Perceptron (recap), Logistic Regression

Sriram Sankararaman

The instructor gratefully acknowledges Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Outline

1. Review of previous lecture
   - The perceptron
2. Convergence of the Perceptron Learning Algorithm
3. Variants of perceptron
4. Logistic regression

Perceptron learning

Special case: binary classification

- Instance (feature vectors): $\underline{\underline{\boldsymbol{x}}} \in \mathbb{R}^{\mathsf{D}}$
- Label: $\underline{y \in \{-1, +1\}}$
- Model/Hypotheses:
  $H = \{h | h : \mathbb{X} \rightarrow \{-1, +1\}, h(\boldsymbol{x}) = sign(\underline{\sum_{d=1}^{D} w_d x_d \boxed{+ b}})\}$.
- Learning goal: $y = \underline{\underline{h(\boldsymbol{x})}}$

Perceptron learning

Special case: binary classification

- Instance (feature vectors): $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$
- Label: $y \in \{-1, +1\}$
- Model/Hypotheses:
  $H = \{h | h : \mathbb{X} \to \{-1, +1\}, h(\boldsymbol{x}) = sign(\sum_{d=1}^{D} w_d x_d + b)\}.$
- Learning goal: $y = h(\boldsymbol{x})$
  - Learn $\underline{w_1, \ldots, w_D}, b.$
  - Parameters: $w_1, \ldots, w_D, b.$
  - $\boldsymbol{w}$: weights, b: bias

Perceptron predict

- Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$, $\boldsymbol{w} \in \mathbb{R}^{\mathsf{D}}$, $b \in \mathbb{R}$.

$$a = \sum_{d=1}^{D} w_d x_d + b = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$$

$$\hat{y} = sign(a)$$

- Output: $\hat{y}$.
- $\sum_{d=1}^{D} w_d x_d + b$: hyperplane in D dimensions with parameters $(\boldsymbol{w}, b)$.
- $\boldsymbol{w}$: weights, b: bias
- a: activation
- $sign(\sum_{d=1}^{D} w_d x_d + b)$: Linear Threshold Unit (LTU)

Perceptron learning

Iteratively solving one case at a time

- REPEAT
- Pick a data point $\boldsymbol{x}_n$
- Compute $\underline{a = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n}$ using the *current* $\boldsymbol{w}$
- If $\underline{ay_n > 0,}$ do nothing. Else,

$$\underline{\boldsymbol{w} \leftarrow \boldsymbol{w} + y_n\boldsymbol{x}_n}$$

- UNTIL converged.

Properties of perceptron learning

- This is an online algorithm – looks at one instance at a time.
- Convergence
  - If training data is not linearly separable, the algorithm does not converge.
  - If the training data is linearly separable, the algorithm stops in a finite number of steps (converges).
- How long to convergence ?
  - Depends on the difficulty of the problem.

Outline

1. Review of previous lecture
2. Convergence of the Perceptron Learning Algorithm
3. Variants of perceptron
4. Logistic regression

Convergence

Convergence theorem

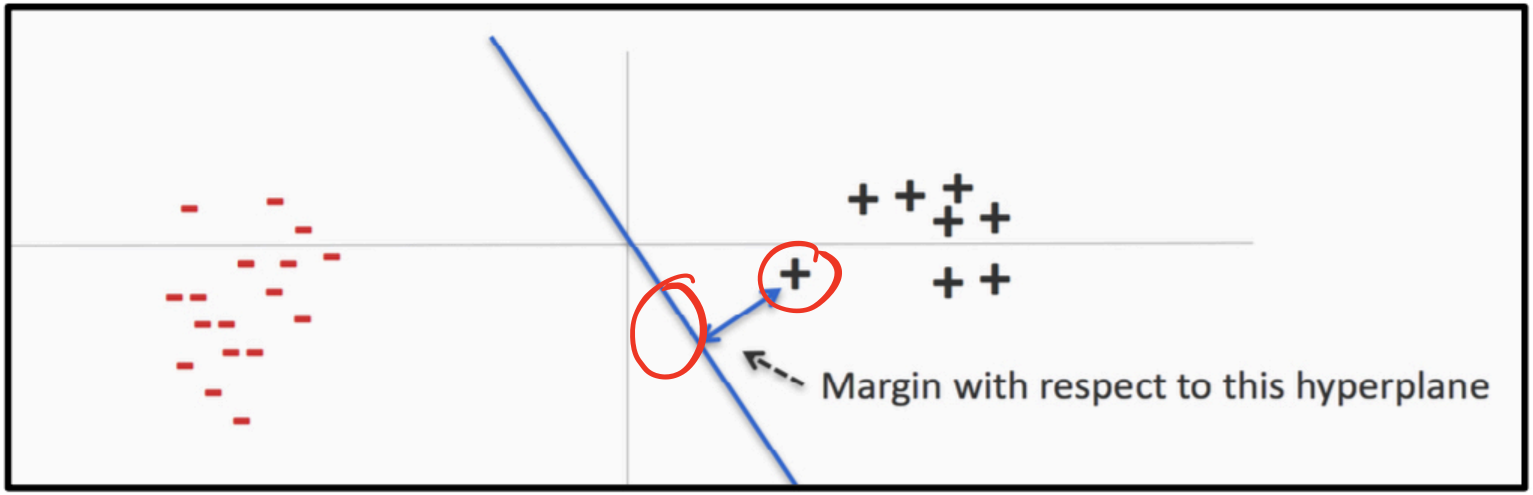
- If the data is linearly separable, the perceptron algorithm will converge after making mistakes that depend on the difficulty of the problem (margin).

Margin

$$\text{Margin}(\mathcal{D}, \boldsymbol{w}) = \begin{cases} min_{(\boldsymbol{x},y)\in\mathcal{D}} y_n \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n & \text{for a separating hyperplane } \boldsymbol{w} \\ -\infty & \text{else} \end{cases}$$

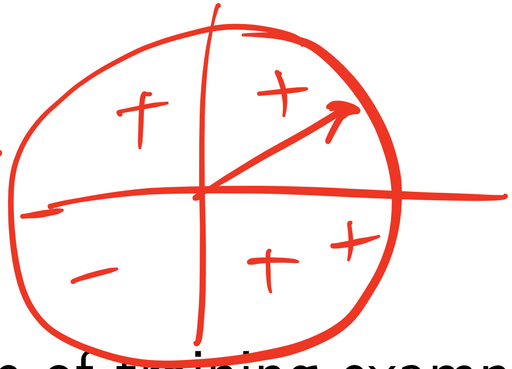
$$\text{Margin}(\mathcal{D}) = sup_{\boldsymbol{w}} \text{Margin}(\mathcal{D}, \boldsymbol{w})$$

Margin



The Mistake Bound Theorem (Novikoff 1962, Block 1962)

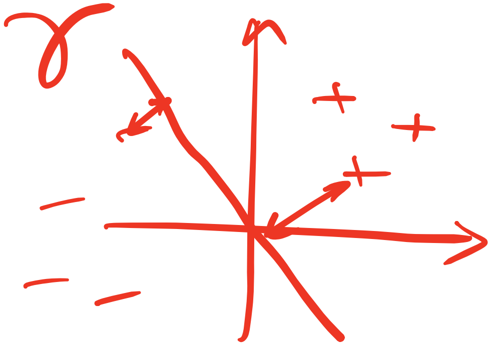
$\|x_n\|_2 \le R$

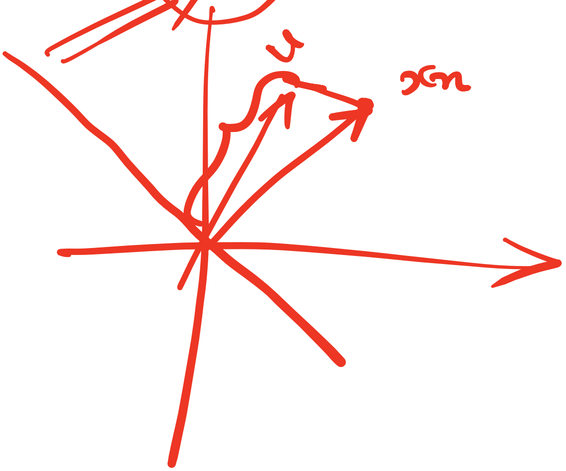


- Let $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_\mathsf{N}, y_\mathsf{N})\}$ be a sequence of training examples such that $\|\boldsymbol{x}_n\|_2 \leq R$ and label $y_n \in \{-1, +1\}$.
- Suppose there exists a unit vector $\boldsymbol{u} \in \mathbb{R}^D$ such that for some $\gamma > 0$, we have $y_n \boldsymbol{u}^\mathrm{T} \boldsymbol{x}_n \geq \gamma$.
- Then the Perceptron algorithm will make at most $\frac{R^2}{\gamma^2}$ mistakes on the training sequence.

