Logistic regression and linear classifiers

COMS 4721 Spring 2018
1. Logistic regression
Distributions over labeled examples

Distribution $P$ of random pair $(X, Y)$ taking values in $\mathcal{X} \times \{0, 1\}$ can be thought of in two parts:

1. Marginal distribution $P_X$ of $X$: $P_X$ is a distribution on $\mathcal{X}$.
2. Conditional distribution $P_{Y|X=x}$ of $Y$ given $X=x$ for each $x \in \mathcal{X}$:
   
   $$P_{Y|X=x} = \text{Bern}(\eta(x)),$$

   where $\eta: \mathcal{X} \rightarrow [0, 1]$ is the conditional probability function $\eta(x) := P(Y=1|X=x)$, $x \in \mathcal{X}$.

The Bayes optimal classifier $f^\star: \mathcal{X} \rightarrow \mathcal{Y}$ is

$$f^\star(x) = \begin{cases} 1 & \text{if } \eta(x) > 1/2, \\ 0 & \text{if } \eta(x) \leq 1/2, \end{cases} x \in \mathcal{X}.$$
Distributions over labeled examples

Distribution $P$ of random pair $(X, Y)$ taking values in $\mathcal{X} \times \{0, 1\}$ can be thought of in two parts:

1. **Marginal distribution** $P_X$ of $X$:
   
   $P_X$ is a distribution on $\mathcal{X}$.
Distributions over labeled examples

Distribution $P$ of random pair $(X,Y)$ taking values in $\mathcal{X} \times \{0, 1\}$ can be thought of in two parts:

1. **Marginal distribution** $P_X$ of $X$:
   $P_X$ is a distribution on $\mathcal{X}$.

2. **Conditional distribution** $P_{Y|X=x}$ of $Y$ given $X = x$ for each $x \in \mathcal{X}$:
   $$P_{Y|X=x} = \text{Bern}(\eta(x)), \quad x \in \mathcal{X},$$
   where $\eta: \mathcal{X} \rightarrow [0, 1]$ is the *conditional probability function*
   $$\eta(x) := \mathbb{P}(Y = 1 \mid X = x), \quad x \in \mathcal{X}.$$
Distributions over labeled examples

Distribution \( P \) of random pair \((X, Y)\) taking values in \( \mathcal{X} \times \{0, 1\} \) can be thought of in two parts:

1. **Marginal distribution** \( P_X \) of \( X \):
   \( P_X \) is a distribution on \( \mathcal{X} \).

2. **Conditional distribution** \( P_{Y|X=x} \) of \( Y \) given \( X = x \) for each \( x \in \mathcal{X} \):
   \[
   P_{Y|X=x} = \text{Bern}(\eta(x)), \quad x \in \mathcal{X},
   \]
   where \( \eta: \mathcal{X} \to [0, 1] \) is the *conditional probability function*
   \[
   \eta(x) := \mathbb{P}(Y = 1 \mid X = x), \quad x \in \mathcal{X}.
   \]

The Bayes optimal classifier \( f^*: \mathcal{X} \to \mathcal{Y} \) is

\[
f^*(x) = \begin{cases} 
1 & \text{if } \eta(x) > 1/2, \\
0 & \text{if } \eta(x) \leq 1/2,
\end{cases} \quad x \in \mathbb{R}.
\]
Logistic regression

Suppose $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{0, 1\}$. A \textit{logistic regression model} is a statistical model where the conditional probability function has a particular form:

$$Y \mid X = x \sim \text{Bern}(\eta_\beta(x)), \quad x \in \mathbb{R}^d,$$

with

$$\eta_\beta(x) := \text{logistic}(x^T \beta), \quad x \in \mathbb{R}^d,$$

and

$$\text{logistic}(z) := \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z}, \quad z \in \mathbb{R}.$$
Logistic regression

Suppose $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{0, 1\}$.
A *logistic regression model* is a statistical model where the conditional probability function has a particular form:

$$ Y \mid \mathbf{X} = \mathbf{x} \sim \text{Bern}(\eta_{\beta}(\mathbf{x})), \quad \mathbf{x} \in \mathbb{R}^d, $$

with

$$ \eta_{\beta}(\mathbf{x}) := \text{logistic}(\mathbf{x}^T \beta), \quad \mathbf{x} \in \mathbb{R}^d, $$

and

$$ \text{logistic}(z) := \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z}, \quad z \in \mathbb{R}. $$
Logistic regression

Suppose $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{0, 1\}$.

A \textit{logistic regression model} is a statistical model where the conditional probability function has a particular form:

$$Y \mid X = x \sim \text{Bern}(\eta_\beta(x)), \quad x \in \mathbb{R}^d,$$

with

$$\eta_\beta(x) := \text{logistic}(x^T \beta), \quad x \in \mathbb{R}^d,$$

and

$$\text{logistic}(z) := \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z}, \quad z \in \mathbb{R}.$$
Logistic regression

Suppose $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{0, 1\}$. A \textit{logistic regression model} is a statistical model where the conditional probability function has a particular form:

$$ Y \mid X = x \sim \text{Bern}(\eta_\beta(x)), \quad x \in \mathbb{R}^d, $$

with

$$ \eta_\beta(x) := \text{logistic}(x^T \beta), \quad x \in \mathbb{R}^d, $$

and

$$ \text{logistic}(z) := \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z}, \quad z \in \mathbb{R}. $$

- Parameters: $\beta = (\beta_1, \ldots, \beta_d) \in \mathbb{R}^d$.

