Lecture 7: Logistic Regression

Fall 2022

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikuar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu, Hal Daume whose slides are heavily used, and the many others who made their course material freely available online.

Announcement

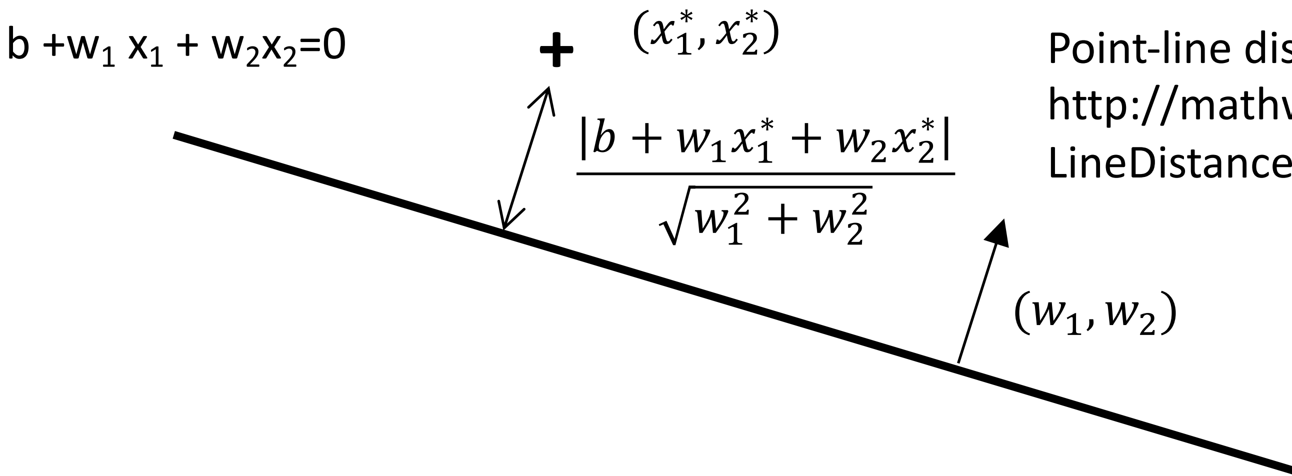
- Quiz 2 is coming!
- Suggested readings in BruinLearn
  http://ciml.info/
- Please don't put CM146 course materials online

- Explanation of margin γ

$$y_i(\boldsymbol{u}^T \boldsymbol{x_i} + b) \geq \gamma$$

Recall: The geometry of a linear classifier

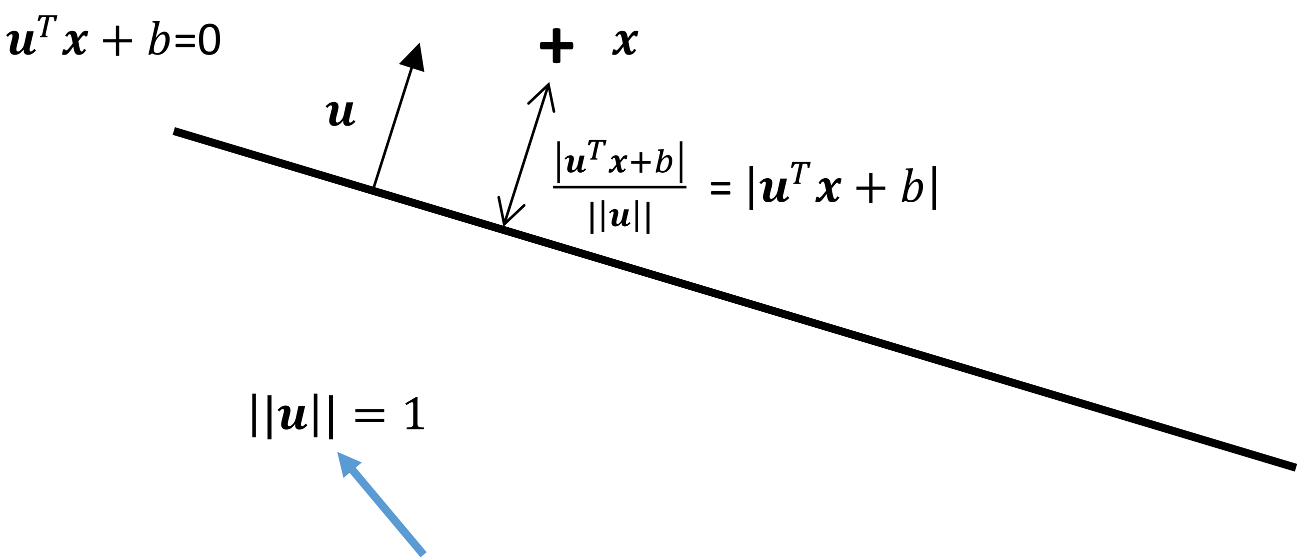
Prediction = sgn($b + w_1 x_1 + w_2 x_2$)