$y_n u^T x_n \geq \gamma$

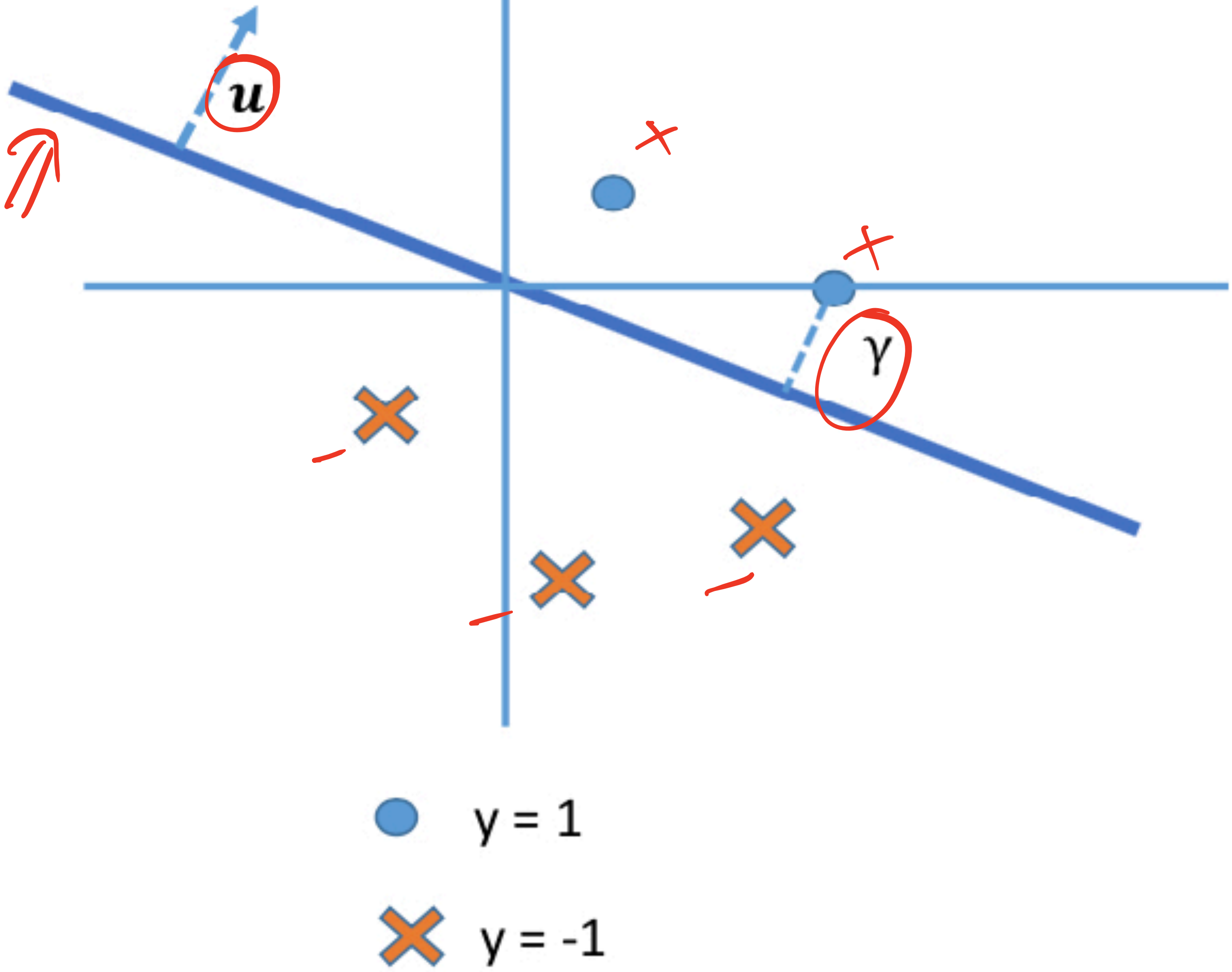
for all $n = 1..N$



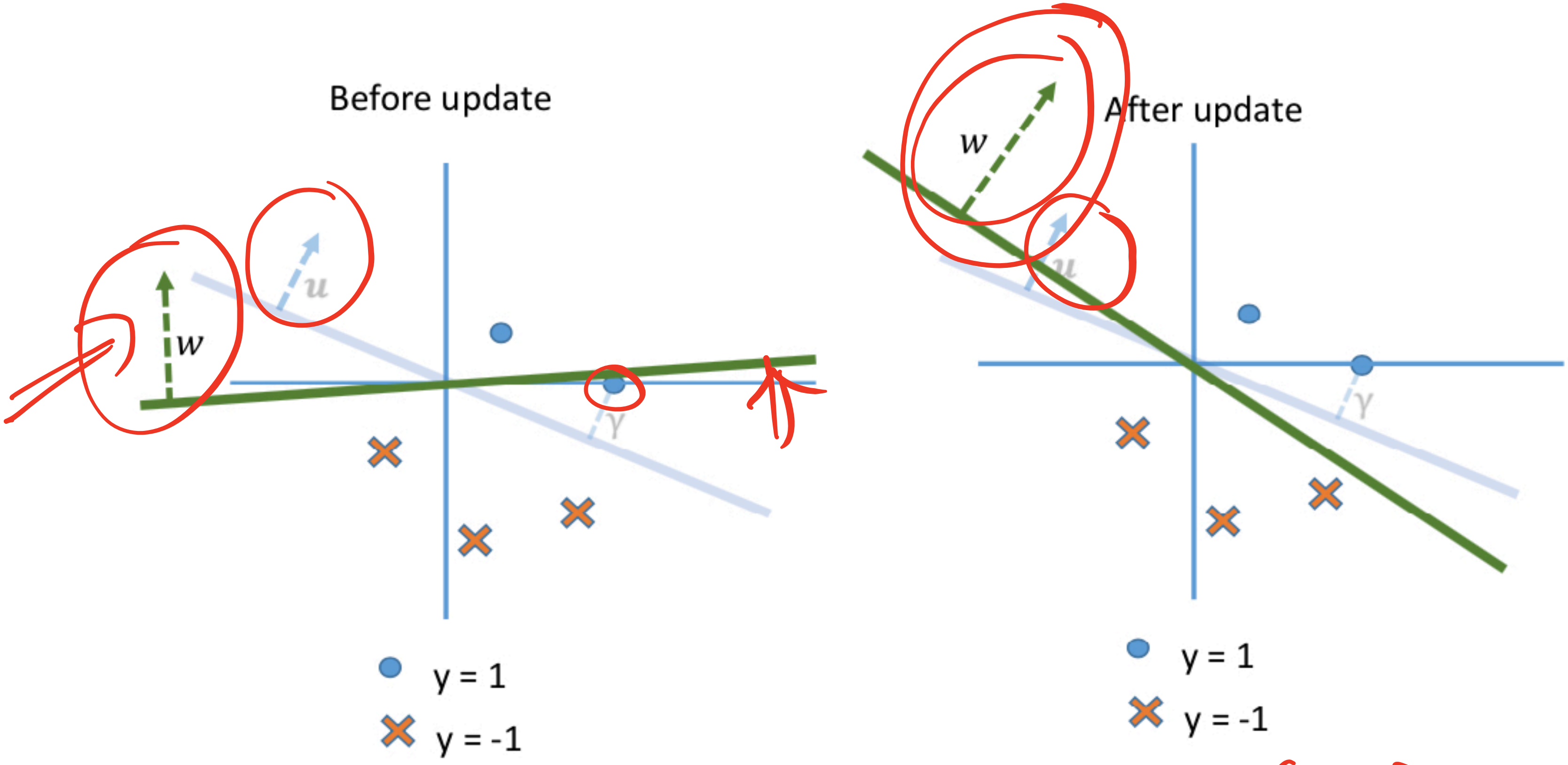


Intuition

$y_n u^T x_n \geq \gamma$

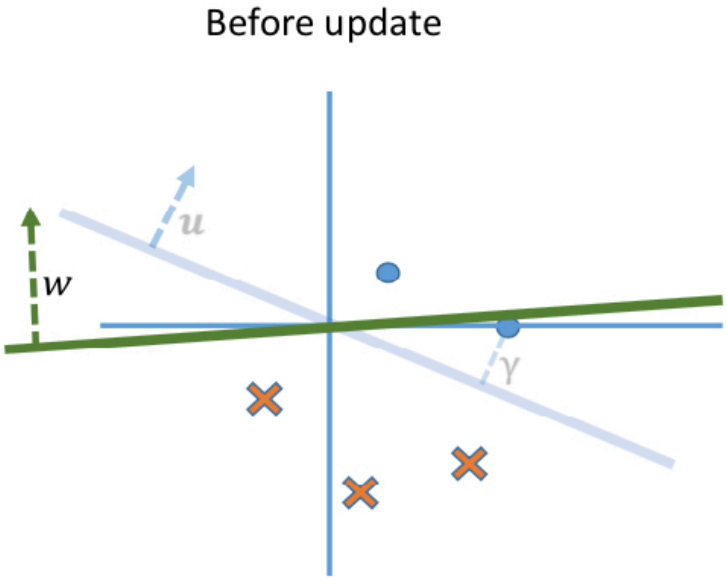


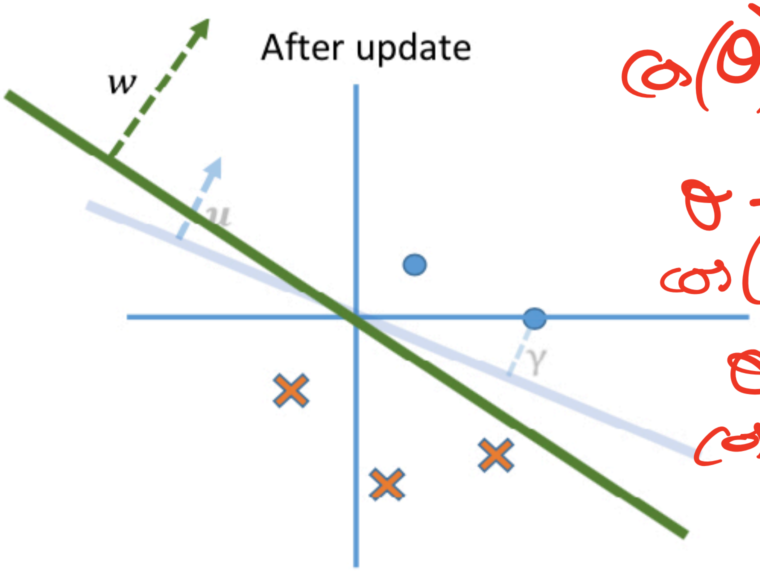
Intuition



angle (w,u)
is smaller after update

Intuition





θ → 0

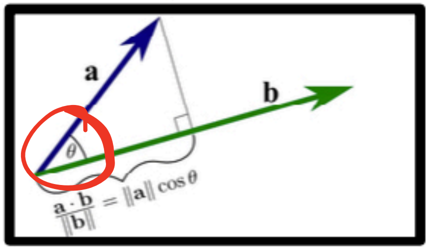
cos(θ) → 1

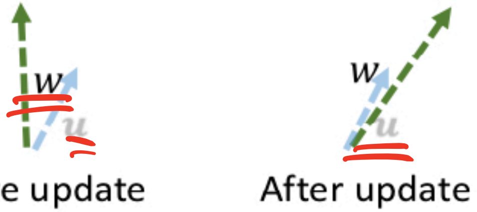
θ → 90°

cos(θ) → 0

θ → 180

cos(θ) → -1


a
b
θ
$\frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{b}\|} = \|\mathbf{a}\| \cos\theta$



a = w

b = u

$$a^T b = \|a\|_2 \|b\|_2 \cos(\theta)$$

$$w^T u = \|w\|_2 \cancel{\|u\|_2} \cos(\theta)$$

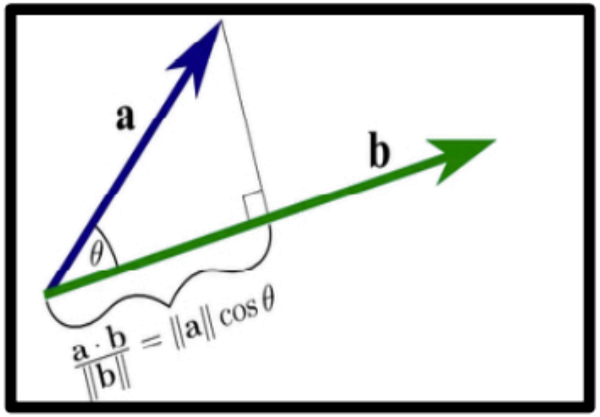
Intuition

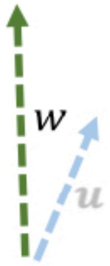
w_t w_{t+1}

- After update, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_{t+1}$ is larger than $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t$. $\|w\|_2 \cos(\theta_t)$
  - After t mistakes, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$.

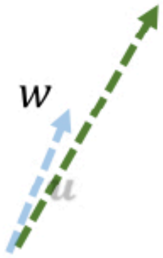
$= u^T w_t$

- The size of $\|\boldsymbol{w}_{t+1}\|$ may increase but not too much.
  - After t mistakes, $\|\boldsymbol{w}_t\|^2 \leq tR^2$.


a
b
θ
$\frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{b}\|} = \|\mathbf{a}\| \cos\theta$



Before update



After update

Proof (Preliminaries)

Setting

- Initial weight vector $\boldsymbol{w}_0 = \mathbf{0}$.
- All training examples are contained in a ball of size R. $\|\boldsymbol{x}_n\| \leq R$.
- The training data is separable by a margin γ using a unit vector $\boldsymbol{u}$. $y_n \boldsymbol{u}^{\mathrm{T}} \boldsymbol{x}_n \geq \gamma$.

Proof (1/3)

Claim 1: After t mistakes, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$.

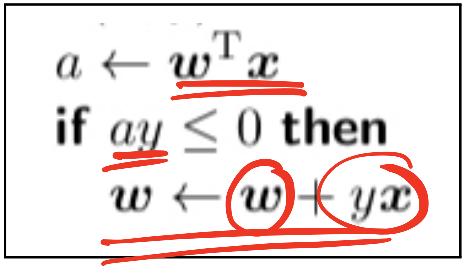
$$\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_{t+1} = \boldsymbol{u}^{\mathrm{T}}(\boldsymbol{w}_t + y_n\boldsymbol{x}_n)$$
$$\geq \boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t + \gamma$$

$u^T w_t \geq u^T w_{t-1} + \gamma$

Because $\boldsymbol{w}_0 = 0$, simple induction gives us: $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$.

$u^T w_{t+1} \geq u^T w_{t-1} + 2\gamma$

$\geq u^T w_0 + (t+1)\gamma$



```
$a \leftarrow \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$
if $ay \leq 0$ then
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$
```

$w_{t+1} = w_t + y_n x_n$

$u^T w_{t+1} \quad ? \quad u^T w_t$

$u^T w_{t+1} = u^T(w_t + y_n x_n)$

$= u^T w_t + u^T(y_n x_n)$

$\geq u^T w_t + \gamma$

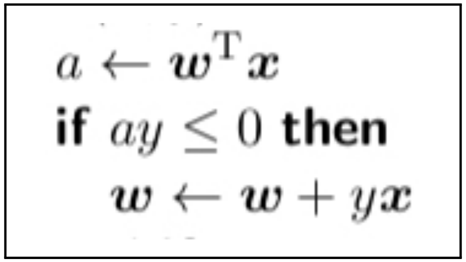
$y_n u^T x_n \geq \gamma$

Proof (1/3)

Claim 1: After t mistakes, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$.

$$\begin{aligned}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_{t+1} &= \boldsymbol{u}^{\mathrm{T}}(\boldsymbol{w}_t + y_n\boldsymbol{x}_n)\\ &\geq \boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t + \gamma\end{aligned}$$

Because $\boldsymbol{w}_0 = 0$, simple induction gives us: $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$.



```
$a \leftarrow \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$
if $ay \leq 0$ then
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$
```

The inner product between the true underlying model and the current model is non-decreasing after each update. This could be because the directions of $\boldsymbol{w}$ and $\boldsymbol{u}$ align or because the length of $\boldsymbol{w}$ increases.