- Conditional distribution of $Y$ given $X$ is Bernoulli; marginal distribution of $X$ not specified.
MLE for logistic regression

Log-likelihood of $\beta$ in iid logistic regression model, given data $(X_i, Y_i) = (x_i, y_i)$ for $i = 1, \ldots, n$:

$$\ln \prod_{i=1}^{n} \eta(\beta(x_i))^{y_i} \left(1 - \eta(\beta(x_i))\right)^{1-y_i} = \sum_{i=1}^{n} y_i \ln \eta(\beta(x_i)) + (1 - y_i) \ln(1 - \eta(\beta(x_i))).$$
Log-likelihood of $\beta$ in iid logistic regression model, given data $(X_i, Y_i) = (x_i, y_i)$ for $i = 1, \ldots, n$:

$$\ln \prod_{i=1}^{n} \eta_{\beta}(x_i)^{y_i} (1 - \eta_{\beta}(x_i))^{1-y_i} = \sum_{i=1}^{n} y_i \ln \eta_{\beta}(x_i) + (1 - y_i) \ln (1 - \eta_{\beta}(x_i)).$$

- No closed-form formula for MLE.
MLE for logistic regression

Log-likelihood of $\beta$ in iid logistic regression model, given data $(X_i, Y_i) = (x_i, y_i)$ for $i = 1, \ldots, n$:

$$\ln \prod_{i=1}^{n} \eta_{\beta}(x_i)^{y_i} (1 - \eta_{\beta}(x_i))^{1-y_i} = \sum_{i=1}^{n} y_i \ln \eta_{\beta}(x_i) + (1 - y_i) \ln(1 - \eta_{\beta}(x_i)).$$

- No closed-form formula for MLE.
- Nevertheless, there are efficient algorithms that obtain an approximate maximizer of the log-likelihood function.
Log-odds function and classifier

In logistic regression model, *log-odds function* is

\[
x \mapsto \ln \frac{\eta_\beta(x)}{1 - \eta_\beta(x)} = \ln \left( \frac{e^{x^\top \beta}}{1 + e^{x^\top \beta}} \right) = x^\top \beta,
\]

i.e., a linear function.\(^1\)

\(^1\)Some authors allow affine function; we can get this using affine expansion.
In logistic regression model, \textit{log-odds function} is

\[
\mathbf{x} \mapsto \ln \frac{\eta_{\beta}(\mathbf{x})}{1 - \eta_{\beta}(\mathbf{x})} = \ln \left( \frac{e^{\mathbf{x}^T \beta}}{1 + e^{\mathbf{x}^T \beta}} \right) = \mathbf{x}^T \beta,
\]

i.e., a linear function.\(^1\)

Bayes optimal classifier in logistic regression model:

\[
\mathbf{x} \mapsto \begin{cases} 
0 & \text{if } \mathbf{x}^T \beta \leq 0, \\
1 & \text{if } \mathbf{x}^T \beta > 0.
\end{cases}
\]

\(^1\)Some authors allow affine function; we can get this using affine expansion.
In logistic regression model, \textit{log-odds function} is
\[
x \mapsto \ln \frac{\eta_\beta(x)}{1 - \eta_\beta(x)} = \ln \left( \frac{e^{x^T\beta}}{1 + e^{x^T\beta}} \right) = x^T\beta,
\]
i.e., a linear function.\(^1\)

Bayes optimal classifier in logistic regression model:
\[
x \mapsto \begin{cases} 
0 & \text{if } x^T\beta \leq 0, \\
1 & \text{if } x^T\beta > 0.
\end{cases}
\]
Such classifiers are called \textit{linear classifiers}.

\(^1\)Some authors allow affine function; we can get this using affine expansion.
2. Linear classifiers
A linear classifier is specified by a weight vector $w \in \mathbb{R}^d$:

$$f_w(x) := \begin{cases} 0 & \text{if } x^T w \leq 0, \\ 1 & \text{if } x^T w > 0. \end{cases}$$
A **linear classifier** is specified by a **weight vector** $\mathbf{w} \in \mathbb{R}^d$:

$$f_{\mathbf{w}}(\mathbf{x}) := \begin{cases} 
0 & \text{if } \mathbf{x}^T \mathbf{w} \leq 0, \\
1 & \text{if } \mathbf{x}^T \mathbf{w} > 0.
\end{cases}$$

**Interpretation**: does a linear combination of input features exceed 0? For $\mathbf{w} = (w_1, \ldots, w_d)$ and $\mathbf{x} = (x_1, \ldots, x_d)$,

$$\mathbf{x}^T \mathbf{w} = \sum_{i=1}^{d} w_i x_i > 0.$$
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 $w \in \mathbb{R}^d$.

