}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}\left(H_{m-1}(\mathbf{x}^{(i)}) + \alpha h(\mathbf{x}^{(i)}), t^{(i)}\right)$$

   - Add it to the additive model

$$H_m = H_{m-1} + \alpha_m h_m$$

Additive Models with Exponential Loss

Consider the exponential loss

$$\mathcal{L}_{\mathrm{E}}(y, t) = \exp(-ty).$$

We want to see how the stagewise training of additive models can be done.

$$\begin{aligned}
(h_m, \alpha_m) \leftarrow & \underset{h \in \mathcal{H}, \alpha}{\operatorname{argmin}} \sum_{i=1}^{N} \exp\left(-\left[H_{m-1}(\mathbf{x}^{(i)}) + \alpha h(\mathbf{x}^{(i)})\right] t^{(i)}\right) \\
&= \sum_{i=1}^{N} \exp\left(-H_{m-1}(\mathbf{x}^{(i)}) t^{(i)} - \alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) \\
&= \sum_{i=1}^{N} \exp\left(-H_{m-1}(\mathbf{x}^{(i)}) t^{(i)}\right) \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) \\
&= \sum_{i=1}^{N} w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right).
\end{aligned}$$

Here we defined $w_i^{(m)} \triangleq \exp\left(-H_{m-1}(\mathbf{x}^{(i)}) t^{(i)}\right)$.

Additive Models with Exponential Loss

We want to solve the following minimization problem:

$$(h_m, \alpha_m) \leftarrow \underset{h \in \mathcal{H}, \alpha}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right).$$

- If $h(\mathbf{x}^{(i)}) = t^{(i)}$, we have $\exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) = \exp(-\alpha)$.
- If $h(\mathbf{x}^{(i)}) \neq t^{(i)}$, we have $\exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) = \exp(+\alpha)$.

(recall that we are in the binary classification case with $\{-1, +1\}$ output values).
We can divide the summation to two parts:

$$\sum_{i=1}^{N} w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) = \underbrace{e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) = t_i\}}_{\text{correct predictions}} + \underbrace{e^{\alpha} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\}}_{\text{incorrect predictions}}$$

$$= (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\} +$$

$$e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \left[\mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\} + \mathbb{I}\{h(\mathbf{x}^{(i)}) = t_i\}\right]$$

Additive Models with Exponential Loss

We can divide the summation to two parts:

$$\sum_{i=1}^{N} w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) = \underbrace{e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) = t_i\}}_{\text{correct predictions}} + \underbrace{e^{\alpha} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\}}_{\text{incorrect predictions}}$$

$$= e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) = t_i\} + e^{\alpha} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\}$$

$$-e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\} + e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\}$$

$$= (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\} +$$

$$e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \left[\mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\} + \mathbb{I}\{h(\mathbf{x}^{(i)}) = t_i\}\right]$$

Additive Models with Exponential Loss

$$\sum_{i=1}^{N} w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) = (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)} \neq t_i\} +$$

$$e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \left[\mathbb{I}\{h(\mathbf{x}^{(i)} \neq t_i\} + \mathbb{I}\{h(\mathbf{x}^{(i)}) = t_i\}\right]$$

$$= (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\} + e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)}.$$

Let us first optimize h: The second term on the RHS does not depend on h. So we get

$$h_m \leftarrow \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) \equiv \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\}.$$

This means that h_m is the minimizer of the weighted 0/1-loss.

Additive Models with Exponential Loss

Now that we obtained h_m, we want to find α: Define the weighted classification error:

$$\mathrm{err}_m = \frac{\sum_{i=1}^N w_i^{(m)} \mathbb{I}\{h_m(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^N w_i^{(m)}}$$

With this definition and $\min_{h \in \mathcal{H}} \sum_{i=1}^N w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) = \sum_{i=1}^N w_i^{(m)} \mathbb{I}\{h_m(\mathbf{x}^{(i)}) \neq t_i\}$, we have

$$\min_{\alpha} \min_{h \in \mathcal{H}} \sum_{i=1}^N w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) =$$

$$\min_{\alpha} \left\{ (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^N w_i^{(m)} \mathbb{I}\{h_m(\mathbf{x}^{(i)}) \neq t_i\} + e^{-\alpha} \sum_{i=1}^N w_i^{(m)} \right\}$$

$$= \min_{\alpha} \left\{ (e^{\alpha} - e^{-\alpha}) \mathrm{err}_m \left(\sum_{i=1}^N w_i^{(m)} \right) + e^{-\alpha} \left(\sum_{i=1}^N w_i^{(m)} \right) \right\}$$

Take derivative w.r.t. α and set it to zero. We get that

$$e^{2\alpha} = \frac{1 - \mathrm{err}_m}{\mathrm{err}_m} \Rightarrow \alpha = \frac{1}{2} \log \left(\frac{1 - \mathrm{err}_m}{\mathrm{err}_m} \right).$$

Additive Models with Exponential Loss

The updated weights for the next iteration is

$$\begin{aligned}
w_i^{(m+1)} &= \exp\left(-H_m(\mathbf{x}^{(i)})t^{(i)}\right) \\
&= \exp\left(-\left[H_{m-1}(\mathbf{x}^{(i)}) + \alpha_m h_m(\mathbf{x}^{(i)})\right] t^{(i)}\right) \\
&= \exp\left(-H_{m-1}(\mathbf{x}^{(i)})t^{(i)}\right) \exp\left(-\alpha_m h_m(\mathbf{x}^{(i)})t^{(i)}\right) \\
&= w_i^{(m)} \exp\left(-\alpha_m h_m(\mathbf{x}^{(i)})t^{(i)}\right)
\end{aligned}$$

Additive Models with Exponential Loss

To summarize, we obtain the additive model $H_m(x) = \sum_{i=1}^{m} \alpha_i h_i(\mathbf{x})$ with

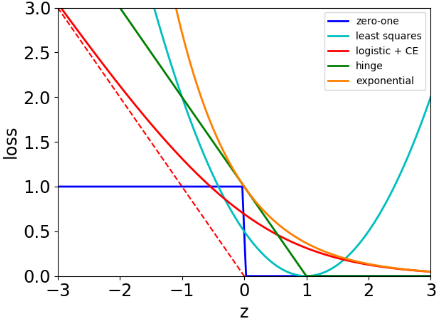
$$h_m \leftarrow \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\},$$

$$\alpha = \frac{1}{2} \log \left(\frac{1 - \text{err}_m}{\text{err}_m} \right), \qquad \text{where } \text{err}_m = \frac{\sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h_m(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i^{(m)}},$$

$$w_i^{(m+1)} = w_i^{(m)} \exp \left(-\alpha_m h_m(\mathbf{x}^{(i)}) t^{(i)} \right).$$

We derived the AdaBoost algorithm!

Revisiting Loss Functions for Classification



- If AdaBoost is minimizing exponential loss, what does that say about its behavior (compared to, say, logistic regression)?
- This interpretation allows boosting to be generalized to lots of other loss functions.