Proof (2/3)

$w_t^T(y_n x_n) = \underset{\text{True}}{y_n}\,\underset{\text{Prediction}}{(w_t^T x_n)} \le 0$

$\|x\|_2^2 = x^T x$

$w_0 = 0$

Claim 2: After t mistakes, $\|w_t\|^2 \le tR^2$

$$\|w_{t+1}\|^2 = \|w_t + y_n x_n\|^2$$
$$= \|w_t\|^2 + 2 w_t^T (y_n x_n) + \|y_n x_n\|^2$$

$\|w_{t+1}\|_2^2 \le$ ~~$\|w_0\|_2^2$~~ $+ (t+1)R^2$

$(y_n x_n)^T (y_n x_n) = y_n y_n (x_n^T x_n) = y_n^2 (x_n^T x_n) = \|x_n\|_2^2 \le R^2$

$\|w_{t+1}\|_2^2 = \|w_t + y_n x_n\|_2^2$

$= (w_t + y_n x_n)^T (w_t + y_n x_n)$

$= w_t^T w_t + w_t^T (y_n x_n) + (y_n x_n)^T w_t + (y_n x_n)^T (y_n x_n)$

$= \|w_t\|_2^2 + y_n (w_t^T x_n) + y_n (x_n^T w_t)$

$\|w_{t+1}\|_2^2 \le \|w_t\|_2^2 + R^2$

$\|w_t\|^2 \le \|w_{t-1}\|_2^2 + R^2$

$$\|w_{t+1}\|_2^2 = \|w_t\|_2^2 + 2\underbrace{y_n w_t^T x_n} + y_n^2 x_n^T x_n$$

$$\underset{\substack{\uparrow \\ \text{True label}}}{\underline{y_n}} \underset{\substack{\uparrow \\ \text{Prediction}}}{(\boxed{w_t}^T x_n)} \le 0$$

$$y_n^2 x_n^T x_n \le R^2$$

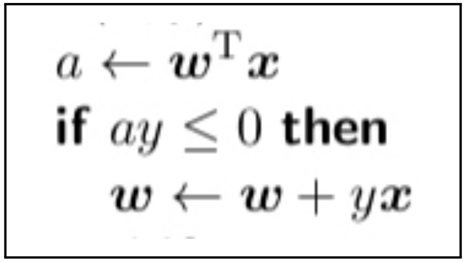
$$\le \|w_t\|_2^2 + \cancel{2.0} + \underbrace{y_n^2 x_n^T x_n}$$

$$\le \|w_t\|_2^2 + R^2$$

Proof (2/3)

Claim 2: After t mistakes, $\|w_t\|^2 \leq tR^2$

$$\begin{aligned}
\|\boldsymbol{w}_{t+1}\|^2 &= \|\boldsymbol{w}_t + y_n\boldsymbol{x}_n\|^2 \\
&= \|\boldsymbol{w}_t\|^2 + 2\boldsymbol{w}_t^{\mathrm{T}}(y_n\boldsymbol{x}_n) + \|y_n\boldsymbol{x}_n\|^2 \\
&= \|\boldsymbol{w}_t\|^2 + 2y_n(\boldsymbol{w}_t^{\mathrm{T}}\boldsymbol{x}_n) + y_n^2\|\boldsymbol{x}_n\|^2
\end{aligned}$$



```
$a \leftarrow \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$
if $ay \leq 0$ then
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$
```

Proof (2/3)

Claim 2: After t mistakes, $\|w_t\|^2 \leq tR^2$

$$\begin{aligned}\|\boldsymbol{w}_{t+1}\|^2 &= \|\boldsymbol{w}_t + y_n\boldsymbol{x}_n\|^2 \\ &= \|\boldsymbol{w}_t\|^2 + 2\boldsymbol{w}_t^{\mathrm{T}}(y_n\boldsymbol{x}_n) + \|y_n\boldsymbol{x}_n\|^2 \\ &\leq \|\boldsymbol{w}_t\|^2 + R^2\end{aligned}$$

Because $\boldsymbol{w}_0 = 0$, simple induction gives us: $\underline{\underline{\|\boldsymbol{w}_t\|^2 \leq tR^2.}}$

Proof (3/3)

$u^T w_t = \underline{\|w_t\|} \cos(\theta)$

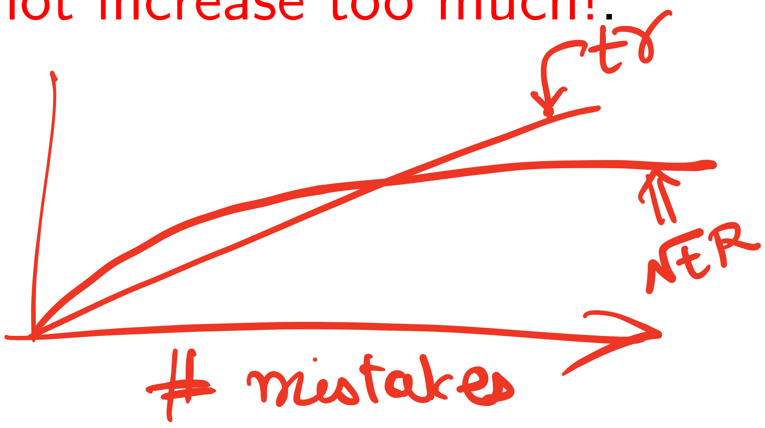
What we know

1. After t mistakes, $\boldsymbol{u}^{\mathrm{T}} \boldsymbol{w}_t \geq t\gamma$.
2. After t mistakes, $\underline{\|\boldsymbol{w}_t\|^2} \leq \underline{tR^2}$

$\Rightarrow \|w_t\| \leq \underline{\sqrt{t} R}$

The inner product between the true underlying model and the current model is non-decreasing after each update. This could be because the directions of $\boldsymbol{w}$ and $\boldsymbol{u}$ align or because the length of $\boldsymbol{w}$ increases.

But the length of $\boldsymbol{w}$ does not increase too much!.



Proof (3/3)

What we know

1. After t mistakes, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$.
2. After t mistakes, $\|\boldsymbol{w}_t\|^2 \leq tR^2$

$$\underline{\underline{R\sqrt{t} \geq \|\boldsymbol{w}_t\|}}$$

Proof (3/3)

$$\frac{R\cancel{\sqrt{t}}}{\cancel{\sqrt{t}}} \geq \frac{t\gamma}{\sqrt{t}}$$

$$\frac{R}{\gamma} \geq \sqrt{t}$$

$$\Rightarrow \frac{R^2}{\gamma^2} \geq t$$

What we know

1. After t mistakes, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$.
2. After t mistakes, $\|\boldsymbol{w}_t\|^2 \leq tR^2$

$$R\sqrt{t} \geq \|\boldsymbol{w}_t\| \geq \boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$$

$$u^T w_t = \|w_t\| \cos(\theta)$$

$$\leq \|w_t\|$$

$$\cos(\theta) \leq 1$$

Proof (3/3)

What we know

1. After t mistakes, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$.
2. After t mistakes, $\|\boldsymbol{w}_t\|^2 \leq tR^2$

$$R\sqrt{t} \geq \|\boldsymbol{w}_t\| \geq \boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$$

Proof (3/3)

What we know

1. After t mistakes, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$.
2. After t mistakes, $\|\boldsymbol{w}_t\|^2 \leq tR^2$

$$R\sqrt{t} \geq \|\boldsymbol{w}_t\| \geq \boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$$

Number of mistakes $t \leq \frac{R^2}{\underline{\underline{\gamma^2}}}$.

Bounds the total number of mistakes!

Beyond the separable case

- Good news
    - Perceptron makes no assumptions about the data, could be even adversarial.
    - After a fixed number of mistakes, you are done. Do not need to see any more data.
- Bad news
    - Real world data is often not linearly separable.

Outline

1 Review of previous lecture

2 Convergence of the Perceptron Learning Algorithm

3 Variants of perceptron

4 Logistic regression

Voting and averaging

- Vanilla perceptron returns final weight vector.
- Might lose good weight vectors that were learned during training.
- Aggregating the models (or weight vectors) seen during training may give better results (especially when data is not separable).

Voted perceptron

$$\underset{1}{\mathrm{sign}(w_0^T x)} \quad \underset{\boxed{100}}{\mathrm{sign}(w_1^T x)} \quad \underset{2}{\mathrm{sign}(w_2^T x)} \quad \ldots$$

- Remember every weight vector in your sequence of updates.
- At final prediction time, each weight vector gets to vote on the label.
- The number of votes it gets is the number of iterations it survived before being updated
- Comes with strong theoretical guarantees about generalization, impractical because of storage issues

Averaged perceptron

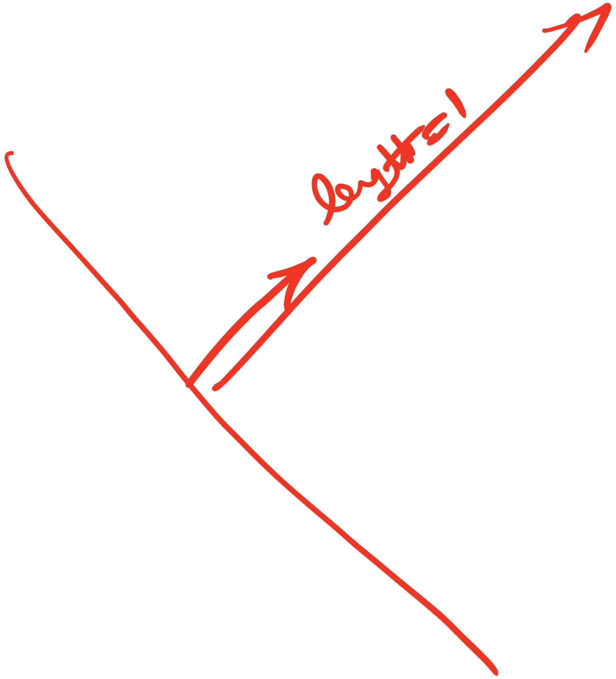
- Instead of using all weight vectors, use the average weight vector (i.e longer surviving weight vectors get more say)
- More practical alternative and widely used

Averaged Perceptron training/learning

$data$ = N **samples/instances:** $= \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_\mathsf{N}, y_\mathsf{N})\}$

Algorithm 1 AveragedPerceptronTrain $(data, maxIter)$

```
$\boldsymbol{w} \leftarrow \mathbf{0}$, $\boldsymbol{\mu} \leftarrow 0$.
for $iter = 1 \ldots MaxIter$ do
    for $(\boldsymbol{x}, y) \in data$ do
        $a \leftarrow \boldsymbol{w}^\mathrm{T}\boldsymbol{x}$
        if $ay \leq 0$ then
            $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$
        end if
        $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \boldsymbol{w}$
    end for
end for
return $\boldsymbol{\mu}$
```



Prediction: $sign(\boldsymbol{\mu}^\mathrm{T}\boldsymbol{x})$.

Summary

- Perceptron: linear classification
- Perceptron learning: learning from mistakes
- Perceptron convergence: linear separability and margin
- Variants: voting and averaging.
- See chapter 4 of CIML for reference.

Outline

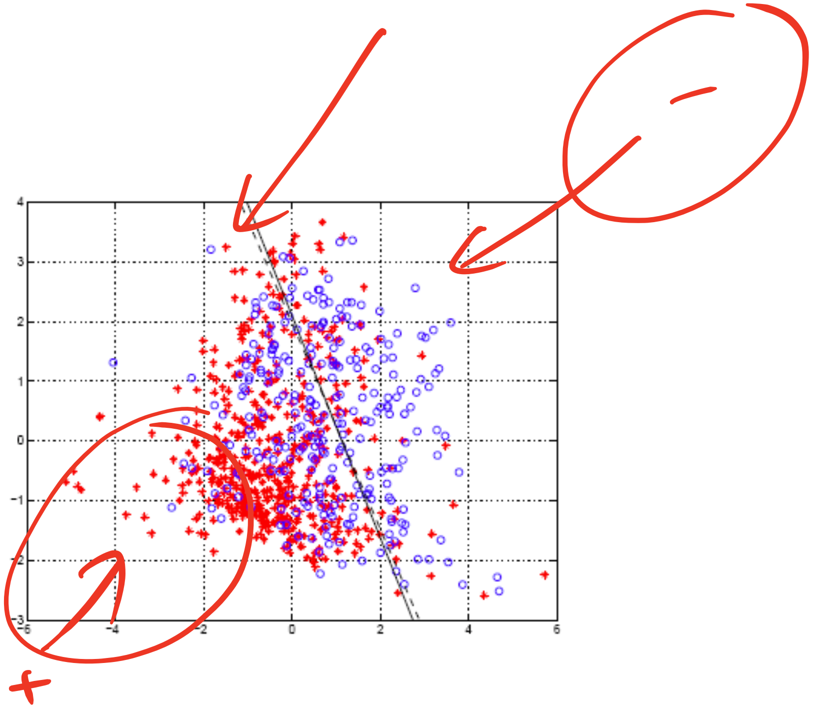
1. Review of previous lecture

2. Convergence of the Perceptron Learning Algorithm

3. Variants of perceptron

4. Logistic regression
   - General setup
   - Prediction in logistic regression
   - Training/Learning a logistic regression model
   - Maximum likelihood estimation
   - Optimization

Dealing with lack of separability



- Instead of predicting the class label $\underline{(-1 \text{ vs } +1)}$, predict the probability of instance being in a class $(\underline{P(y = 1|\boldsymbol{x})})$.
- Perceptron does not produce probability estimates