The hyperplane with normal vector $w$ is the set of points orthogonal to $w$:

$$H = \left\{ x \in \mathbb{R}^d : x^T w = 0 \right\}.$$
Classification with a hyperplane

Projection of $x$ onto $\text{span}\{w\}$ (a line) has coordinate $\|x\|_2 \cdot \cos(\theta)$ where $\cos(\theta) = \frac{x^T w}{\|w\|_2 \|x\|_2}$.

(Distance to hyperplane is $\|x\|_2 \cdot |\cos(\theta)|$.)

Decision boundary is hyperplane (oriented by $w$):

$x^T w > 0 \iff \|x\|_2 \cdot \cos(\theta) > 0 \iff x$ on same side of $H$ as $w$.

What should we do if we want hyperplane decision boundary that doesn’t (necessarily) go through origin?
Classification with a hyperplane

Projection of $x$ onto $\text{span}\{w\}$ (a line) has coordinate

$$\|x\|_2 \cdot \cos(\theta)$$

where

$$\cos \theta = \frac{x^\top w}{\|w\|_2 \|x\|_2}.$$ 

(Distance to hyperplane is $\|x\|_2 \cdot |\cos(\theta)|$.)
Classification with a hyperplane

Projection of $\mathbf{x}$ onto $\text{span}\{\mathbf{w}\}$ (a line) has coordinate

$$\|\mathbf{x}\|_2 \cdot \cos(\theta)$$

where

$$\cos(\theta) = \frac{\mathbf{x}^T \mathbf{w}}{\|\mathbf{w}\|_2 \|\mathbf{x}\|_2}. \quad (\text{Distance to hyperplane is } \|\mathbf{x}\|_2 \cdot |\cos(\theta)|.)$$

*Decision boundary* is hyperplane (oriented by $\mathbf{w}$):

$$\mathbf{x}^T \mathbf{w} > 0 \iff \|\mathbf{x}\|_2 \cdot \cos(\theta) > 0 \iff \mathbf{x} \text{ on same side of } H \text{ as } \mathbf{w}$$
Classification with a hyperplane

Projection of \( x \) onto \( \text{span}\{w\} \) (a line) has coordinate
\[
\|x\|_2 \cdot \cos(\theta)
\]
where
\[
\cos(\theta) = \frac{x^\top w}{\|w\|_2 \|x\|_2}.
\]

(Distance to hyperplane is \( \|x\|_2 \cdot |\cos(\theta)| \).)

**Decision boundary** is hyperplane (oriented by \( w \)):
\[
x^\top w > 0 \iff \|x\|_2 \cdot \cos(\theta) > 0 \iff x \text{ on same side of } H \text{ as } w
\]

What should we do if we want hyperplane decision boundary that doesn’t (necessarily) go through origin?
Decision boundary with quadratic feature expansion

elliptical decision boundary          hyperbolic decision boundary

Same feature expansions we saw for linear regression models can also be used here to "upgrade" linear classifiers.
Same *feature expansions* we saw for linear regression models can also be used here to “upgrade” linear classifiers.
Learning linear classifiers

Even if Bayes optimal classifier is not linear (in our chosen feature space), we can hope that it has a good linear approximation.
Learning linear classifiers

Even if Bayes optimal classifier is not linear (in our chosen feature space), we can hope that it has a good linear approximation.

**Goal**: learn $\hat{w} \in \mathbb{R}^d$ using iid sample $(X_1, Y_1), \ldots, (X_n, Y_n)$ such that

$$\text{Excess risk}(f_{\hat{w}}) := \underbrace{\mathcal{R}(f_{\hat{w}})}_{\text{risk of your classifier}} - \min_{w \in \mathbb{R}^d} \underbrace{\mathcal{R}(f_w)}_{\text{risk of best linear classifier}}$$

is as small as possible, where the **loss function is zero-one loss**.
Learning linear classifiers

Even if Bayes optimal classifier is not linear (in our chosen feature space), we can hope that it has a good linear approximation.

Goal: learn $\hat{w} \in \mathbb{R}^d$ using iid sample $(X_1, Y_1), \ldots, (X_n, Y_n)$ such that

$$\text{Excess risk}(f_{\hat{w}}) := \mathcal{R}(f_{\hat{w}}) - \min_{w \in \mathbb{R}^d} \mathcal{R}(f_w)$$

is as small as possible, where the loss function is zero-one loss.

Empirical risk minimization (ERM): find $f_w$ with minimum empirical risk

$$\hat{\mathcal{R}}(f_w) := \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{f_w(X_i) \neq Y_i\}.$$
Learning linear classifiers

Even if Bayes optimal classifier is not linear (in our chosen feature space), we can hope that it has a good linear approximation.

**Goal:** learn \( \hat{w} \in \mathbb{R}^d \) using iid sample \((X_1, Y_1), \ldots, (X_n, Y_n)\) such that

\[
\text{Excess risk}(f_{\hat{w}}) := \underbrace{\mathcal{R}(f_{\hat{w}})}_{\text{risk of your classifier}} - \underbrace{\min_{w \in \mathbb{R}^d} \mathcal{R}(f_w)}_{\text{risk of best linear classifier}}
\]

is as small as possible, where the loss function is zero-one loss.

