COMS 4771 Lecture 5

1. Linear classifiers (+ linear algebra review)
2. Linearly separable instances.
LINEAR CLASSIFIERS
Axis-aligned threshold functions

Decision tree learning

Basic step in greedy decision tree learning (with axis-aligned splits in $\mathcal{X} = \mathbb{R}^d$):

$$\arg \min_h \frac{|S_{h,0}|}{|S|} u(S_{h,0}) + \frac{|S_{h,1}|}{|S|} u(S_{h,1})$$

where $u$ is some uncertainty measure,

$$S_{h,0} = \{(x, y) \in S : h(x) = 0\}, \quad S_{h,1} = \{(x, y) \in S : h(x) = 1\},$$

and the minimization is over splitting rules of the form

$$h(x) = 1\{x_i > t\}, \quad i \in [d], t \in \mathbb{R}.$$
Axis-aligned threshold functions

Decision tree learning
Basic step in greedy decision tree learning (with axis-aligned splits in $\mathcal{X} = \mathbb{R}^d$):

$$\arg\min_h \frac{|S_{h,0}|}{|S|} u(S_{h,0}) + \frac{|S_{h,1}|}{|S|} u(S_{h,1})$$

where $u$ is some uncertainty measure,

$$S_{h,0} = \{(x, y) \in S : h(x) = 0\}, \quad S_{h,1} = \{(x, y) \in S : h(x) = 1\},$$

and the minimization is over splitting rules of the form

$$h(x) = 1_{\{x_i > t\}}, \quad i \in [d], t \in \mathbb{R}.$$

Axis-aligned threshold functions
When $u$ is classification error and $\mathcal{Y} = \{-1, +1\}$, we are equivalently doing the following:

$$\arg\min_{i \in [d], v \in \{-1, +1\}, t \in \mathbb{R}} \sum_{(x, y) \in S} 1\{\text{sign}(vx_i - t) \neq y\}.$$
Axis-aligned threshold functions

Decision tree learning
Basic step in greedy decision tree learning (with axis-aligned splits in $\mathcal{X} = \mathbb{R}^d$):

$$\arg\min_h \frac{|S_{h,0}|}{|S|} u(S_{h,0}) + \frac{|S_{h,1}|}{|S|} u(S_{h,1})$$

where $u$ is some uncertainty measure,

$$S_{h,0} = \{(x, y) \in S : h(x) = 0\}, \quad S_{h,1} = \{(x, y) \in S : h(x) = 1\},$$

and the minimization is over splitting rules of the form

$$h(x) = 1_{\{x_i > t\}}, \quad i \in [d], t \in \mathbb{R}.$$

Axis-aligned threshold functions
When $u$ is classification error and $\mathcal{Y} = \{-1, +1\}$, we are equivalently doing the following:

$$\arg\min_{i \in [d], v \in \{-1, +1\}, t \in \mathbb{R}} \sum_{(x, y) \in S} 1\{\text{sign}(vx_i - t) \neq y\}.$$

i.e., looking at classifiers of the form $f_{i,v,t}(x) = \text{sign}(vx_i - t)$. 
A natural generalization of axis-aligned threshold functions

\[ f_{i,v,t}(x) = \text{sign}(vx_i - t), \quad i \in [d], v \in \{-1, +1\}, t \in \mathbb{R}, \]

are \textbf{linear threshold functions} (or \textbf{linear classifiers}):

\[ f_{w,t}(x) = \text{sign}(\langle w, x \rangle - t), \quad w \in \mathbb{R}^d, t \in \mathbb{R}. \]
A natural generalization of axis-aligned threshold functions

\[ f_{i,v,t}(x) = \text{sign}(vx_i - t), \quad i \in [d], v \in \{-1, +1\}, t \in \mathbb{R}, \]

are **linear threshold functions** (or **linear classifiers**):

\[ f_{w,t}(x) = \text{sign}(\langle w, x \rangle - t), \quad w \in \mathbb{R}^d, t \in \mathbb{R}. \]