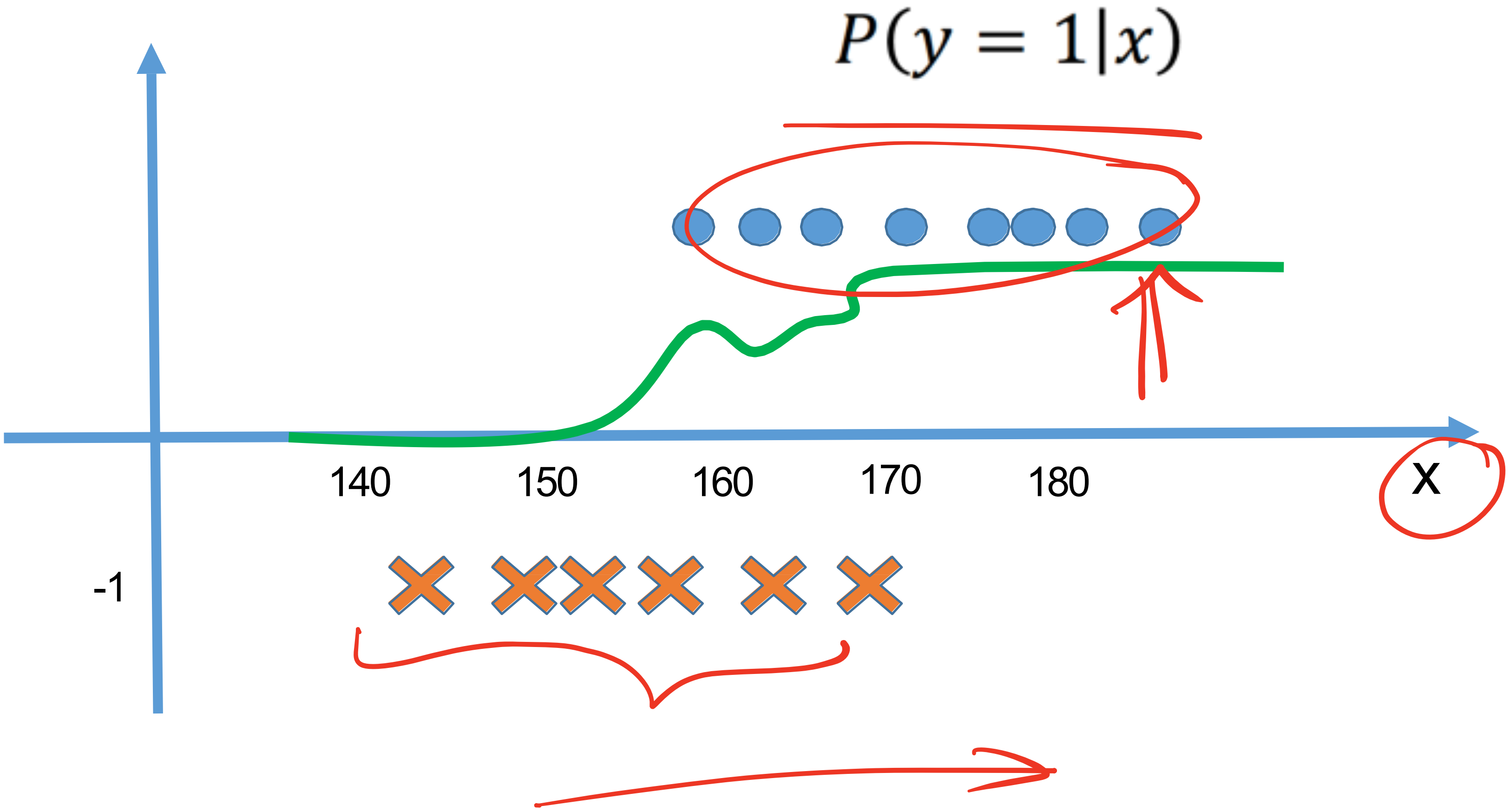
Predict $P(y = 1|\underline{\boldsymbol{x}})$

$$0 \leq P(y=1|x) \leq 1$$

- Previously, hypothesis space consists of functions that output $\{-1, +1\}$. Now change the hypothesis space to functions that output a value in $[0, 1]$.
- Build a model $h(\underline{\boldsymbol{x}}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$ such that $\approx P(y = 1|\boldsymbol{x})$.
- Effectively make the problem a regression problem.

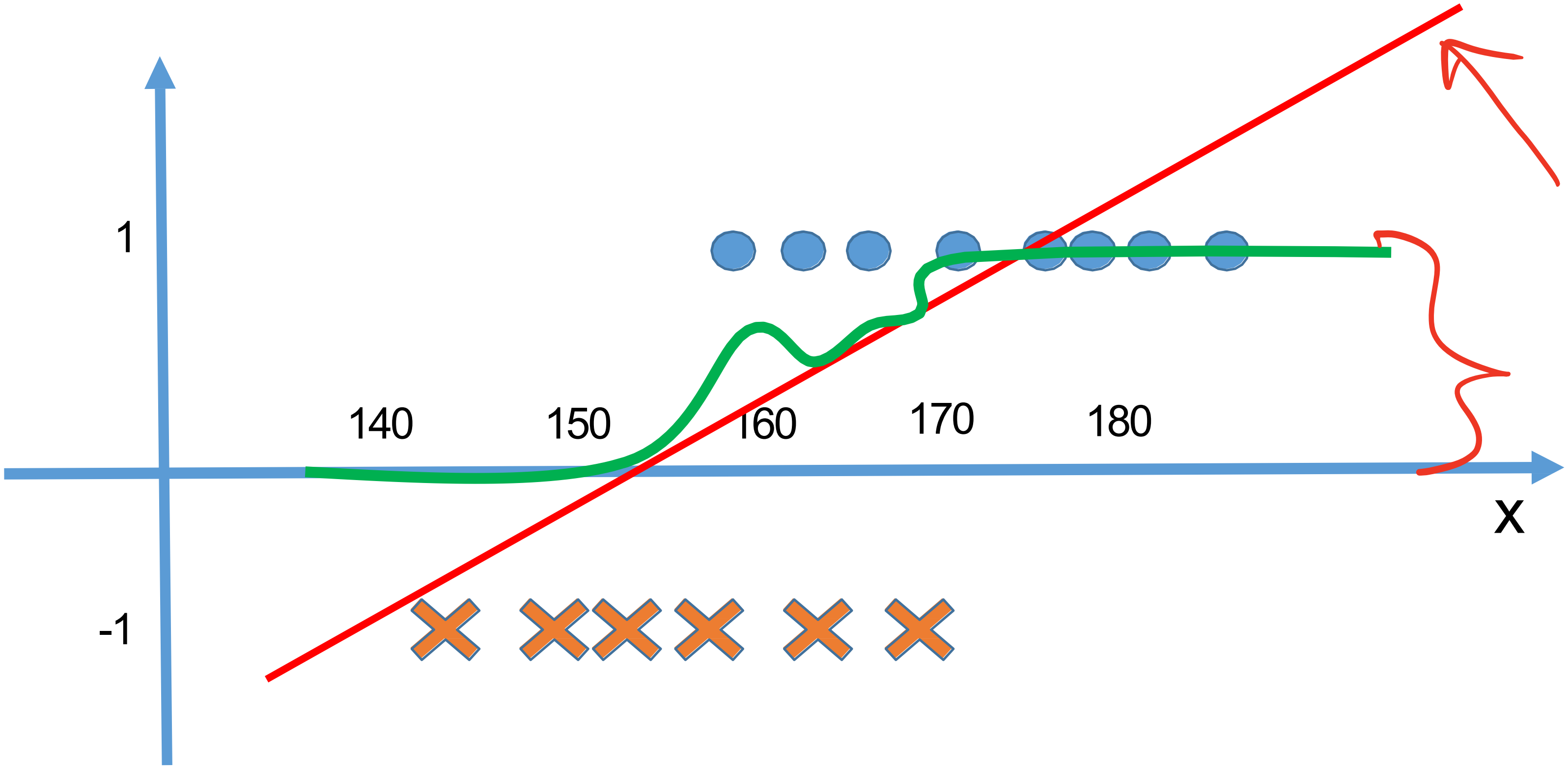
How to model $P(y = 1|x)$

What is the target function? $f(x)$



How to model $\underline{\underline{P(y=1|\boldsymbol{x})}}$

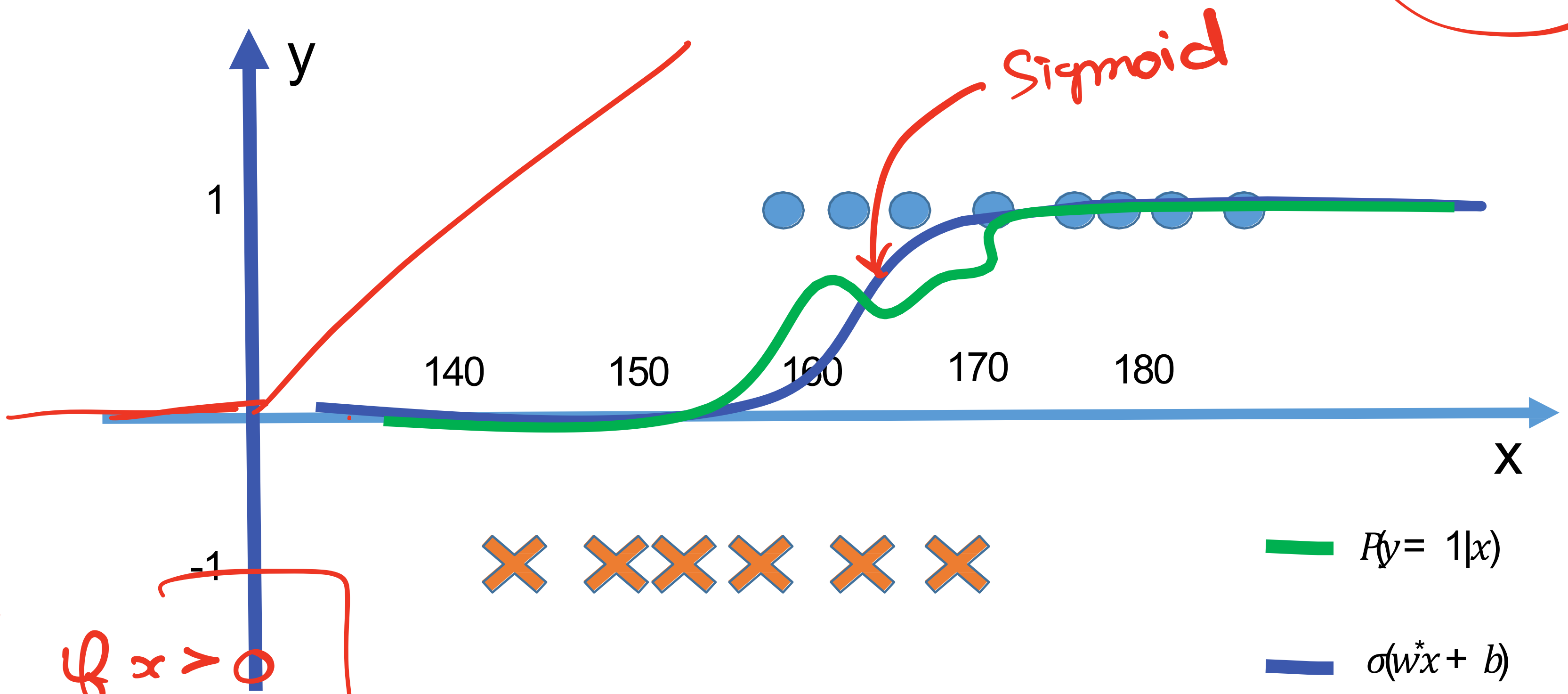
Can we use a linear function $y = \underline{\underline{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}}} + \underline{\underline{b}}$?



How to model $P(y=1|\boldsymbol{x})$

arctan

Define a transformation function $\sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b) \approx P(y=1|\boldsymbol{x})$.

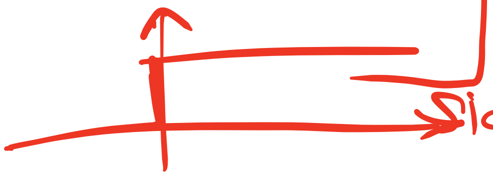


21

$\sigma(x)=1$ if $x>0$

$\sigma(x)=0$ if $x<0$

$$\sigma(x) = \mathrm{ReLu}(x) \begin{cases} 0 & \text{if } x<0 \\ x & \text{if } x \geq 0 \end{cases}$$