*Empirical risk minimization (ERM):* find \( f_w \) with minimum empirical risk

\[
\hat{\mathcal{R}}(f_w) := \frac{1}{n} \sum_{i=1}^{n} 1 \{f_w(X_i) \neq Y_i\}.
\]

**Theorem:** ERM solution \( f_{\hat{w}} \) satisfies

\[
\mathbb{E} \mathcal{R}(f_{\hat{w}}) \to \min_{w \in \mathbb{R}^d} \mathcal{R}(f_w)
\]

as \( n \to \infty \) at rate \( \sqrt{\frac{d}{n}} \) (and sometimes even faster!).
Unfortunately, this is not possible in general.
Unfortunately, this is not possible in general.

- The following problem is NP-hard:

  **input** \( n \) labeled examples from \( \mathbb{R}^d \times \{0, 1\} \) with promise that there is a linear classifier with empirical risk 0.01.
  
  **output** a linear classifier with empirical risk \( \leq 0.49 \).

(Zero-one loss is very different from squared loss!)
Unfortunately, this is not possible in general.

- The following problem is NP-hard:
  
  **input**  \( n \) labeled examples from \( \mathbb{R}^d \times \{0, 1\} \) with promise that there is a linear classifier with empirical risk 0.01.
  
  **output**  a linear classifier with empirical risk \( \leq 0.49 \).

(Zero-one loss is very different from squared loss!)

**Potential saving grace:**

- Real-world problems we need to solve do not look like the encodings of difficult 3-SAT instances.
3. Linearly separable data and Perceptron
Suppose there is a linear classifier that perfectly classifies the training examples

\[ S := ((x_1, y_1), \ldots, (x_n, y_n)) , \]

i.e., for some \( w_\star \in \mathbb{R}^d \),

\[ f_{w_\star}(x) = y, \quad \text{for all} \ (x, y) \in S. \]

In this case, we say the training data is \textit{linearly separable}. 

\[ \text{Linearly separable data} \]
Finding a linear separator

**Problem:** given training examples $S$ from $\mathbb{R}^d \times \{0, 1\}$, determine whether or not there exists $w \in \mathbb{R}^d$ such that

$$f_w(x) = y, \text{ for all } (x, y) \in S;$$

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

**Problem:** given training examples $S$ from $\mathbb{R}^d \times \{0, 1\}$, determine whether or not there exists $w \in \mathbb{R}^d$ such that

$$f_w(x) = y, \quad \text{for all } (x, y) \in S;$$

(and find such a vector if one exists).

- $d$ variables: $w \in \mathbb{R}^d$
- $|S|$ inequalities: for $(x, y) \in S$,

$$\begin{align*}
\text{if } y = 0: & \quad x^T w \leq 0, \\
\text{if } y = 1: & \quad x^T w > 0.
\end{align*}$$

Can be solved in polynomial time using algorithms for **linear programming**.
Finding a linear separator

**Problem:** given training examples $S$ from $\mathbb{R}^d \times \{0, 1\}$, determine whether or not there exists $w \in \mathbb{R}^d$ such that

$$f_w(x) = y, \quad \text{for all } (x, y) \in S;$$

(and find such a vector if one exists).

- $d$ variables: $w \in \mathbb{R}^d$
- $|S|$ inequalities: for $(x, y) \in S$,

  - if $y = 0$: $x^T w \leq 0$,
  - if $y = 1$: $x^T w > 0$.

Can be solved in polynomial time using algorithms for linear programming.

If one exists, and the inequalities can be satisfied with some non-negligible “wiggle room”, then there is a very simple algorithm that finds a solution.
Perceptron (Rosenblatt, 1958)

Notationally simpler to use $\mathcal{Y} := \{-1, +1\}$ instead of $\{0, 1\}$.

---

**Perceptron**

**input** Labeled examples $S$ from $\mathbb{R}^d \times \{-1, +1\}$.

1. **initialize** $\hat{w}_1 := 0$.
2. **for** $t = 1, 2, \ldots, \text{do}$
3. **if** there is an example in $S$ misclassified by $f_{\hat{w}_t}$ **then**
4. Let $(x_t, y_t)$ be any such misclassified example.
5. **Update**: $\hat{w}_{t+1} := \hat{w}_t + y_t x_t$.
6. **else**
7. **return** $\hat{w}_t$.
8. **end if**
9. **end for**

---

Note 1: An example $(x, y)$ is misclassified by $f_w$ if $y x^T w \leq 0$.

Note 2: If Perceptron terminates, then $f_{\hat{w}_t}$ perfectly classifies the data!
Perceptron  (Rosenblatt, 1958)

Notationally simpler to use $\mathcal{Y} := \{-1, +1\}$ instead of $\{0, 1\}$.

---

**Perceptron**

**input** Labeled examples $S$ from $\mathbb{R}^d \times \{-1, +1\}$.

1: **initialize** $\hat{w}_1 := 0$.
2: **for** $t = 1, 2, \ldots, \text{do}$
3: **if** there is an example in $S$ misclassified by $f_{\hat{w}_t}$ **then**
4: Let $(x_t, y_t)$ be any such misclassified example.
5: Update: $\hat{w}_{t+1} := \hat{w}_t + y_t x_t$.
6: **else**
7: **return** $\hat{w}_t$.
8: **end if**
9: **end for**

**Note 1:** An example $(x, y)$ is misclassified by $f_w$ if $yx^T w \leq 0$. 
Notationally simpler to use $\mathcal{Y} := \{-1, +1\}$ instead of $\{0, 1\}$.

