COMS 4771 Lecture 5

1. Linear classifiers.
2. Linearly separable instances.

Announcements

1. Homework 1 due Monday Feb. 9 @ 1 PM.
   - Write-up, as a PDF file.
   - All MATLAB function files and scripts.
   - Everything in a ZIP file; submit on Courseworks.
2. TA office hours:
   - Thursday Feb. 5 @ 6–7 PM cancelled (sorry).
   - Additional TA office hours on Tuesday Feb. 10 @ 6–7 PM.
   - Need help with MATLAB? Your TAs can help! Lots of tips and tricks to be learned . . .

Linear classifiers

Decision tree learning

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

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

where u is some uncertainty measure,

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

and the minimization is over splitting rules of the form

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

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,

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

and the minimization is over splitting rules of the form

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

Axis-aligned threshold functions

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

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

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,

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

and the minimization is over splitting rules of the form

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

Axis-aligned threshold functions

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

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

i.e., looking at classifiers of the form $f_{i,v,t}(\boldsymbol{x}) = \operatorname{sign}(v x_i - t)$.

Linear classifiers (for binary classification)

A natural generalization of axis-aligned threshold functions

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

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

$$f_{\boldsymbol{w},t}(\boldsymbol{x}) = \operatorname{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - t), \quad \boldsymbol{w} \in \mathbb{R}^d,\ t \in \mathbb{R}.$$

Linear classifiers (for binary classification)

A natural generalization of axis-aligned threshold functions

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

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

$$f_{\boldsymbol{w},t}(\boldsymbol{x}) = \operatorname{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - t), \quad \boldsymbol{w} \in \mathbb{R}^d,\ t \in \mathbb{R}.$$

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

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \sum_{i=1}^{d} w_i x_i \ \overset{?}{>} \ t$$

Linear classifiers (for binary classification)

A natural generalization of axis-aligned threshold functions

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

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

$$f_{\boldsymbol{w},t}(\boldsymbol{x}) = \operatorname{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - t), \quad \boldsymbol{w} \in \mathbb{R}^d,\ t \in \mathbb{R}.$$

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

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \sum_{i=1}^{d} w_i x_i \ \stackrel{?}{>}\ t$$

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

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

Linear classifiers (for binary classification)

A natural generalization of axis-aligned threshold functions

$$f_{i,v,t}(\boldsymbol{x}) = \operatorname{sign}(v x_i - t), \quad i \in [d], v \in \{-1,+1\}, t \in \mathbb{R},$$

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

$$f_{\boldsymbol{w},t}(\boldsymbol{x}) = \operatorname{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - t), \quad \boldsymbol{w} \in \mathbb{R}^d,\ t \in \mathbb{R}.$$

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

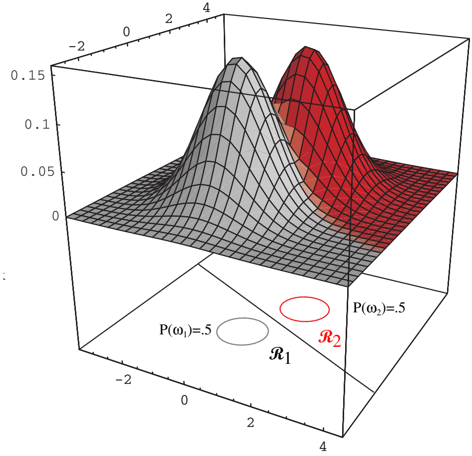
$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \sum_{i=1}^{d} w_i x_i \ \overset{?}{>} \ t$$

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

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

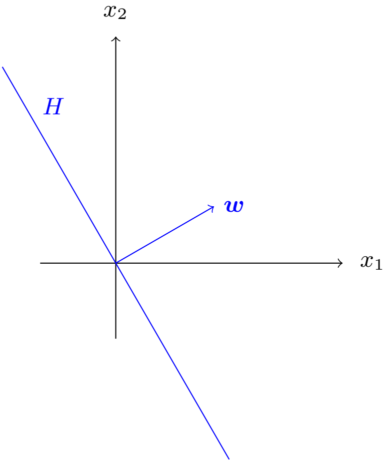
For now, only considering **binary classification**, where $\mathcal{Y} = \{-1,+1\}$.