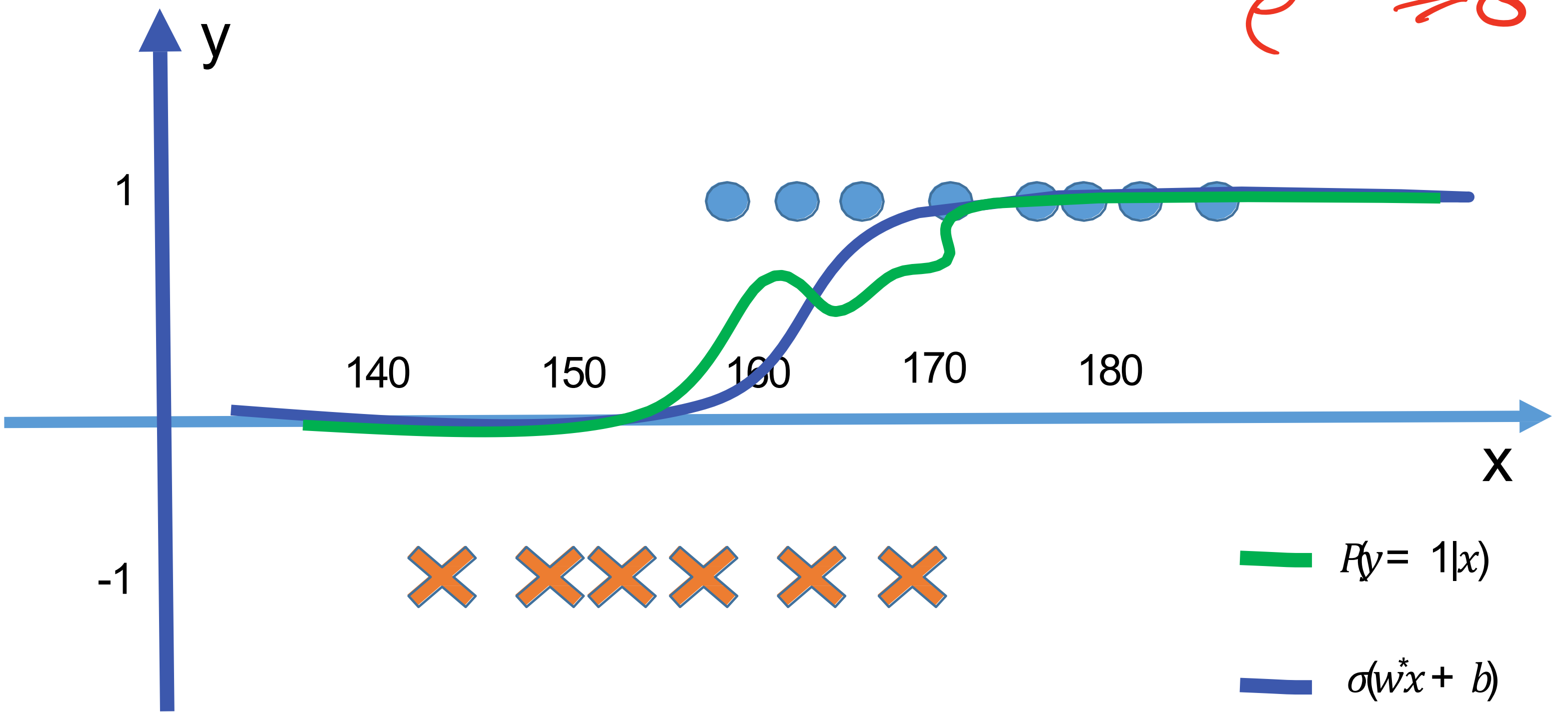


$$\mathrm{sign}(x) = \begin{cases} +1 & x \geq 0 \\ -1 & \text{if } x<0 \end{cases}$$

How to model $P(y = 1|\boldsymbol{x})$

What is a good choice of transformation function ? For example, exponent function $\underline{: e^x}$?

$e^x \geq 0$



21

Logistic regression

Setup for binary classification

- Input: $\boldsymbol{x} \in \mathbb{R}^D$
- Output: $\underline{y \in \{0, 1\}}$
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, N\}$
- Hypotheses/Model:

$$h_{\underline{\boldsymbol{w},b}}(x) = p(y = 1|\boldsymbol{x}; b, \boldsymbol{w}) = \underline{\sigma(a(\boldsymbol{x}))}$$

where

$$a(\boldsymbol{x}) = b + \underline{\sum_d w_d x_d} = b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$

and $\sigma(\cdot)$ stands for the *sigmoid* function

$$0 \leq \sigma(\underline{a}) = \left.\frac{1}{1 + e^{-a}}\right\} \leq 1$$

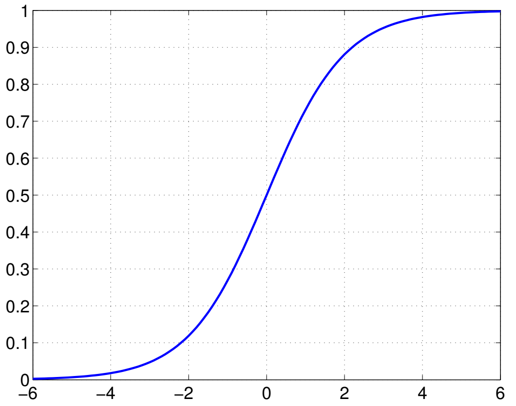
Why the sigmoid function?

What does it look like?

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

where

$$a = b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$



Properties

Why the sigmoid function?

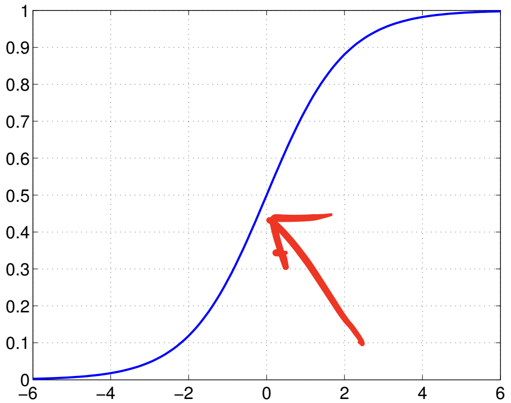
$$\frac{1}{1+e^{-1000}}$$

What does it look like?

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

where

$$a = b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$



Properties

- Bounded between 0 and 1 ← thus, interpretable as probability
- Monotonically increasing thus, usable to derive classification rules
  - $\sigma(a) > 0.5$, positive (classify as '1')
  - $\sigma(a) < 0.5$, negative (classify as '0')
  - $\sigma(a) = 0.5$, undecidable
- Nice computational properties As we will see soon

The hypothesis space for logistic regression

$$\underbrace{(w^T x + b)}$$

All functions of the form

$$h_{\boldsymbol{w},b}(x) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$$

A linear function composed with a sigmoid function.

We want to find a model $h_{\boldsymbol{w},b}(x)$ such that

$$h_{\boldsymbol{w},b}(x) \approx p(y = 1|\boldsymbol{x})$$

Mini-summary

- What is the goal of logistic regression ?
  - Model $P(y=1|\boldsymbol{x})$
- What is the hypothesis space ?
  - $H = \{h | h : \mathbb{X} \rightarrow \underline{[0,1]}, h(\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)\}$

Prediction in logistic regression

$$P(y=0|x) = 1 - P(y=1|x)$$

$$\underbrace{P(y=1|\boldsymbol{x})} = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b) = \frac{1}{1+\exp^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)}}$$

Compute $\sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)$. If this is greater than 0.5, predict 1 else 0.

Decision boundary: Linear or nonlinear?

Decision boundary: Linear or nonlinear?

$\sigma(a)$ **is nonlinear**, however, the decision boundary is determined by

$$\sigma(a) = 0.5 \Rightarrow a(\boldsymbol{x}) = 0 \Rightarrow a(\boldsymbol{x}) = b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} = 0$$

which is a *linear* function in $\boldsymbol{x}$

As in the case of perceptron, b the bias or offset or intercept term.

$\boldsymbol{w}$ the weights .

Logistic regression

Setup for binary classification

- Input: $\boldsymbol{x} \in \mathbb{R}^D$
- Output: $y \in \{0, 1\}$
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, N\}$
- Hypotheses/Model:

$$h_{\boldsymbol{w},b}(x) = p(y = 1|\boldsymbol{x}; b, \boldsymbol{w}) = \sigma(a(\boldsymbol{x}))$$

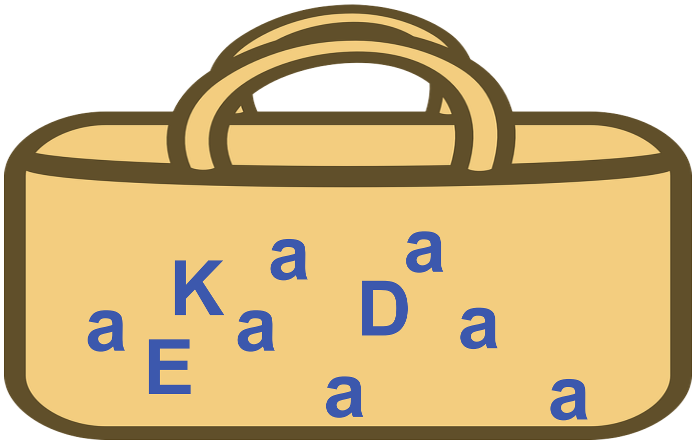
  where

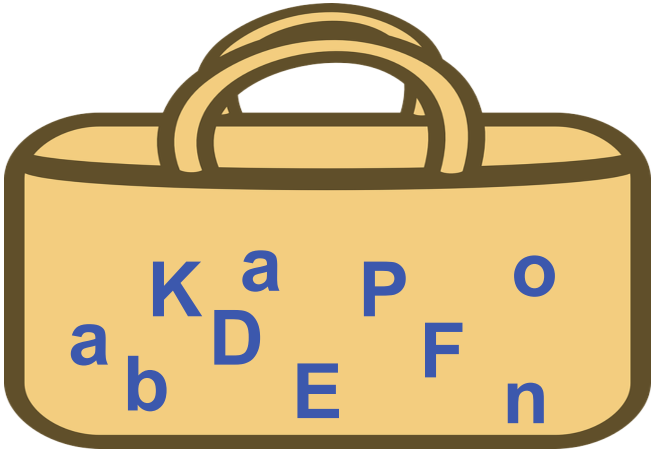
$$a(\boldsymbol{x}) = b + \sum_d w_d x_d = b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$

- Given training data N samples/instances: $\mathcal{D}^{\mathrm{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{N}}, y_{\mathsf{N}})\}$, train/learn/induce $h_{\boldsymbol{w},b}$. Find values for $(\boldsymbol{w}, b)$.

Example: bag of words

Which bag of words is more likely to generate : aDaaa ?

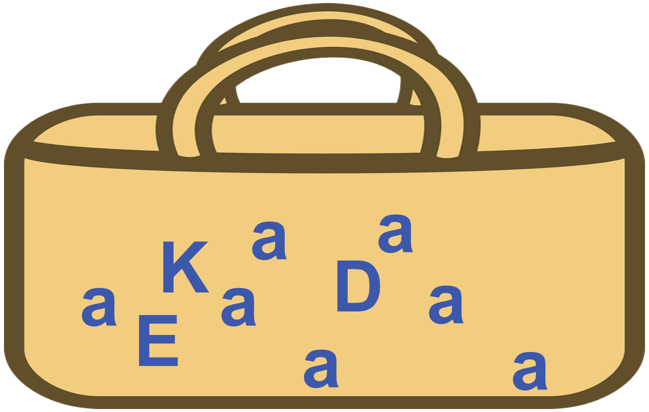




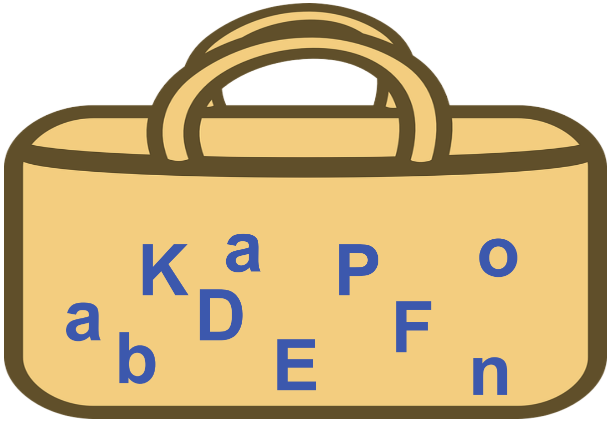
Example: bag of words

Which bag of words is more likely to generate : aDaaa ?

$0.7 \times 0.1 \times 0.7 \times 0.7 \times 0.7$
$= 2.401 \times 10^{-2}$



$0.2 \times 0.1 \times 0.2 \times 0.2 \times 0.2$
$= 1.6 \times 10^{-4}$



Example: drawing color cards from an envelope

- An envelope with two colors of cards: yellow and purple.
- Assume if you draw a card at random, probability of drawing a yellow card is θ and probability of drawing a purple card is $1 - \theta$.
- Sample with replacement n times.
- k times we get yellow, $n - k$ times we get purple.
- The joint probability (likelihood) : $\theta^k(1-\theta)^{n-k}$.
- What is the value of θ that maximizes the joint probability?

Example: drawing color cards from an envelope

Solve $argmax_\theta \theta^k (1-\theta)^{n-k}$

Equivalently, we can solve:

$$\begin{aligned} & argmax_\theta \log(\theta^k (1-\theta)^{n-k}) \\ & = argmax_\theta k \log \theta + (n-k) \log(1-\theta) \end{aligned}$$

Example: drawing color cards from an envelope

Solve $argmax_\theta \theta^k (1-\theta)^{n-k}$

Equivalently, we can solve:

$$\begin{aligned} & argmax_\theta \log(\theta^k (1-\theta)^{n-k}) \\ & = argmax_\theta k \log\theta + (n-k)\log(1-\theta) \end{aligned}$$

At the optimum: $\frac{(k\log\theta + (n-k)\log(1-\theta))}{d\theta} = 0$

Example: drawing color cards from an envelope

Maximum likelihood estimate (MLE): $\hat{\theta} = \frac{k}{n}$

These are easy examples. We don't always have a closed-form solution for the MLE typically !

Likelihood Function

Let $X_1, \ldots, X_N$ be IID (independent and identically distributed) random variables with PDF $p(x|\theta)$ (also written as $p(x;\theta)$). The *likelihood function* is defined by $L(\theta)$,

$$L(\theta) = p(X_1, \ldots, X_N; \theta). \qquad = \prod_{i=1}^{N} p(X_i; \theta).$$

Notes The likelihood function is just the joint density of the data, except that we treat it as a function of the parameter θ.

Maximum Likelihood Estimator

Definition: The maximum likelihood estimator (MLE) $\hat{\theta}$, is the value of θ that maximizes $L(\theta)$.