**Perceptron**

**input** Labeled examples $S$ from $\mathbb{R}^d \times \{-1, +1\}$.
1. initialize $\hat{w}_1 := 0$.
2. for $t = 1, 2, \ldots, \text{do}$
3. if there is an example in $S$ misclassified by $f_{\hat{w}_t}$ then
4. Let $(x_t, y_t)$ be any such misclassified example.
5. Update: $\hat{w}_{t+1} := \hat{w}_t + y_t x_t$.
6. else
7. return $\hat{w}_t$.
8. end if
9. end for

**Note 1:** An example $(x, y)$ is misclassified by $f_w$ if $yx^T w \leq 0$.
**Note 2:** If Perceptron terminates, then $f_{\hat{w}_t}$ perfectly classifies the data!
Scenario 1

\[ \mathbf{x}_t \quad \hat{\mathbf{w}}_t \quad y_t = 1 \quad \mathbf{x}_t^\mathsf{T} \hat{\mathbf{w}}_t \leq 0 \]

Current vector \( \hat{\mathbf{w}}_t \) comparable to \( \mathbf{x}_t \) in length.
Scenario 1

Updated vector $\hat{w}_{t+1}$ now correctly classifies $(x_t, y_t)$.
Scenario 2

\[ \hat{\mathbf{w}}_t \]

\[ y_t = 1 \]

\[ \mathbf{x}_t^\top \hat{\mathbf{w}}_t \leq 0 \]

Current vector \( \hat{\mathbf{w}}_t \) much longer than \( \mathbf{x}_t \).
Scenario 2

\[ \hat{w}_{t+1} \]

\[ \hat{w}_t \]

\[ x_t \]

\[ y_t = 1 \]

\[ x_t^T \hat{w}_t \leq 0 \]

Updated vector \( \hat{w}_{t+1} \) does not correctly classify \((x_t, y_t)\).
Updated vector $\hat{w}_{t+1}$ does not correctly classify $(x_t, y_t)$.

Not obvious that Perceptron will eventually terminate!
When is there a lot of “wiggle room”? 

Suppose $w \in \mathbb{R}^d$ satisfies 

$$\min_{(x,y) \in S} yx^T w > 0.$$
When is there a lot of “wiggle room”? 

Suppose \( \boldsymbol{w} \in \mathbb{R}^d \) satisfies

\[
\min_{(x, y) \in S} yx^\top \boldsymbol{w} > 0.
\]

Then so does, e.g., \( \boldsymbol{w}/100 \).
When is there a lot of “wiggle room”?

Suppose $\mathbf{w} \in \mathbb{R}^d$ satisfies

$$\min_{(x, y) \in S} yx^T \mathbf{w} > 0.$$ 

Then so does, e.g., $\mathbf{w}/100$. Let’s fix a particular scaling of $\mathbf{w}$. 

"Theorem": Perceptron converges quickly when there short $\mathbf{w}$ with

$$\min_{(x, y) \in S} yx^T \mathbf{w} \geq 1.$$
When is there a lot of “wiggle room”?  

Suppose \( w \in \mathbb{R}^d \) satisfies 
\[
\min_{(x,y) \in S} yx^T w > 0.
\]

Then so does, e.g., \( w/100 \). Let’s fix a particular scaling of \( w \). Let \((\tilde{x}, \tilde{y})\) be any example in \( S \) that achieves the minimum.
When is there a lot of “wiggle room”?

Suppose \( \mathbf{w} \in \mathbb{R}^d \) satisfies

\[
\min_{(x,y) \in S} yx^\top \mathbf{w} > 0.
\]

Then so does, e.g., \( \mathbf{w}/100 \). Let’s fix a particular scaling of \( \mathbf{w} \).

Let \((\tilde{x}, \tilde{y})\) be any example in \( S \) that achieves the minimum.

Let’s rescale \( \mathbf{w} \) so that \( \tilde{y}\tilde{x}^\top \mathbf{w} = 1 \).

Now distance from \( \tilde{y}\tilde{x} \) to \( H \) is \( \|\mathbf{w}\|_2 \).

This distance is called the margin.

Therefore, if \( \mathbf{w} \) is a short vector satisfying \( \min_{(x,y) \in S} yx^\top \mathbf{w} \geq 1 \), then it corresponds to a linear separator with large margin on examples in \( S \).
When is there a lot of “wiggle room”?

Suppose \( \mathbf{w} \in \mathbb{R}^d \) satisfies

\[
\min_{(x,y) \in S} yx^T \mathbf{w} > 0.
\]

Then so does, e.g., \( \mathbf{w}/100 \). Let’s fix a particular scaling of \( \mathbf{w} \).

Let \((\tilde{x}, \tilde{y})\) be any example in \( S \) that achieves the minimum.

- Rescale \( \mathbf{w} \) so that \( \tilde{y}\tilde{x}^T \mathbf{w} = 1 \).
  (Now scaling is fixed.)
When is there a lot of “wiggle room”?

