Overview

- Support Vector Machines
- Connection between Exponential Loss and AdaBoost
Classification: Predict a discrete-valued target

Binary classification: Targets $t \in \{-1, +1\}$

Linear model:

$$z = \mathbf{w}^\top \mathbf{x} + b$$

$$y = \text{sign}(z)$$

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

\[ L_{0-1}(y, t) = \begin{cases} 
0 & \text{if } y = t \\
1 & \text{if } y \neq t 
\end{cases} = \mathbb{I}\{y \neq t\}. \]

- 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 $L_{CE}$.

- Let's consider a different approach, starting from the geometry of binary classifiers.
Suppose we are given these data points from two different classes and want to find a linear classifier that separates them.
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 $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b = 0$. 
There are multiple separating hyperplanes, described by different parameters \((w, b)\).
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.
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 $\mathbf{w}^*$. 
Let’s compute the distance between a point $x'$ and the hyperplane $H = \{ x : w^T x + b = 0 \}$

We can write: $x' = x_{\text{proj}} + x_N$ where $x_{\text{proj}} \in H$ and $x_N \in \text{span}(w)$

Note: $\|x_N\|_2$ is the distance from $x'$ to $H$
Geometry of Points and Planes

Since $x_N \in \text{span}(w)$, can write $x_N = \lambda w$ for some $\lambda$

Then:

$$w^T x' + b = w^T (x_{\text{proj}} + x_N) + b$$
$$= w^T x_{\text{proj}} + b + w^T (\lambda w)$$
$$= 0 + \lambda \|w\|^2$$
$$= \lambda \|w\|\|w\|$$

Hence, $\|x_N\| = |\lambda| \|w\| = \frac{|w^T x' + b|}{\|w\|}$
The (signed) distance of a point $x'$ to the hyperplane is

$$
\frac{w^\top x' + b}{\|w\|_2}
$$
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 $C$:
  \[ t^{(i)} \cdot \frac{(\mathbf{w}^\top \mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|_2} \geq C \]
  
  \[
  \begin{array}{c}
  \underbrace{\text{signed distance}} \\
  \end{array}
  \]
Maximizing Margin as an Optimization Problem

Max-margin objective:

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

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

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

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

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

$$\|w\|_2 = \frac{1}{C}$$
Maximizing Margin as an Optimization Problem

\[
\begin{align*}
\max_{w,b} & \quad C \\
\text{s.t.} & \quad \frac{t^{(i)}(w^\top x^{(i)} + b)}{\|w\|_2} \geq C \quad i = 1, \ldots, N \\
& \quad \|w\|_2 = \frac{1}{C}
\end{align*}
\]

Note that if \( \|w\|_2 = \frac{1}{C} \) then:

\[
\begin{align*}
\frac{t^{(i)}(w^\top x^{(i)} + b)}{\|w\|_2} \geq \frac{1}{\|w\|_2} & \iff t^{(i)}(w^\top x^{(i)} + b) \geq 1 \\
\text{geometric margin constraint} & \text{algebraic margin constraint}
\end{align*}
\]

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

\[
\begin{align*}
\min & \quad \|w\|_2^2 \\
\text{s.t.} & \quad t^{(i)}(w^\top x^{(i)} + b) \geq 1 \quad i = 1, \ldots, N
\end{align*}
\]
Maximizing Margin as an Optimization Problem

\[ C = \frac{1}{\|w\|_2} \]

\[ f(x) = b + w^\top x = 0 \]
Algebraic max-margin objective:

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

- Observe: if the margin constraint is not tight for \(x^{(i)}\), we could remove it from the training set and the optimal \(w\) 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?
Main Idea:
- Allow some points to be within the margin or even be misclassified; we represent this with slack variables $\xi_i$.
- 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

\[
\begin{align*}
\max_{w, b, \xi} & \quad C + \gamma \sum_{i=1}^{N} \xi_i \\
\text{s.t.} & \quad \frac{t^{(i)}(w^\top x^{(i)} + b)}{\|w\|_2} \geq C(1 - \xi_i) \quad i = 1, \ldots, N \\
& \quad \xi_i \geq 0 \quad i = 1, \ldots, N
\end{align*}
\]
Maximizing Margin for Non-Separable Data Points

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

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

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

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

- $\gamma$ is a hyperparameter that trades off the margin with the amount of slack.
  - For $\gamma = 0$, we'll get $w = 0$. (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.
Let’s simplify the soft margin constraint by eliminating $\xi_i$. Recall:

$$
\begin{align*}
t^{(i)}(w^T x^{(i)} + b) &\geq 1 - \xi_i & i = 1, \ldots, N \\
\xi_i &\geq 0 & i = 1, \ldots, N
\end{align*}
$$

- Rewrite as $\xi_i \geq 1 - t^{(i)}(w^T x^{(i)} + b)$.

**Case 1:** $1 - t^{(i)}(w^T x^{(i)} + b) \leq 0$

  - The smallest non-negative $\xi_i$ that satisfies the constraint is $\xi_i = 0$.