**Interpretation**: does a weighted linear combination of input features exceed a threshold?

\[ \langle w, x \rangle = \sum_{i=1}^{d} w_i x_i > t \]
A natural generalization of axis-aligned threshold functions

\[ f_{i,v,t}(\mathbf{x}) = \text{sign}(vx_i - t), \quad i \in [d], v \in \{-1, +1\}, t \in \mathbb{R}, \]

are **linear threshold functions** (or **linear classifiers**):

\[ f_{\mathbf{w},t}(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle - t), \quad \mathbf{w} \in \mathbb{R}^d, \; t \in \mathbb{R}. \]

**Interpretation:** does a weighted linear combination of input features exceed a threshold?

\[ \langle \mathbf{w}, \mathbf{x} \rangle = \sum_{i=1}^{d} w_i x_i > t \]

Tie at zero can go either way; we’ll use the following convention:

\[ \text{sign}(z) = \begin{cases} +1 & \text{if } z > 0 \\ -1 & \text{if } z \leq 0. \end{cases} \]
Linear classifiers (for binary classification)

A natural generalization of axis-aligned threshold functions

\[ f_{i,v,t}(x) = \text{sign}(vx_i - t), \quad i \in [d], v \in \{-1, +1\}, t \in \mathbb{R}, \]

are linear threshold functions (or linear classifiers):

\[ f_{w,t}(x) = \text{sign}(\langle w, x \rangle - t), \quad w \in \mathbb{R}^d, t \in \mathbb{R}. \]

**Interpretation:** does a weighted linear combination of input features exceed a threshold?

\[ \langle w, x \rangle = \sum_{i=1}^{d} w_i x_i \quad ? \quad t \]

Tie at zero can go either way; we’ll use the following convention:

\[ \text{sign}(z) = \begin{cases} +1 & \text{if } z > 0 \\ -1 & \text{if } z \leq 0. \end{cases} \]

For now, only considering binary classification, where \( \mathcal{Y} = \{-1, +1\} \).
We’ve seen these before: the (binary) Bayes classifier when class conditional densities are multivariate Gaussians with the same covariance.
A hyperplane in $\mathbb{R}^d$ is a linear subspace of dimension $(d - 1)$.

- A $\mathbb{R}^2$-hyperplane is a line.
- A $\mathbb{R}^3$-hyperplane is a plane.
- As a linear subspace, a hyperplane always contains the origin.

A hyperplane $H$ can be specified by a (non-zero) normal vector.

The hyperplane with normal vector $\mathbf{w} \in \mathbb{R}^d$ is the set

$$H = \{ \mathbf{x} \in \mathbb{R}^d : \langle \mathbf{w}, \mathbf{x} \rangle = 0 \}.$$ 

It becomes oriented if we pick a particular normal vector $\mathbf{w} \in \mathbb{R}^d$. 

Geometric interpretation of linear classifiers
Distance from the hyperplane

- The projection of \( x \) onto the direction of \( w \) has length \( \frac{|\langle w, x \rangle|}{\|w\|_2} \).

- Cosine rule: \( \cos \theta = \frac{\langle w, x \rangle}{\|w\|_2 \|x\|_2} \).

- The distance of \( x \) from the hyperplane is given by \( \frac{|\langle w, x \rangle|}{\|w\|_2} = \|x\|_2 \cdot |\cos \theta| \).

Which side of the hyperplane?

- The cosine satisfies \( \cos \theta > 0 \) iff \( \theta \in (\frac{-\pi}{2}, \frac{\pi}{2}) \).

- We can determine which side of the hyperplane \( H \) that \( x \) is on, using

\[
\text{sign}(\cos \theta) = \text{sign}(\langle w, x \rangle).
\]
Affine hyperplanes

- An **affine hyperplane** $H$ is a hyperplane translated (shifted) by a vector $b$: i.e., $H = b + H'$ for some hyperplane $H'$.
  
  Without loss of generality, $H = bw + H'$, for some hyperplane $H$, $b \geq 0$, and normal vector $w$ for $H'$.
  