Suppose \( w \in \mathbb{R}^d \) satisfies

\[
\min_{(x, y) \in S} yx^Tw > 0.
\]

Then so does, e.g., \( w/100 \). Let’s fix a particular scaling of \( w \).

Let \((\tilde{x}, \tilde{y})\) be any example in \( S \) that achieves the minimum.

- Rescale \( w \) so that \( \tilde{y}\tilde{x}^Tw = 1 \).
  (Now scaling is fixed.)
- Now distance from \( \tilde{y}\tilde{x} \) to \( H \) is \( \frac{1}{\|w\|_2} \).
  This distance is called the margin.
When is there a lot of “wiggle room”? 

Suppose \( \mathbf{w} \in \mathbb{R}^d \) satisfies

\[
\min_{(x,y) \in S} y \mathbf{x}^T \mathbf{w} > 0.
\]

Then so does, e.g., \( \mathbf{w}/100 \). Let’s fix a particular scaling of \( \mathbf{w} \).

Let \((\tilde{x}, \tilde{y})\) be any example in \( S \) that achieves the minimum.

- Rescale \( \mathbf{w} \) so that \( \tilde{y} \tilde{x}^T \mathbf{w} = 1 \).
  (Now scaling is fixed.)
- Now distance from \( \tilde{y} \tilde{x} \) to \( H \) is \( \frac{1}{\|\mathbf{w}\|_2} \).

  This distance is called the \textit{margin}.

Therefore, if \( \mathbf{w} \) is a short vector satisfying

\[
\min_{(x,y) \in S} y \mathbf{x}^T \mathbf{w} \geq 1,
\]

then it corresponds to a linear separator with large margin on examples in \( S \).
When is there a lot of “wiggle room”?

Suppose \( \mathbf{w} \in \mathbb{R}^d \) satisfies
\[
\min_{(x,y) \in S} yx^T \mathbf{w} > 0.
\]
Then so does, e.g., \( \mathbf{w}/100 \). Let’s fix a particular scaling of \( \mathbf{w} \).
Let \((\tilde{x}, \tilde{y})\) be any example in \( S \) that achieves the minimum.

- Rescale \( \mathbf{w} \) so that \( \tilde{y}\tilde{x}^T \mathbf{w} = 1 \).
  (Now scaling is fixed.)
- Now distance from \( \tilde{y}\tilde{x} \) to \( H \) is \( \frac{1}{\|\mathbf{w}\|_2} \).
  This distance is called the margin.

Therefore, if \( \mathbf{w} \) is a short vector satisfying
\[
\min_{(x,y) \in S} yx^T \mathbf{w} \geq 1,
\]
then it corresponds to a linear separator with large margin on examples in \( S \).

"Theorem": Perceptron converges quickly when there short \( \mathbf{w} \) with \( \min_{(x,y) \in S} yx^T \mathbf{w} \geq 1 \).
Sneak peek: maximum margin linear classifier

We can obtain the linear separator with the largest margin by finding the *shortest* vector $\mathbf{w}$ such that

$$\min_{(x,y) \in S} yx^T \mathbf{w} \geq 1.$$
We can obtain the linear separator with the largest margin by finding the *shortest* vector $w$ such that

$$
\min_{(x,y) \in S} yx^T w \geq 1.
$$

This can be expressed as a mathematical optimization problem that is the basis of *support vector machines*.

(More on this in next lecture.)
4. Online Perceptron
Online Perceptron

If data is not linearly separable, Perceptron runs forever!
Online Perceptron

If data is not linearly separable, Perceptron runs forever!

**Alternative:** consider each example once, then stop.

---

**Online Perceptron**

**input** Labeled examples \(((x_i, y_i))_{i=1}^{n}\) from \(\mathbb{R}^d \times \{-1, +1\}\).

1: initialize \(\hat{w}_1 := 0\).
2: for \(t = 1, 2, \ldots, n\) do
3: \hspace{1em} if \(y_t \langle \hat{w}_t, x_t \rangle \leq 0\) then
4: \hspace{2em} \(\hat{w}_{t+1} := \hat{w}_t + y_t x_t\).
5: \hspace{1em} else
6: \hspace{2em} \(\hat{w}_{t+1} := \hat{w}_t\)
7: \hspace{1em} end if
8: end for
9: return \(\hat{w}_{n+1}\).
If data is not linearly separable, Perceptron runs forever!

**Alternative:** consider each example once, then stop.

---

**Online Perceptron**

**input** Labeled examples $((x_i, y_i))_{i=1}^n$ from $\mathbb{R}^d \times \{-1, +1\}$.

1. **initialize** $\hat{w}_1 := 0$.
2. **for** $t = 1, 2, \ldots, n$ **do**
3. **if** $y_t \langle \hat{w}_t, x_t \rangle \leq 0$ **then**
4. $\hat{w}_{t+1} := \hat{w}_t + y_t x_t$.
5. **else**
6. $\hat{w}_{t+1} := \hat{w}_t$
7. **end if**
8. **end for**
9. **return** $\hat{w}_{n+1}$.

Final classifier $f_{\hat{w}_{n+1}}$ is not necessarily a linear separator (even if one exists!).
Online learning

*Online learning algorithms:*

Go through examples \((x_1, y_1), (x_2, y_2), \ldots\) one-by-one.
Online learning

*Online learning algorithms:*

Go through examples \((x_1, y_1), (x_2, y_2), \ldots\) one-by-one.