**Case 2:** $1 - t^{(i)}(w^T x^{(i)} + b) > 0$

  - The smallest $\xi_i$ that satisfies the constraint is $\xi_i = 1 - t^{(i)}(w^T x^{(i)} + b)$.

Hence, $\xi_i = \max\{0, 1 - t^{(i)}(w^T x^{(i)} + b)\}$.

Therefore, the slack penalty can be written as

$$
\sum_{i=1}^{N} \xi_i = \sum_{i=1}^{N} \max\{0, 1 - t^{(i)}(w^T x^{(i)} + b)\}.
$$
If we write $y^{(i)}(w, b) = w^\top x + b$, then the optimization problem can be written as

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

- The loss function $L_H(y, t) = \max\{0, 1 - ty\}$ 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

```
loss

0.0 0.5 1.0 1.5 2.0 2.5 3.0

-3 -2 -1 0 1 2 3
```

Legend:
- blue: zero-one
- cyan: least squares
- purple: logistic + LS
- red: logistic + CE
- green: hinge
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.

\[
H_{\text{final}} = \text{sign} \left( 0.42 + 0.65 + 0.92 \right)
\]
AdaBoost Revisited

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

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

\[ \alpha_t = \frac{1}{2} \log \left( \frac{1 - err_t}{err_t} \right) \]

\[ err_t = \frac{\sum_{i=1}^{N} w_i \mathbb{I}\{h_t(x^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i} \]
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

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

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.
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:
     $$(h_m, \alpha_m) \leftarrow \arg\min_{h \in \mathcal{H}, \alpha} \sum_{i=1}^{N} L \left( H_{m-1}(x^{(i)}) + \alpha h(x^{(i)}), t^{(i)} \right)$$
   - Add it to the additive model
     $$H_m = H_{m-1} + \alpha_m h_m$$
Consider the exponential loss

\[ \mathcal{L}_E(y, t) = \exp(-ty). \]

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

\[
(h_m, \alpha_m) \leftarrow \arg\min_{h \in \mathcal{H}, \alpha} \sum_{i=1}^{N} \exp \left( - \left[ H_{m-1}(x^{(i)}) + \alpha h(x^{(i)}) \right] t^{(i)} \right)
\]

\[
= \sum_{i=1}^{N} \exp \left( -H_{m-1}(x^{(i)})t^{(i)} - \alpha h(x^{(i)})t^{(i)} \right)
\]

\[
= \sum_{i=1}^{N} \exp \left( -H_{m-1}(x^{(i)})t^{(i)} \right) \exp \left( -\alpha h(x^{(i)})t^{(i)} \right)
\]

\[
= \sum_{i=1}^{N} w_i^{(m)} \exp \left( -\alpha h(x^{(i)})t^{(i)} \right)
\]

Here we defined \( w_i^{(m)} \triangleq \exp \left( -H_{m-1}(x^{(i)})t^{(i)} \right). \)
Additive Models with Exponential Loss

We want to solve the following minimization problem:

\[(h_m, \alpha_m) \leftarrow \argmin_{h \in \mathcal{H}, \alpha} \sum_{i=1}^{N} w_i^{(m)} \exp \left(-\alpha h(x^{(i)}) t^{(i)}\right) .\]

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

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

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

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

\[
eq e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \left[\mathbb{I}\{h(x^{(i)}) \neq t_i\} + \mathbb{I}\{h(x^{(i)}) = t_i\}\right]
\]
Additive Models with Exponential Loss

We can divide the summation to two parts:

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

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

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

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

\[
e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \left[ \mathbb{1}\{h(x^{(i)}) \neq t_i\} + \mathbb{1}\{h(x^{(i)}) = t_i\} \right]
\]
Let us first optimize $h$: The second term on the RHS does not depend on $h$. So we get

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

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 $\alpha$: Define the weighted classification error:

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

With this definition and

$$\min_{h \in \mathcal{H}} \sum_{i=1}^{N} w_i^{(m)} \exp (-\alpha h(x^{(i)}) t^{(i)}) = \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h_m(x^{(i)}) \neq t_i\},$$

we have

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

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

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

Take derivative w.r.t. $\alpha$ and set it to zero. We get that

$$e^{2\alpha} = \frac{1 - err_m}{err_m} \Rightarrow \alpha = \frac{1}{2} \log \left( \frac{1 - err_m}{err_m} \right).$$
The updated weights for the next iteration is

\[ w_i^{(m+1)} = \exp \left( -H_m(x^{(i)}) t^{(i)} \right) \]
\[ = \exp \left( - \left[ H_{m-1}(x^{(i)}) + \alpha_m h_m(x^{(i)}) \right] t^{(i)} \right) \]
\[ = \exp \left( -H_{m-1}(x^{(i)}) t^{(i)} \right) \exp \left( -\alpha_m h_m(x^{(i)}) t^{(i)} \right) \]
\[ = w_i^{(m)} \exp \left( -\alpha_m h_m(x^{(i)}) t^{(i)} \right) \]
To summarize, we obtain the additive model $H_m(x) = \sum_{i=1}^{m} \alpha_i h_i(x)$ with

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

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

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

We derived the AdaBoost algorithm!
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.