- If $b > 0$, naturally oriented by which side contains the origin.

Which side of the affine hyperplane?

- We can determine which side of the affine hyperplane $H$ that $x$ is on using

$$\text{sign}(\langle x, w \rangle - b\|w\|_2^2).$$

Side of affine hyperplane that $x$ is on $\equiv$ linear classification of $x$
Linear classifiers

\[ \text{sign}(\langle \mathbf{w}, x \rangle - b\|\mathbf{w}\|_2^2) \leq 0 \]

\[ \text{sign}(\langle \mathbf{w}, x \rangle - b\|\mathbf{w}\|_2^2) > 0 \]
Even if the Bayes classifier is not a linear classifier, we can hope that it has a good linear approximation.
Even if the Bayes classifier is not a linear classifier, we can hope that it has a good linear approximation.

**Goal:** learning algorithm for linear classifiers with low *excess error*:

\[
\mathbb{E}[\text{err}(f_{\hat{w}, \hat{t}})] - \min_{w, t} \text{err}(f_{w, t})
\]

expected error of your classifier

error of best linear classifier

where \(f_{\hat{w}, \hat{t}}\) is the linear classifier picked by the learning algorithm on the basis of an i.i.d. sample \(S\) from \(P\). (Expectation is over \(S\).)
Learning linear classifiers

Even if the Bayes classifier is not a linear classifier, we can hope that it has a good linear approximation.

**Goal:** learning algorithm for linear classifiers with low *excess error*:

\[
\mathbb{E}[\text{err}(f_{\hat{w}, \hat{t}})] - \min_{w, t} \text{err}(f_{w, t})
\]

where \( f_{\hat{w}, \hat{t}} \) is the linear classifier picked by the learning algorithm on the basis of an i.i.d. sample \( S \) from \( P \). (Expectation is over \( S \).)

A natural approach is "empirical risk minimization" (ERM): find a linear classifier \( f_{w, t} \) with low training error (or *empirical risk*):

\[
\arg \min_{w, t} \text{err}(f_{w, t}, S) = \arg \min_{w, t} \frac{1}{|S|} \sum_{(x, y) \in S} 1\{\text{sign}(\langle w, x \rangle - t) \neq y\}
\]

\[
= \arg \min_{w, t} \frac{1}{|S|} \sum_{(x, y) \in S} 1\{y(\langle w, x \rangle - t) \leq 0\}.
\]
Empirical risk minimization

Unfortunately, this is not possible in general.

The following problem is NP-hard:

Given a set of labeled examples $S$ in $\mathbb{R}^d \times \{\pm 1\}$ with the promise that there is a linear classifier with training error 0.01, find a linear classifier with training error $\leq 0.49$. 

Potential saving grace:

Real-world problems we need to solve do not look like reductions from difficult Satisfiability instances.

Plan:

1. Study the linearly separable instances: where there is a linear classifier with zero training error.
2. Study convex loss functions, which can be efficiently minimized, and how they are related to classification error.
Unfortunately, this is not possible in general.

- The following problem is NP-hard:

  Given a set of labeled examples $S$ in $\mathbb{R}^d \times \{\pm 1\}$ with the promise that there is a linear classifier with training error $0.01$, find a linear classifier with training error $\leq 0.49$.

Potential saving grace:

- Real-world problems we need to solve do not look like reductions from difficult Satisfiability instances.
Empirical risk minimization

Unfortunately, this is not possible in general.

- The following problem is NP-hard:

  Given a set of labeled examples $S$ in $\mathbb{R}^d \times \{\pm 1\}$ with the promise that there is a linear classifier with training error $0.01$, find a linear classifier with training error $\leq 0.49$.

Potential saving grace:

- Real-world problems we need to solve do not look like reductions from difficult \textsc{Satisfiability} instances.