The log-likelihood function is defined by $l(\theta) = \log L(\theta)$. Its maximum occurs at the same place as that of the likelihood function.

Maximum Likelihood Estimator

Definition: The maximum likelihood estimator (MLE) $\hat{\theta}$, is the value of θ that maximizes $L(\theta)$.

The log-likelihood function is defined by $l(\theta) = \log L(\theta)$. Its maximum occurs at the same place as that of the likelihood function.

- Using logs simplifies mathematical expressions (converts exponents to products and products to sums)
- Using logs helps with numerical stabilitity

The same is true of the likelihood function times any constant. Thus we shall often drop constants in the likelihood function.

Likelihood function for logistic regression

Probability of a single training sample $(\boldsymbol{x}_n, y_n)$

$$p(y_n|\boldsymbol{x}_n; b, \boldsymbol{w}) = \begin{cases} h_{\boldsymbol{w},b}(\boldsymbol{x}_n) = \sigma(b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) & \text{if} \quad y_n = 1 \\ = 1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n) = 1 - \sigma(b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) & \text{otherwise} \end{cases}$$

Likelihood function for logistic regression

Probability of a single training sample $(\boldsymbol{x}_n, y_n)$

$$p(y_n|\boldsymbol{x}_n; b, \boldsymbol{w}) = \begin{cases} h_{\boldsymbol{w},b}(\boldsymbol{x}_n) = \sigma(b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) & \text{if} \quad y_n = 1 \\ = 1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n) = 1 - \sigma(b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) & \text{otherwise} \end{cases}$$

Compact expression, exploring that y_n is either 1 or 0

$$p(y_n|\boldsymbol{x}_n; b; \boldsymbol{w}) = h_{\boldsymbol{w},b}(\boldsymbol{x}_n)^{y_n}[1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]^{1-y_n}$$

Log Likelihood

Log-likelihood of the whole training data $\mathcal{D}$

$$l(\boldsymbol{w}, b) = \sum_n \{y_n \log h_{\boldsymbol{w},\boldsymbol{b}}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]\}$$

Log Likelihood

Log-likelihood of the whole training data $\mathcal{D}$

$$l(\boldsymbol{w}, b) = \sum_n \{y_n \log h_{\boldsymbol{w},\boldsymbol{b}}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]\}$$

It is convenient to work with its negation termed negative log likelihood

$$J(b, \boldsymbol{w}) = -\sum_n \{y_n \log h_{\boldsymbol{w},b}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]\}$$

We can ignore the distinction between bias and weights

This is for convenience

- Append 1 to $\boldsymbol{x}$

$$\boldsymbol{x} \leftarrow [1 \quad x_1 \quad x_2 \quad \cdots \quad x_D]$$

- Append b to $\boldsymbol{w}$

$$\boldsymbol{\theta} \leftarrow [b \quad w_1 \quad w_2 \quad \cdots \quad w_D]$$

- $$J(\boldsymbol{\theta}) = -\sum_n \{y_n \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]\}$$

- Same trick as in the case of perceptrons
  - we are rewriting a hyperplane in D dimensions as one in $D + 1$ dimensions that passes through the origin.

How to find the optimal parameters for logistic regression?

We will minimize the negative log likelihood

$$J(\boldsymbol{\theta}) = -\sum_n \{y_n \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]\}$$

How to find the optimal parameters for logistic regression?

We will minimize the negative log likelihood

$$J(\boldsymbol{\theta}) = -\sum_n \{y_n \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]\}$$

How do we find its minimum?

Optimization

Given a function $f(x)$, find its minimum (or maximum).

- f is called the objective function.

Optimization

Given a function $f(x)$, find its minimum (or maximum).

- f is called the objective function.
- Maximizing f is equivalent to minimizing $-f$.

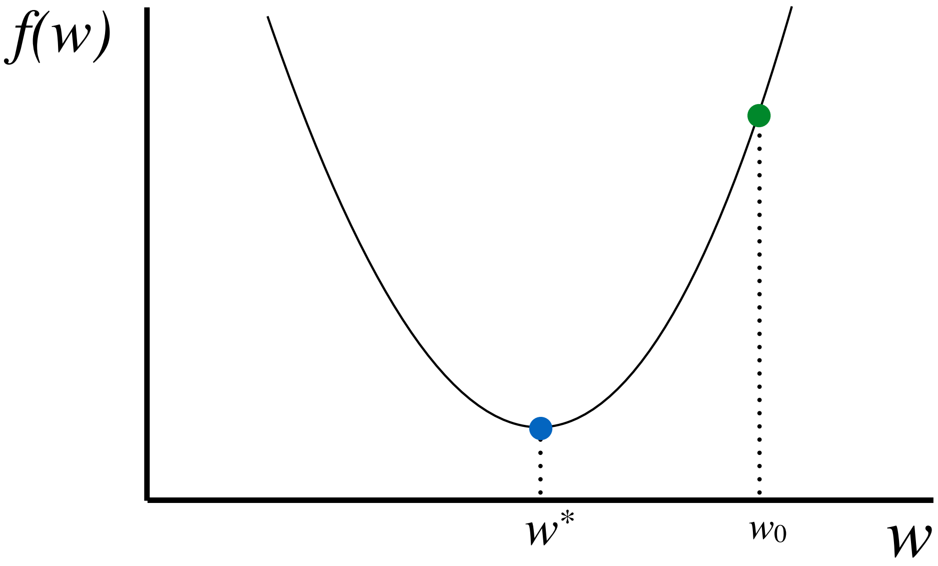
So we only need to consider minimization problems.

One way to minimize a function f

Gradient descent

Gradient Descent

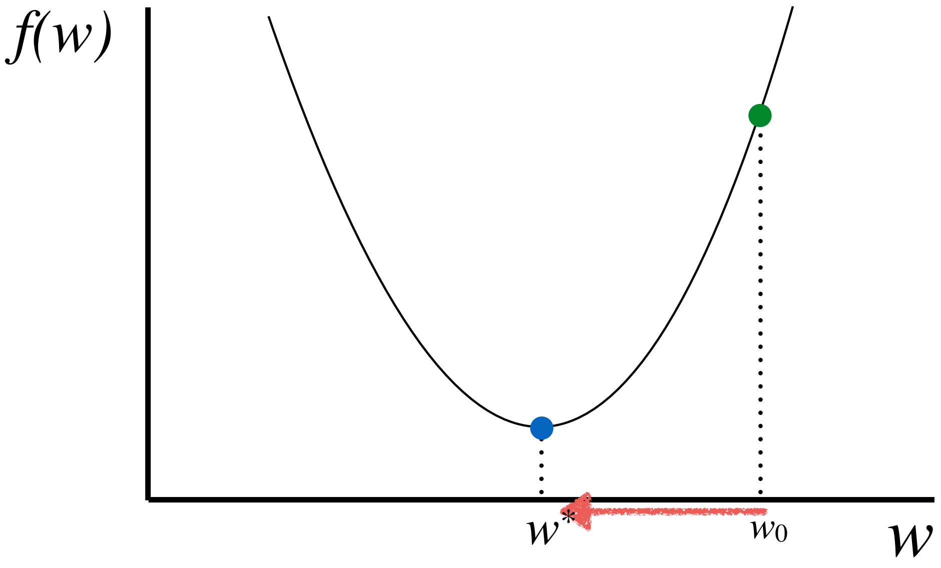
Start at a random point



Gradient Descent

Start at a random point

Determine a descent direction

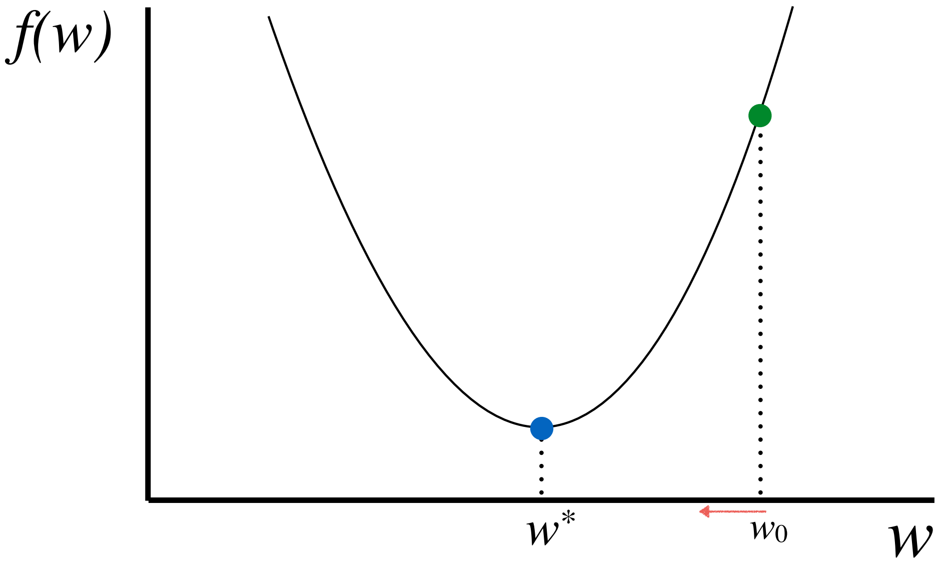


Gradient Descent

Start at a random point

Determine a descent direction

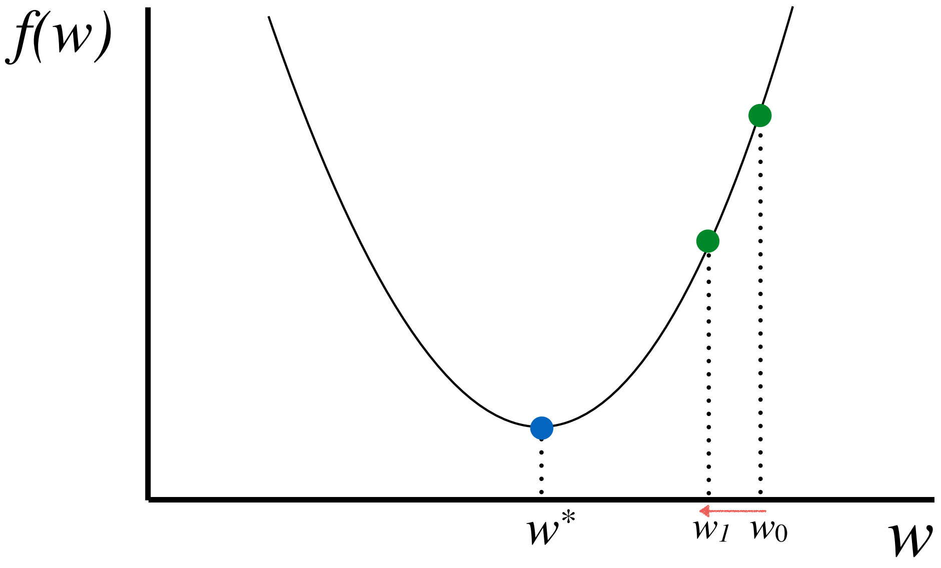
Choose a step size



Gradient Descent

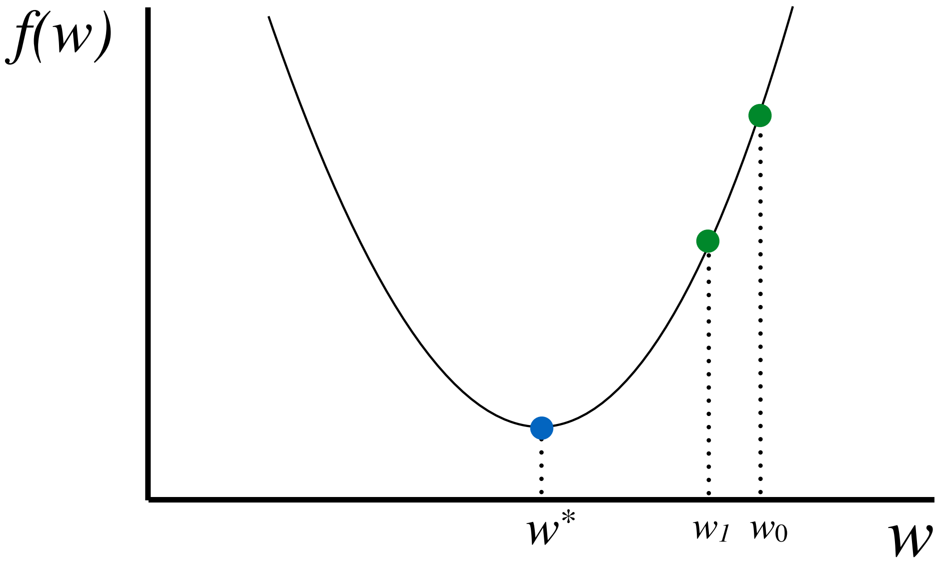
Start at a random point

- Determine a descent direction
- Choose a step size
- Update



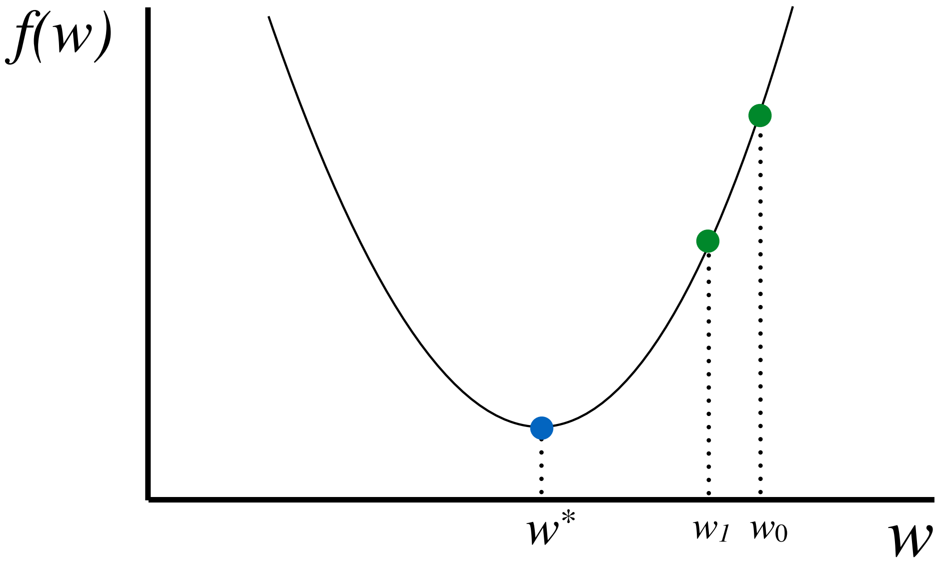
Gradient Descent

```
Start at a random point
Repeat
    Determine a descent direction
    Choose a step size
    Update
Until stopping criterion is satisfied
```