Point-line distance: http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

Recall: The geometry of a linear classifier

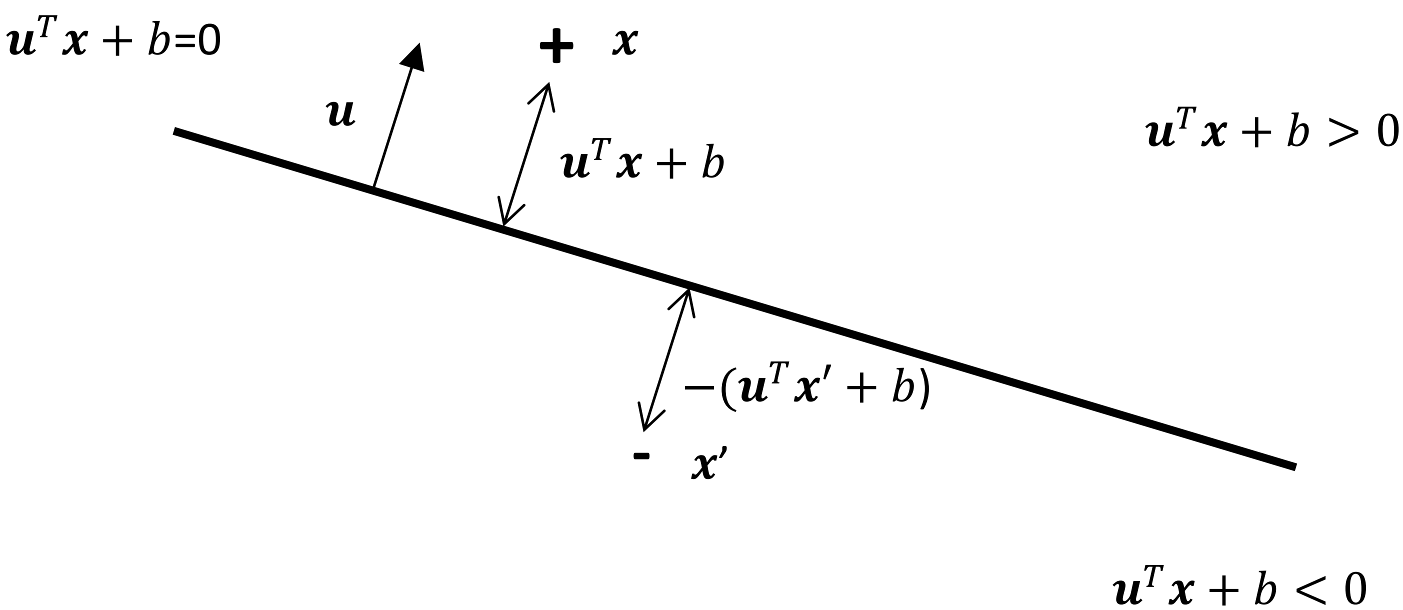
Prediction = $\mathrm{sgn}(\boldsymbol{u}^T\boldsymbol{x} + b)$



If we use a unit normal vector $\boldsymbol{u}$ represents the hyperplane, the distance between point $\boldsymbol{x}$ to plane is $|\boldsymbol{u}^T\boldsymbol{x} + b|$ or $y(\boldsymbol{u}^T\boldsymbol{x} + b)$