- Before seeing \((x_t, y_t)\), learner has a “current” classifier \(\hat{f}_t\) in hand.

Typically, update is very computationally cheap to compute.

"Theorem": If there is a short \(w\) with \(\min_t y_t x_t^T w \geq 1\), then Online Perceptron makes a small number of mistakes.
**Online learning algorithms:**

Go through examples \((x_1, y_1), (x_2, y_2), \ldots \) one-by-one.

- Before seeing \((x_t, y_t)\), learner has a “current” classifier \(\hat{f}_t\) in hand.
- Upon seeing \(x_t\), learner makes a prediction: \(\hat{y}_t := \hat{f}_t(x_t)\).
Online learning

**Online learning algorithms:**

Go through examples \((x_1, y_1), (x_2, y_2), \ldots\) one-by-one.

- Before seeing \((x_t, y_t)\), learner has a “current” classifier \(\hat{f}_t\) in hand.
- Upon seeing \(x_t\), learner makes a prediction: \(\hat{y}_t := \hat{f}_t(x_t)\).
- If \(\hat{y}_t \neq y_t\), then the prediction was a “mistake”.

"Theorem": If there is a short \(w\) with \(\min_t y_t x^T_t w \geq 1\), then Online Perceptron makes a small number of mistakes.
Online learning

**Online learning algorithms:**

Go through examples \((x_1, y_1), (x_2, y_2), \ldots\) one-by-one.

- Before seeing \((x_t, y_t)\), learner has a “current” classifier \(\hat{f}_t\) in hand.
- Upon seeing \(x_t\), learner makes a prediction: \(\hat{y}_t := \hat{f}_t(x_t)\).
- If \(\hat{y}_t \neq y_t\), then the prediction was a “mistake”.
- See correct label \(y_t\); update \(\hat{f}_t\) to get \(\hat{f}_{t+1}\).

Typically, update is **very computationally cheap** to compute.
Online learning

**Online learning algorithms:**

Go through examples \((x_1, y_1), (x_2, y_2), \ldots\) one-by-one.

- Before seeing \((x_t, y_t)\), learner has a “current” classifier \(\hat{f}_t\) in hand.
- Upon seeing \(x_t\), learner makes a prediction: \(\hat{y}_t := \hat{f}_t(x_t)\).
- If \(\hat{y}_t \neq y_t\), then the prediction was a “mistake”.
- See correct label \(y_t\); update \(\hat{f}_t\) to get \(\hat{f}_{t+1}\).

Typically, update is very computationally cheap to compute.

**“Theorem”:** If there is a short \(w\) with \(\min_t y_t x_t^\top w \geq 1\), then Online Perceptron makes a small number of mistakes.
What good is a small mistake bound?

- **Sequence of classifiers** \( \hat{f}_1, \hat{f}_2, \ldots \) is highly accurate (on average) in predicting labels of iid sequence \((X_1, Y_1), (X_2, Y_2), \ldots\)

<table>
<thead>
<tr>
<th>(X_1)</th>
<th>(X_2)</th>
<th>(X_3)</th>
<th>(X_4)</th>
<th>(X_5)</th>
<th>(X_6)</th>
<th>(X_7)</th>
<th>(X_8)</th>
<th>(X_9)</th>
<th>(X_{10})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\hat{f}_1)</td>
<td>(\hat{f}_2)</td>
<td>(\hat{f}_3)</td>
<td>(\hat{f}_4)</td>
<td>(\hat{f}_5)</td>
<td>(\hat{f}_6)</td>
<td>(\hat{f}_7)</td>
<td>(\hat{f}_8)</td>
<td>(\hat{f}_9)</td>
<td>(\hat{f}_{10})</td>
</tr>
<tr>
<td>(\hat{y}_1)</td>
<td>(\hat{y}_2)</td>
<td>(\hat{y}_3)</td>
<td>(\hat{y}_4)</td>
<td>(\hat{y}_5)</td>
<td>(\hat{y}_6)</td>
<td>(\hat{y}_7)</td>
<td>(\hat{y}_8)</td>
<td>(\hat{y}_9)</td>
<td>(\hat{y}_{10})</td>
</tr>
<tr>
<td>(Y_1)</td>
<td>(Y_2)</td>
<td>(Y_3)</td>
<td>(Y_4)</td>
<td>(Y_5)</td>
<td>(Y_6)</td>
<td>(Y_7)</td>
<td>(Y_8)</td>
<td>(Y_9)</td>
<td>(Y_{10})</td>
</tr>
</tbody>
</table>
What good is a small mistake bound?

- **Sequence of classifiers** \( \hat{f}_1, \hat{f}_2, \ldots \) is highly accurate (on average) in predicting labels of iid sequence \((X_1, Y_1), (X_2, Y_2), \ldots\)

- Combine classifiers \( \hat{f}_1, \hat{f}_2, \ldots \) into a **single, highly accurate classifier** \( \hat{f} \).
What good is a small mistake bound?