Gradient Descent

```
Start at a random point
Repeat
    Determine a descent direction
    Choose a step size
    Update
Until stopping criterion is satisfied
```



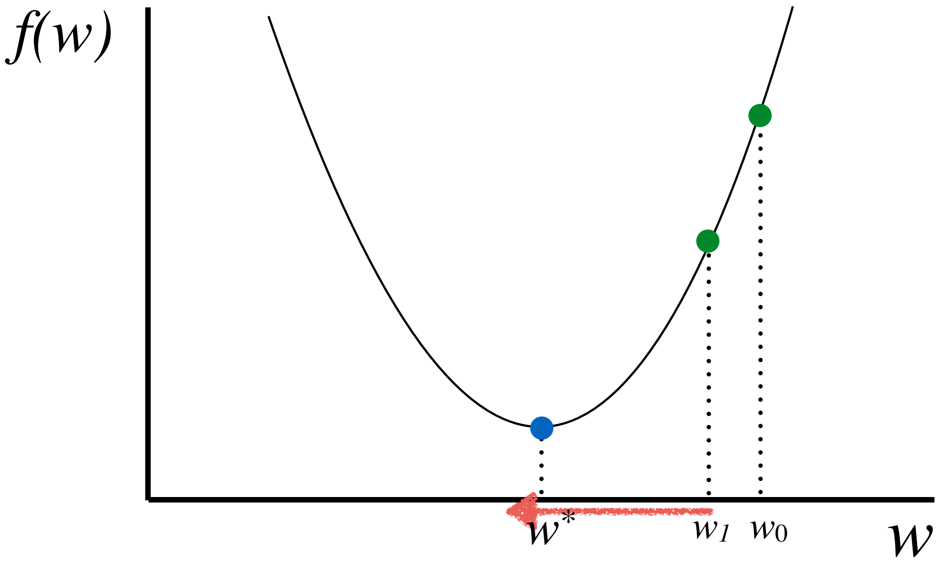
Gradient Descent

Start at a random point

Repeat

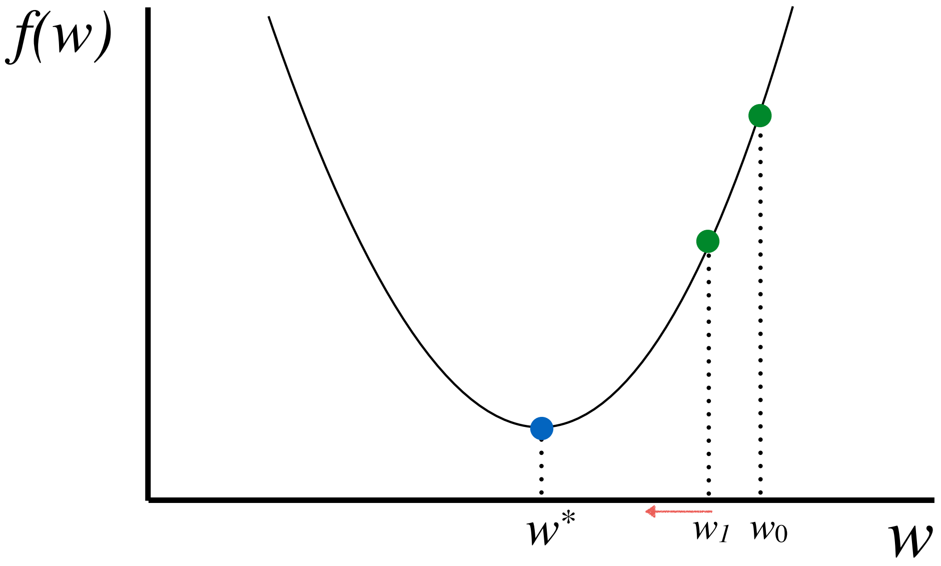
- Determine a descent direction
- Choose a step size
- Update

Until stopping criterion is satisfied



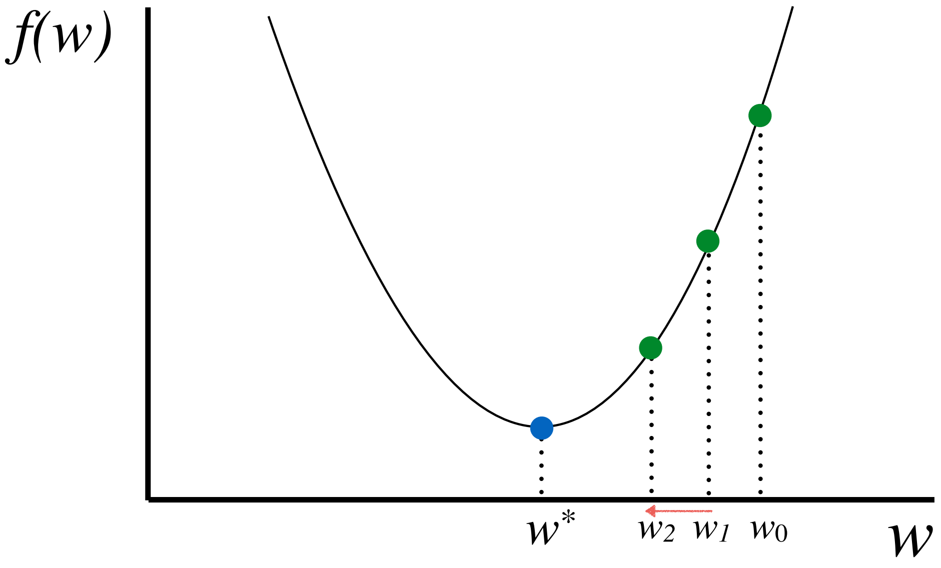
Gradient Descent

```
Start at a random point
Repeat
    Determine a descent direction
    Choose a step size
    Update
Until stopping criterion is satisfied
```



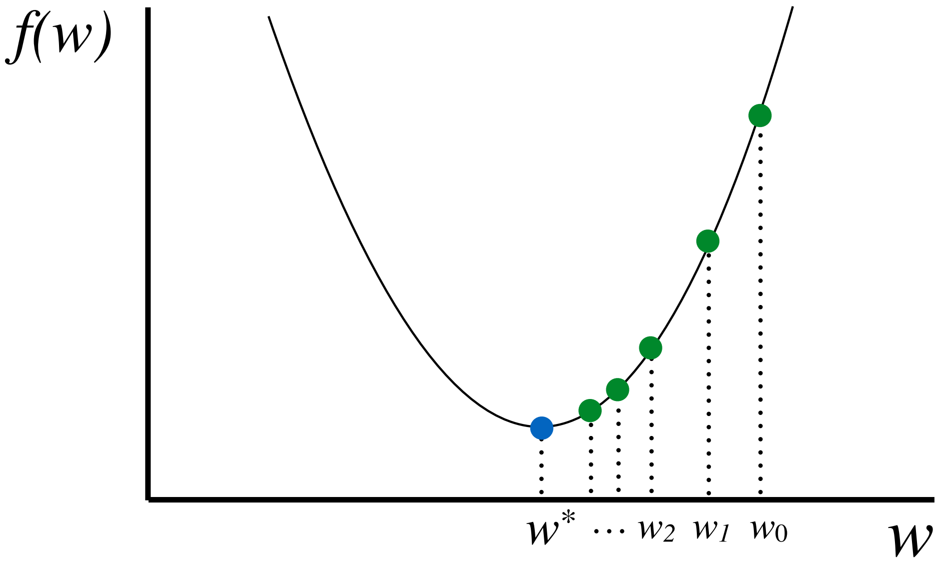
Gradient Descent

```
Start at a random point
Repeat
    Determine a descent direction
    Choose a step size
    Update
Until stopping criterion is satisfied
```

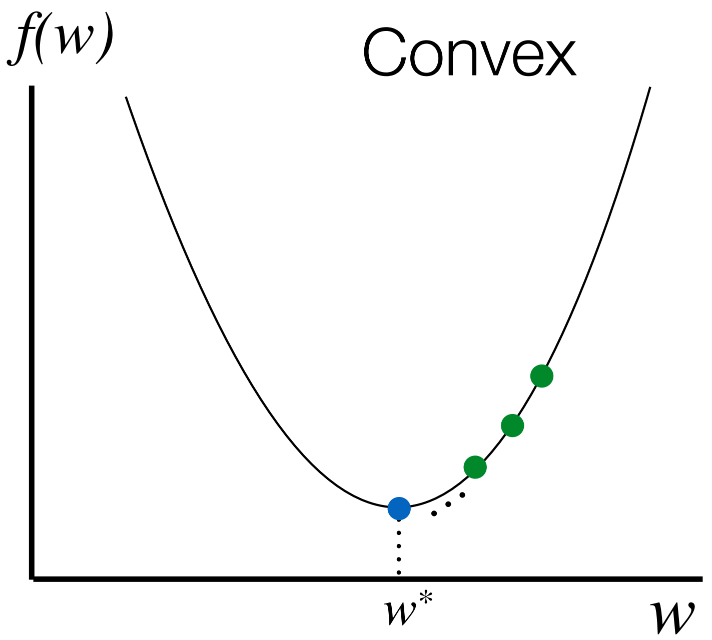


Gradient Descent

```
Start at a random point
Repeat
    Determine a descent direction
    Choose a step size
    Update
Until stopping criterion is satisfied
```

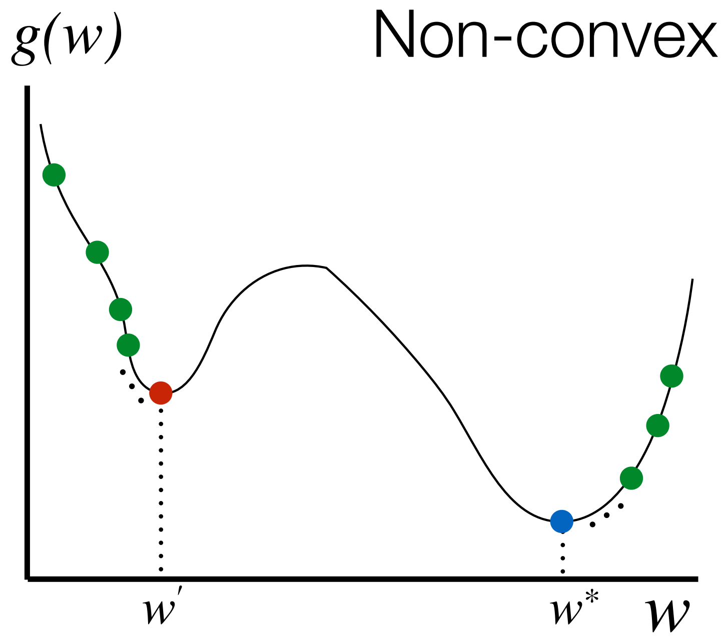


Where Will We Converge?



Convex

Any local minimum is a global minimum



Non-convex

Multiple local minima may exist

Least Squares, Ridge Regression and Logistic Regression are all convex!