Recall: The geometry of a linear classifier

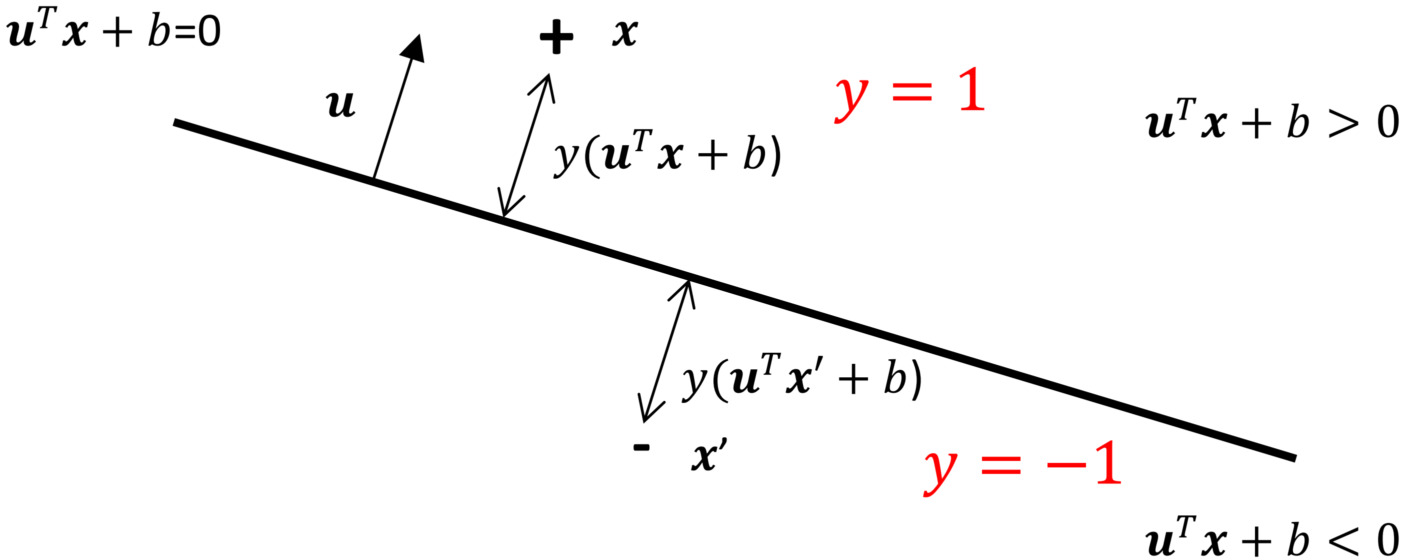
Prediction = $\mathrm{sgn}(\boldsymbol{u}^T\boldsymbol{x} + b)$



If we use a unit normal vector $\boldsymbol{u}$ represents the hyperplane, the distance between point $\boldsymbol{x}$ to plane is $|\boldsymbol{u}^T\boldsymbol{x} + b|$ or $y(\boldsymbol{u}^T\boldsymbol{x} + b)$

Recall: The geometry of a linear classifier

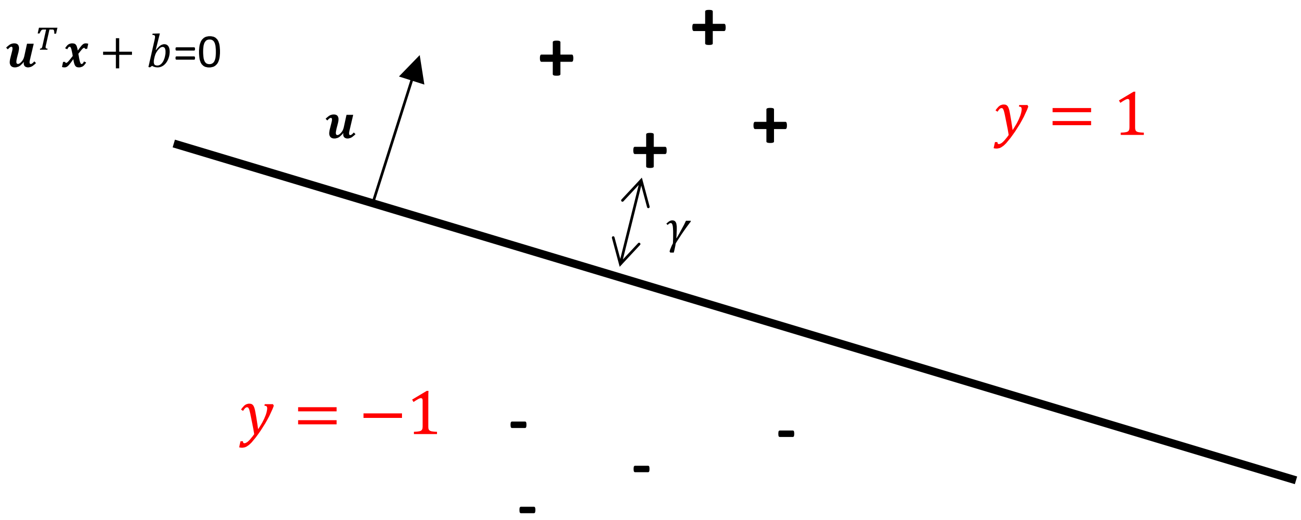
Prediction = sgn($\boldsymbol{u}^T\boldsymbol{x} + b$)



If we use a unit normal vector $\boldsymbol{u}$ represents the hyperplane, the distance between point $\boldsymbol{x}$ to plane is $|\boldsymbol{u}^T\boldsymbol{x} + b|$ or $y(\boldsymbol{u}^T\boldsymbol{x} + b)$