Plan:

1. Study the \textbf{linearly separable} instances: where there is a linear classifier with zero training error.

2. Study \textbf{convex loss functions}, which can be efficiently minimized, and how they are related to classification error.
LINEARLY SEPARABLE INSTANCES
**Easy case: linearly separable data**

Suppose there is a linear classifier with zero training error on $S$: for some $w_\star \in \mathbb{R}^d$ and $\theta \in \mathbb{R}$,

$$y(\langle w_\star, x \rangle - \theta_\star) > 0, \quad \text{for all } (x, y) \in S.$$

In this case, we say the training data is **linearly separable**.
Linear classifiers $f_{w,\theta}$ with $\theta = 0$ are called **homogeneous linear classifiers**.
Linear classifiers $f_{w, \theta}$ with $\theta = 0$ are called **homogeneous linear classifiers**.

**Claim:** There is a mapping $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^{d+1}$ with the following property. For any linear classifier $f_{w, \theta} : \mathbb{R}^d \rightarrow \{\pm 1\}$, there is a homogeneous linear classifier $f_{\tilde{w}, 0} : \mathbb{R}^{d+1} \rightarrow \{\pm 1\}$ such that $f_{w, \theta}(x) = f_{\tilde{w}, 0}(\phi(x))$ for all $x \in \mathbb{R}^d$. 

**Proof:** Let $\phi(x) := (x, 1)$—i.e., add a $(d+1)$-th coordinate that always takes value $1$. For any $w \in \mathbb{R}^d$ and $\theta \in \mathbb{R}$, let $\tilde{w} := (w, -\theta)$. 

Data in $\mathbb{R}^1$

Data in $\mathbb{R}^2$
Linear classifiers $f_{w, \theta}$ with $\theta = 0$ are called **homogeneous linear classifiers**.

**Claim:** There is a mapping $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^{d+1}$ with the following property. For any linear classifier $f_{w, \theta}: \mathbb{R}^d \rightarrow \{\pm 1\}$, there is a homogeneous linear classifier $f_{\tilde{w}, 0}: \mathbb{R}^{d+1} \rightarrow \{\pm 1\}$ such that $f_{w, \theta}(x) = f_{\tilde{w}, 0}(\phi(x))$ for all $x \in \mathbb{R}^d$.

**Proof:** Let $\phi(x) := (x, 1)$—i.e., add a $(d + 1)$-th coordinate that always takes value 1. For any $w \in \mathbb{R}^d$ and $\theta \in \mathbb{R}$, let $\tilde{w} := (w, -\theta)$.
Finding a Homogeneous Linear Separator

Problem: given training data \( S \) in \( \mathbb{R}^d \times \{ \pm 1 \} \), determine whether or not there exists \( w \in \mathbb{R}^d \)

\[ y\langle w, x \rangle > 0, \text{ for all } (x, y) \in S; \]

(and find such a vector if one exists).
Finding a homogeneous linear separator

**Problem**: given training data \( S \) in \( \mathbb{R}^d \times \{\pm 1\} \), determine whether or not there exists \( w \in \mathbb{R}^d \)

\[
y\langle w, x \rangle > 0, \quad \text{for all } (x, y) \in S;
\]

(and find such a vector if one exists).

This is a system of \(|S|\) linear inequalities over \( d \) variables, and hence can be solved in polynomial time using algorithms for **linear programming** (e.g., ellipsoid algorithm, interior point).
**Problem:** given training data $S$ in $\mathbb{R}^d \times \{\pm 1\}$, determine whether or not there exists $w \in \mathbb{R}^d$

$$y\langle w, x \rangle > 0, \quad \text{for all } (x, y) \in S;$$

(and find such a vector if one exists).

This is a system of $|S|$ linear inequalities over $d$ variables, and hence can be solved in polynomial time using algorithms for **linear programming** (e.g., ellipsoid algorithm, interior point).

If one exists, and the inequalities in fact hold with some non-negligible "margin" $\gamma > 0$:

$$y\langle w, x \rangle \geq \gamma, \quad \text{for all } (x, y) \in S;$$

then there is a very simple algorithm that finds a solution: **Perceptron**.