We've seen these before: the (binary) Bayes classifier when class conditional densities are multivariate Gaussians with the same covariance.



Hyperplanes

Geometric interpretation of linear classifiers



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.

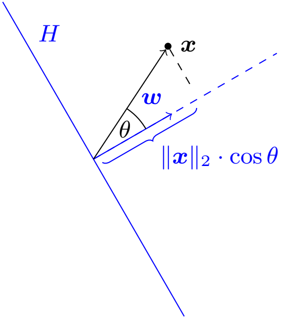
A hyperplane H can be specified by a (non-zero) **normal vector**.

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

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

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

Which side of the hyperplane are we on?



Distance from the hyperplane

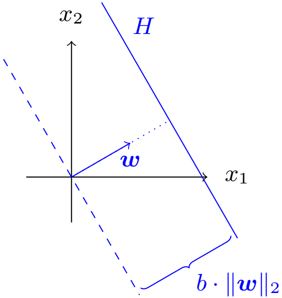
- The projection of $\boldsymbol{x}$ onto the direction of $\boldsymbol{w}$ has length $\dfrac{|\langle \boldsymbol{w}, \boldsymbol{x} \rangle|}{\|\boldsymbol{w}\|_2}$.
- Cosine rule: $\cos\theta = \dfrac{\langle \boldsymbol{w}, \boldsymbol{x} \rangle}{\|\boldsymbol{w}\|_2 \|\boldsymbol{x}\|_2}$.
- The distance of $\boldsymbol{x}$ from the hyperplane is given by $\dfrac{|\langle \boldsymbol{w}, \boldsymbol{x} \rangle|}{\|\boldsymbol{w}\|_2} = \|\boldsymbol{x}\|_2 \cdot |\cos\theta|$.

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 $\boldsymbol{x}$ is on, using

$$\operatorname{sign}(\cos\theta) = \operatorname{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle).$$

Affine hyperplanes



- An **affine hyperplane** H is a hyperplane translated (shifted) by a vector $\boldsymbol{b}$: i.e., $H = \boldsymbol{b} + H'$ for some hyperplane H'.

  Without loss of generality, $H = b\boldsymbol{w} + H'$, for some hyperplane H, $b \geq 0$, and normal vector $\boldsymbol{w}$ for H'.
- If $b > 0$, naturally oriented by which side contains the origin.

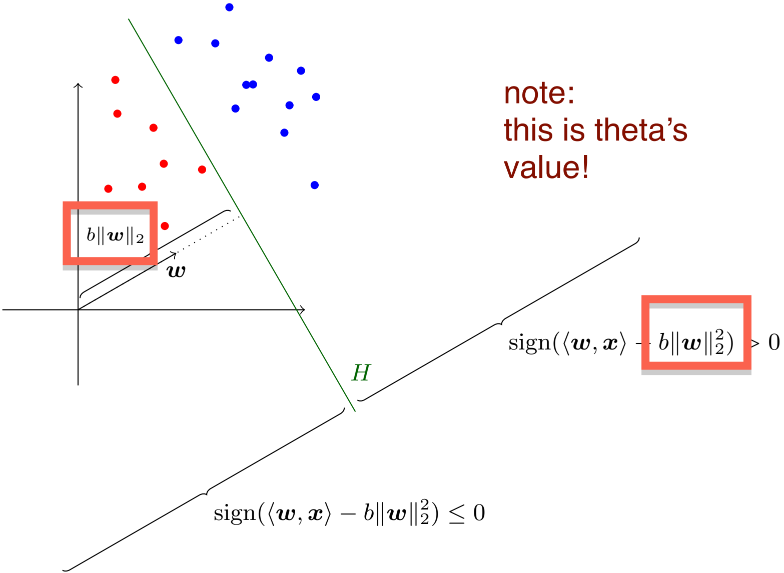
Which side of the affine hyperplane?