- **Sequence of classifiers** \( \hat{f}_1, \hat{f}_2, \ldots \) is highly accurate (on average) in predicting labels of iid sequence \((X_1, Y_1), (X_2, Y_2), \ldots\)

- Combine classifiers \( \hat{f}_1, \hat{f}_2, \ldots \) into a **single, highly accurate classifier** \( \hat{f} \).

This is achieved via *online-to-batch conversion*. 
Online-to-batch conversion

- Run online learning algorithm on sequence of examples

$$(x_1, y_1), \ldots, (x_n, y_n)$$

(in random order) to produce sequence of binary classifiers

$$\hat{f}_1, \ldots, \hat{f}_{n+1},$$

$$\hat{f}_i : X \to \{\pm 1\}.$$
Online-to-batch conversion

- Run online learning algorithm on sequence of examples
\[(x_1, y_1), \ldots, (x_n, y_n)\]
(in random order) to produce sequence of binary classifiers
\[\hat{f}_1, \ldots, \hat{f}_{n+1},\]
\[(\hat{f}_i : \mathcal{X} \to \{\pm 1\}).\]

- Final classifier: majority vote over the \(\hat{f}_i\)’s
\[\hat{f}(x) := \begin{cases} -1 & \text{if } \sum_{i=1}^{n+1} \hat{f}_i(x) \leq 0, \\ +1 & \text{if } \sum_{i=1}^{n+1} \hat{f}_i(x) > 0. \end{cases} \]
Online-to-batch conversion

Run online learning algorithm on sequence of examples

\[(x_1, y_1), \ldots, (x_n, y_n)\]

(in random order) to produce sequence of binary classifiers

\[\hat{f}_1, \ldots, \hat{f}_{n+1},\]

\[(\hat{f}_i : \mathcal{X} \rightarrow \{\pm 1\}).\]

Final classifier: majority vote over the \(\hat{f}_i\)'s

\[
\hat{f}(\mathbf{x}) := \begin{cases} 
-1 & \text{if } \sum_{i=1}^{n+1} \hat{f}_i(\mathbf{x}) \leq 0, \\
+1 & \text{if } \sum_{i=1}^{n+1} \hat{f}_i(\mathbf{x}) > 0.
\end{cases}
\]

There are many variants of online-to-batch conversions that make sense!
Practical suggestions for online-to-batch variant

▶ Run online learning algorithm to make a few (e.g., two) passes over training examples, each time in a different random order.

▶ For linear classifiers, instead of using majority vote to combine, just average weight vectors.

▶ Don't use weight vectors from first pass through data. (Weight vectors at start are rubbish anyway.)
Practical suggestions for online-to-batch variant

- Run online learning algorithm to make a few (e.g., two) passes over training examples, each time in a different random order.
Practical suggestions for online-to-batch variant

- Run online learning algorithm to make a few (e.g., two) passes over training examples, each time in a different random order.

- For linear classifiers, instead of using majority vote to combine, just average weight vectors.
Practical suggestions for online-to-batch variant

- Run online learning algorithm to make a few (e.g., two) passes over training examples, each time in a different random order.
- For linear classifiers, instead of using majority vote to combine, just average weight vectors.
- Don't use weight vectors from first pass through data. (Weight vectors at start are rubbish anyway.)
Example: restaurant review classification

Data

- 1M reviews of Pittsburgh restaurants (from Yelp).
- \( \mathcal{Y} = \{ \text{at least 4 stars, below 4 stars} \} \)
  (66.2% have at least 4 (out of 5) stars.)
- Reviews represented as bags-of-words:
  e.g., \( x_{\text{great}} = \# \) times “great” appears in review.
  \( (d \approx 200000 \text{ unique words.}) \)
Example: restaurant review classification

Data

- 1M reviews of Pittsburgh restaurants (from Yelp).
- $\mathcal{Y} = \{ \text{at least 4 stars, below 4 stars} \}$
  
  (66.2% have at least 4 (out of 5) stars.)
- Reviews represented as bags-of-words:
  
  e.g., $x_{\text{great}} = \#$ times “great” appears in review.
  
  ($d \approx 200000$ unique words.)

Results

Online Perceptron + affine expansion:

- Training risk: 23.3%.
- Test risk (on 320123 held-out test examples): 23.4%.
Example: restaurant review classification

Data

- 1M reviews of Pittsburgh restaurants (from Yelp).
- $\mathcal{Y} = \{\text{at least 4 stars, below 4 stars}\}$
  
  (66.2% have at least 4 (out of 5) stars.)

- Reviews represented as bags-of-words:
  e.g., $x_{\text{great}} = \# \text{ times "great" appears in review.}$
  $(d \approx 200000 \text{ unique words.})$

Results

Online Perceptron + affine expansion:

- Training risk: 23.3%.
- Test risk (on 320123 held-out test examples): 23.4%.

Online Perceptron + affine expansion + online-to-batch variant:

- Training risk: 10.1%.
- Test risk (on 320123 held-out test examples): 10.5%.

Running time: $\approx 1 \text{ minute on single Intel Xeon 2.30GHz CPU.}$
1. Logistic regression model; structure/geometry of linear classifiers.
2. Empirical risk minimization for linear classifiers and intractability.
3. Linear separability; two approaches to find a linear separator.
4. Online Perceptron; online-to-batch conversion via voting.