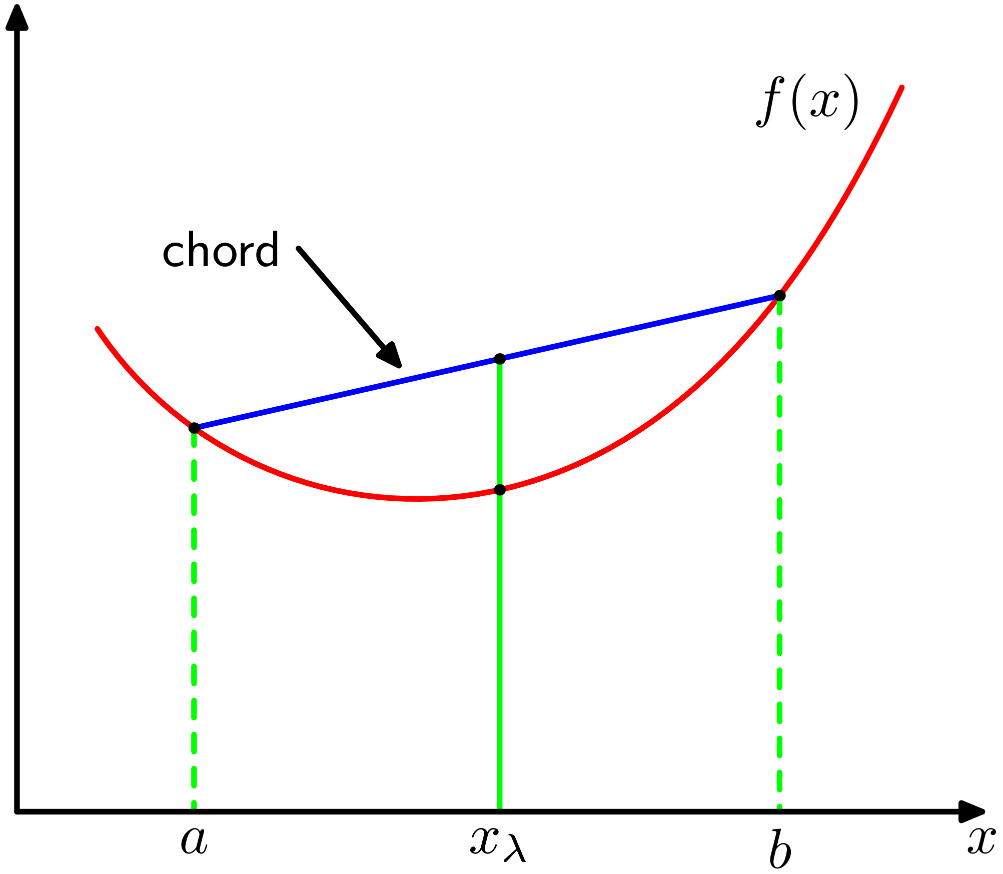
Convex functions

A function $f(x)$ is convex if

$$f(\lambda a + (1-\lambda)b) \leq \lambda f(a) + (1-\lambda) f(b)$$

for

$$0 \leq \lambda \leq 1$$



How to determine convexity?

$f(x)$ is convex if

$$f^{''}(x) \geq 0$$

Examples:

$$f(x) = x^2, f^{''}(x) = 2 > 0$$

Examples

Convex functions

$$f(x) = ax + b$$

$$f(x) = x^2$$

$$f(x) = e^x$$

$$f(x) = \frac{1}{x}, x \geq 0$$

Examples

Nonconvex functions

$$f(x) = \cos(x)$$

$$f(x) = e^x - x^2$$

$$f(x) = \log(x)$$

Multi-variate functions

Definition

$f(\boldsymbol{x})$ is convex

$$f(\lambda \boldsymbol{a} + (1-\lambda)\boldsymbol{b}) \leq \lambda f(\boldsymbol{a}) + (1-\lambda) f(\boldsymbol{b})$$

for all $\boldsymbol{a}$, $\boldsymbol{b}$, $0 \leq \lambda \leq 1$

Multi-variate functions

How to determine convexity in this case?

Matrix of second-order derivatives (Hessian)

$$\boldsymbol{H} = \begin{pmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial {x_1}^2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_D} \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_D} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_D} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_D} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_D^2} \end{pmatrix}$$

Multi-variate functions

How to determine convexity in this case?

If the Hessian is positive semi-definite $\boldsymbol{H} \succeq 0$, then f is convex.

A matrix $\boldsymbol{H}$ is positive semi-definite if and only if

$$\boldsymbol{z}^T \boldsymbol{H} \boldsymbol{z} = \sum_{j,k} H_{j,k} z_j z_k \geq 0$$

for all $\boldsymbol{z}$.

Multi-variate functions

Example

$$f(\boldsymbol{x}) = x_1^2 + 2x_2^2$$

$$\boldsymbol{H} = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$$

Multi-variate functions

Example

$$f(\boldsymbol{x}) = x_1^2 + 2x_2^2$$

$$\boldsymbol{H} = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$$

$$\boldsymbol{z}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{z} = 2z_1^2 + 4z_2^2 \geq 0$$

Example: $\min f(\boldsymbol{\theta}) = 0.5(\theta_1^2 - \theta_2)^2 + 0.5(\theta_1 - 1)^2$

- We compute the gradients

$$\frac{\partial f}{\partial \theta_1} = 2(\theta_1^2 - \theta_2)\theta_1 + \theta_1 - 1 \tag{1}$$

$$\frac{\partial f}{\partial \theta_2} = -(\theta_1^2 - \theta_2) \tag{2}$$

- Use the following *iterative* procedure for *gradient descent*
  1. Initialize $\theta_1^{(0)}$ and $\theta_2^{(0)}$, and $t = 0$
  2. do

$$\theta_1^{(t+1)} \leftarrow \theta_1^{(t)} - \eta \left[2({\theta_1^{(t)}}^2 - \theta_2^{(t)})\theta_1^{(t)} + \theta_1^{(t)} - 1\right] \tag{3}$$

$$\theta_2^{(t+1)} \leftarrow \theta_2^{(t)} - \eta \left[-({\theta_1^{(t)}}^2 - \theta_2^{(t)})\right] \tag{4}$$

$$t \leftarrow t + 1 \tag{5}$$

  3. until $f(\boldsymbol{\theta}^{(t)})$ *does not change much*

Gradient descent

General form for minimizing $f(\boldsymbol{\theta})$

$$\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta} - \eta \frac{\partial f}{\partial \boldsymbol{\theta}}$$

Remarks

- η is often called *step size* – literally, how far our update will go along the the direction of the negative gradient
- Note that this is for *minimizing* a function, hence the subtraction $(-\eta)$
- With a *suitable* choice of η, the iterative procedure converges to a stationary point where
$$\frac{\partial f}{\partial \boldsymbol{\theta}} = 0$$
- A stationary point is only necessary for being the minimum.

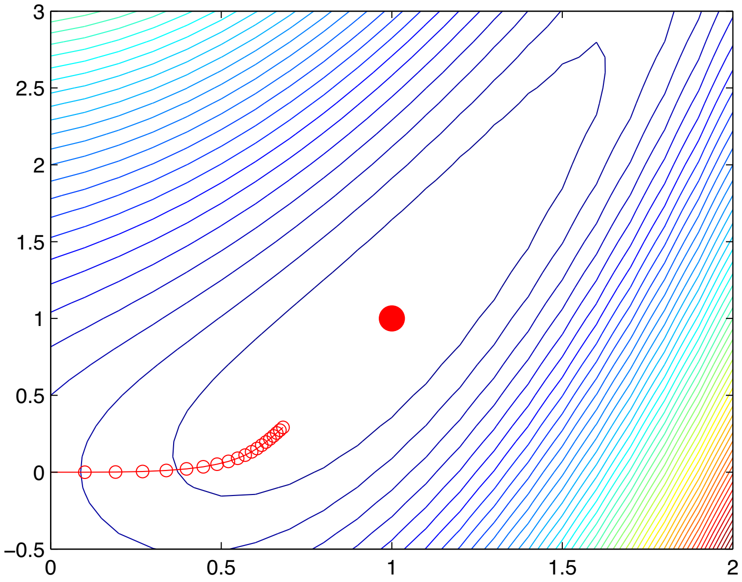
Seeing in action

Choosing the right η is important

Seeing in action

Choosing the right η is important

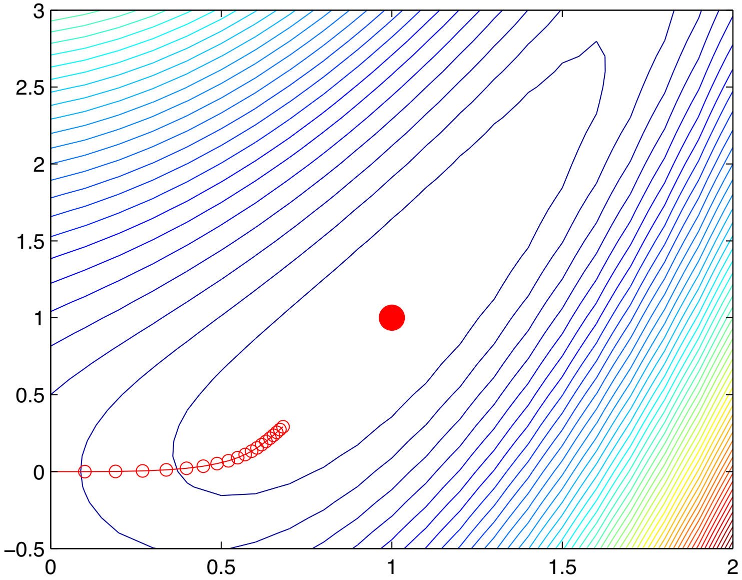
small η is too slow?



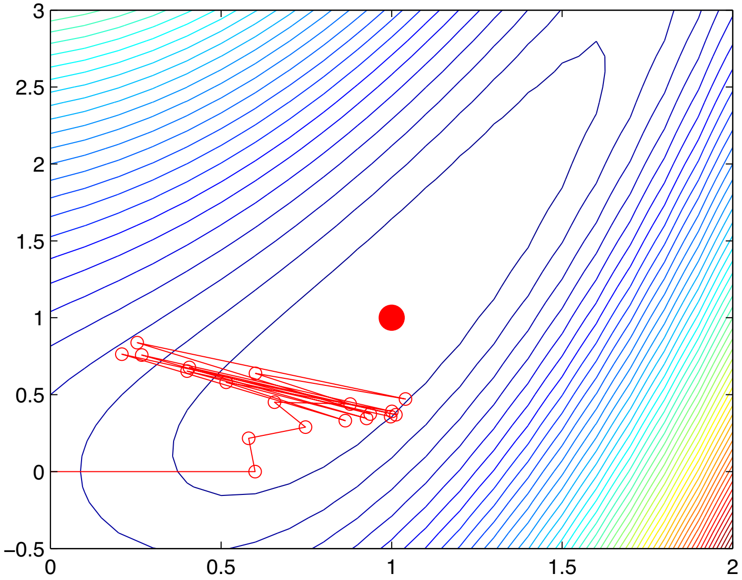
Seeing in action

Choosing the right η is important

small η is too slow?



large η is too unstable?



Gradient Descent Update for Logistic Regression

Simple fact: derivatives of $\sigma(a)$

$$
\begin{aligned}
\frac{d\,\sigma(a)}{d\,a} &= \frac{d}{d\,a}\left(1 + e^{-a}\right)^{-1} = \frac{-(1+e^{-a})'}{(1+e^{-a})^2} \\
&= \frac{e^{-a}}{(1+e^{-a})^2} = \frac{1}{1+e^{-a}}\frac{e^{-a}}{1+e^{-a}} \\
&= \sigma(a)[1-\sigma(a)]
\end{aligned}
$$

Gradients of the negative log likelihood

Negative log likelihood

$$J(\boldsymbol{\theta}) = -\sum_n \{y_n \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]\}$$

Gradients

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\sum_n \left\{y_n[1 - \sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\boldsymbol{x}_n - (1 - y_n)\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\boldsymbol{x}_n\right\} \tag{6}$$

$$= \sum_n \left\{\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n) - y_n\right\} \boldsymbol{x}_n \tag{7}$$

$$= \sum_n \left\{h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - y_n\right\} \boldsymbol{x}_n \tag{8}$$

Remark

Gradients of the negative log likelihood

Negative log likelihood

$$J(\boldsymbol{\theta}) = -\sum_n \{y_n \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]\}$$

Gradients

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\sum_n \left\{y_n[1 - \sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\boldsymbol{x}_n - (1 - y_n)\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\boldsymbol{x}_n\right\} \tag{6}$$

$$= \sum_n \left\{\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n) - y_n\right\} \boldsymbol{x}_n \tag{7}$$

$$= \sum_n \left\{h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - y_n\right\} \boldsymbol{x}_n \tag{8}$$

Remark

- $e_n = \{h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - y_n\}$ is called *error* for the nth training sample.

Numerical optimization

Gradient descent

- Choose a proper step size $\eta > 0$
- Iteratively update the parameters following the negative gradient to minimize the error function

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \sum_n \left\{ \sigma(\boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}_n) - y_n \right\} \boldsymbol{x}_n$$

Remarks

- The step size needs to be chosen carefully to ensure convergence.
- The step size can be adaptive (i.e. varying from iteration to iteration). For example, we can use techniques such as *line search*

Summary

Setup for binary classification

- Logistic Regression models conditional distribution as: $p(y = 1|\boldsymbol{x}; \boldsymbol{\theta}) = \sigma[a(\boldsymbol{x})]$ where $a(\boldsymbol{x}) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}$
- Linear decision boundary: $a(\boldsymbol{x}) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x} = 0$

Minimizing the negative log-likelihood

- $J(\boldsymbol{\theta}) = -\sum_n \{y_n \log \sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\}$
- No closed form solution; must rely on iterative solvers

Numerical optimization

- Gradient descent: simple, scalable to large-scale problems
  - move in direction opposite of gradient!
  - gradient of logistic function takes nice form