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

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

side of affine hyperplane that x is on $\equiv$ linear classification of x

Linear classifiers



note:
this is theta's
value!

Learning linear classifiers

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

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**:

$$\underbrace{\mathbb{E}\big[\mathrm{err}(f_{\hat{\boldsymbol{w}},\hat{t}})\big]}_{\text{expected error of your classifier}} \quad - \quad \underbrace{\min_{\boldsymbol{w},t} \mathrm{err}(f_{\boldsymbol{w},t})}_{\text{error of \textit{best} linear classifier}}$$

where $f_{\hat{\boldsymbol{w}},\hat{t}}$ 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.)

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**:

$$\underbrace{\mathbb{E}\left[\operatorname{err}(f_{\hat{\boldsymbol{w}},\hat{t}})\right]}_{\text{expected error of your classifier}} \quad - \quad \underbrace{\min_{\boldsymbol{w},t} \operatorname{err}(f_{\boldsymbol{w},t})}_{\text{error of \textit{best} linear classifier}}$$

where $f_{\hat{\boldsymbol{w}},\hat{t}}$ 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 $f_{\boldsymbol{w},t}$ with low training error (or **empirical risk**):

$$\begin{aligned} \arg\min_{\boldsymbol{w},t} \operatorname{err}(f_{\boldsymbol{w},b}, S) &= \arg\min_{\boldsymbol{w},t} \frac{1}{|S|} \sum_{(\boldsymbol{x},y)\in S} \mathbb{1}\{\operatorname{sign}(\langle \boldsymbol{w}, \boldsymbol{x}\rangle - t) \neq y\} \\ &= \arg\min_{\boldsymbol{w},t} \frac{1}{|S|} \sum_{(\boldsymbol{x},y)\in S} \boxed{\mathbb{1}\{y(\langle \boldsymbol{w}, \boldsymbol{x}\rangle - t) \leq 0\}.} \end{aligned}$$

Empirical risk minimization

Unforunately, 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 ≤ 0.49.

Unforunately, 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 ≤ 0.49.

Potential saving grace:

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

Unforunately, 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 ≤ 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.

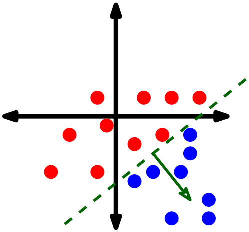
Linearly separable instances

Easy case: linearly separable data

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

$$y(\langle \boldsymbol{w}_\star, \boldsymbol{x} \rangle - \theta) > 0, \quad \text{for all } (x, y) \in S.$$

In this case, we say the training data is **linearly separable**.



Homogeneous linear classification

Linear classifiers $f_{\boldsymbol{w},\theta}$ with $\theta = 0$ are called **homogeneous linear classifiers**.

Homogeneous linear classification

Linear classifiers $f_{\boldsymbol{w},\theta}$ with $\theta = 0$ are called **homogeneous linear classifiers**.

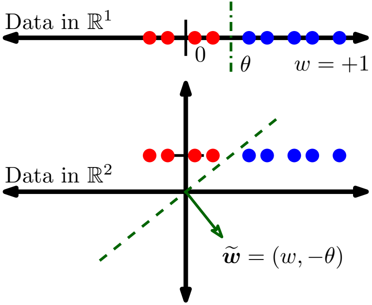
Claim: There is a mapping $\phi \colon \mathbb{R}^d \to \mathbb{R}^{d+1}$ with the following property. For any linear classifier $f_{\boldsymbol{w},\theta} \colon \mathbb{R}^d \to \{\pm 1\}$, there is a homogeneous linear classifier $f_{\tilde{\boldsymbol{w}},0} \colon \mathbb{R}^{d+1} \to \{\pm 1\}$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

Homogeneous linear classification

Linear classifiers $f_{\boldsymbol{w},\theta}$ with $\theta = 0$ are called **homogeneous linear classifiers**.

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

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



Finding a homogeneous linear separator

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

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

(and find such a vector if one exists).

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

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

(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).

Finding a homogeneous linear separator

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

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

larger than 0 means correct predication.

(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" $\gamma > 0$:

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

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

the margin is great!!!