Recall: The geometry of a linear classifier

Prediction = sgn($\boldsymbol{u}^T\boldsymbol{x} + b$)



If we use a unit normal vector $\boldsymbol{u}$ represents the hyperplane, the distance between point $\boldsymbol{x}$ to plane is $|\boldsymbol{u}^T\boldsymbol{x} + b|$ or $y(\boldsymbol{u}^T\boldsymbol{x} + b)$

If the distance between the closest point in dataset D to the plane $\boldsymbol{u}$ is γ

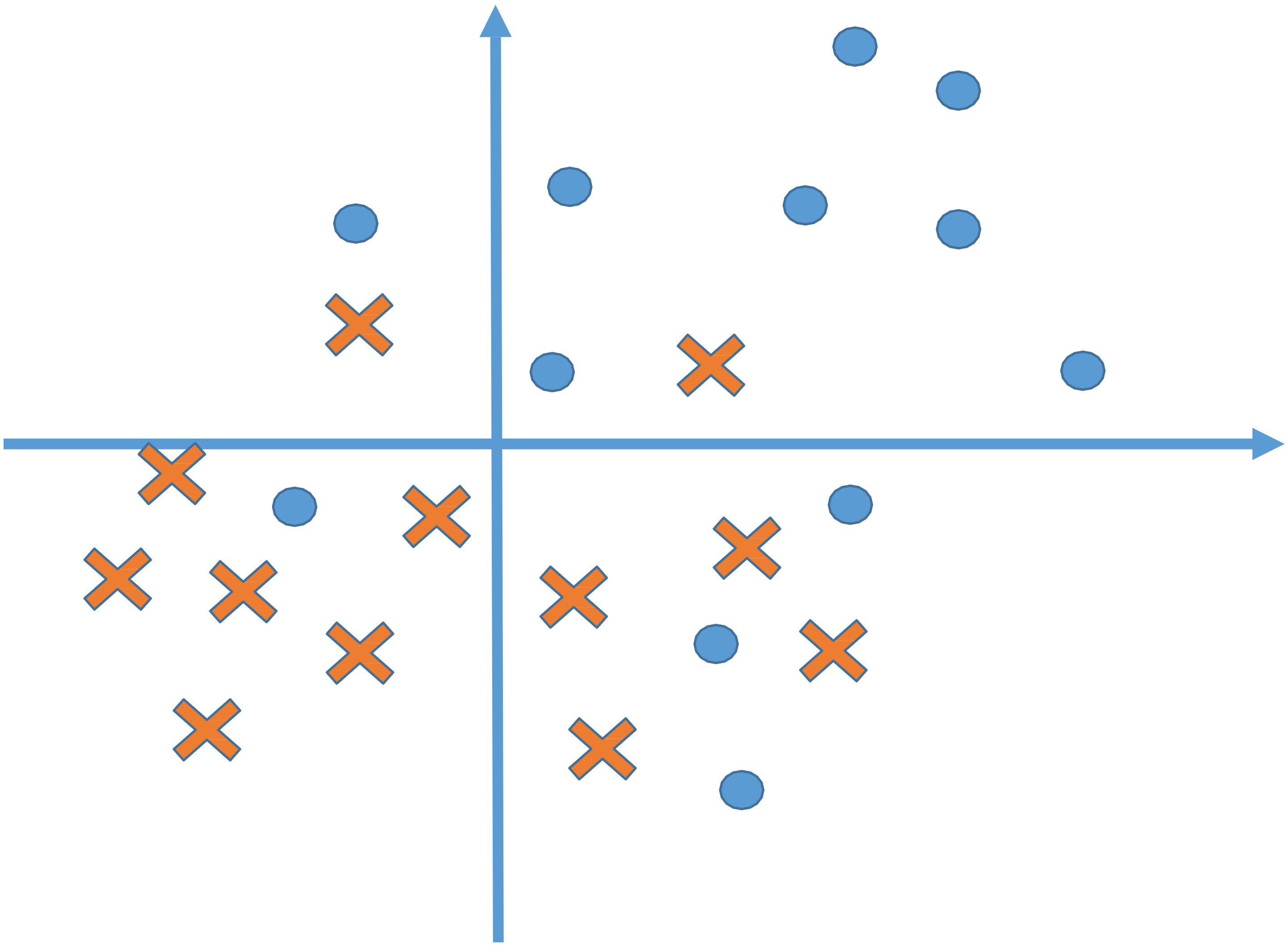
$$y_i(\boldsymbol{u}^T\boldsymbol{x_i} + b) \geq \gamma, \forall (x_i, y_i) \in D$$

Logistic regression

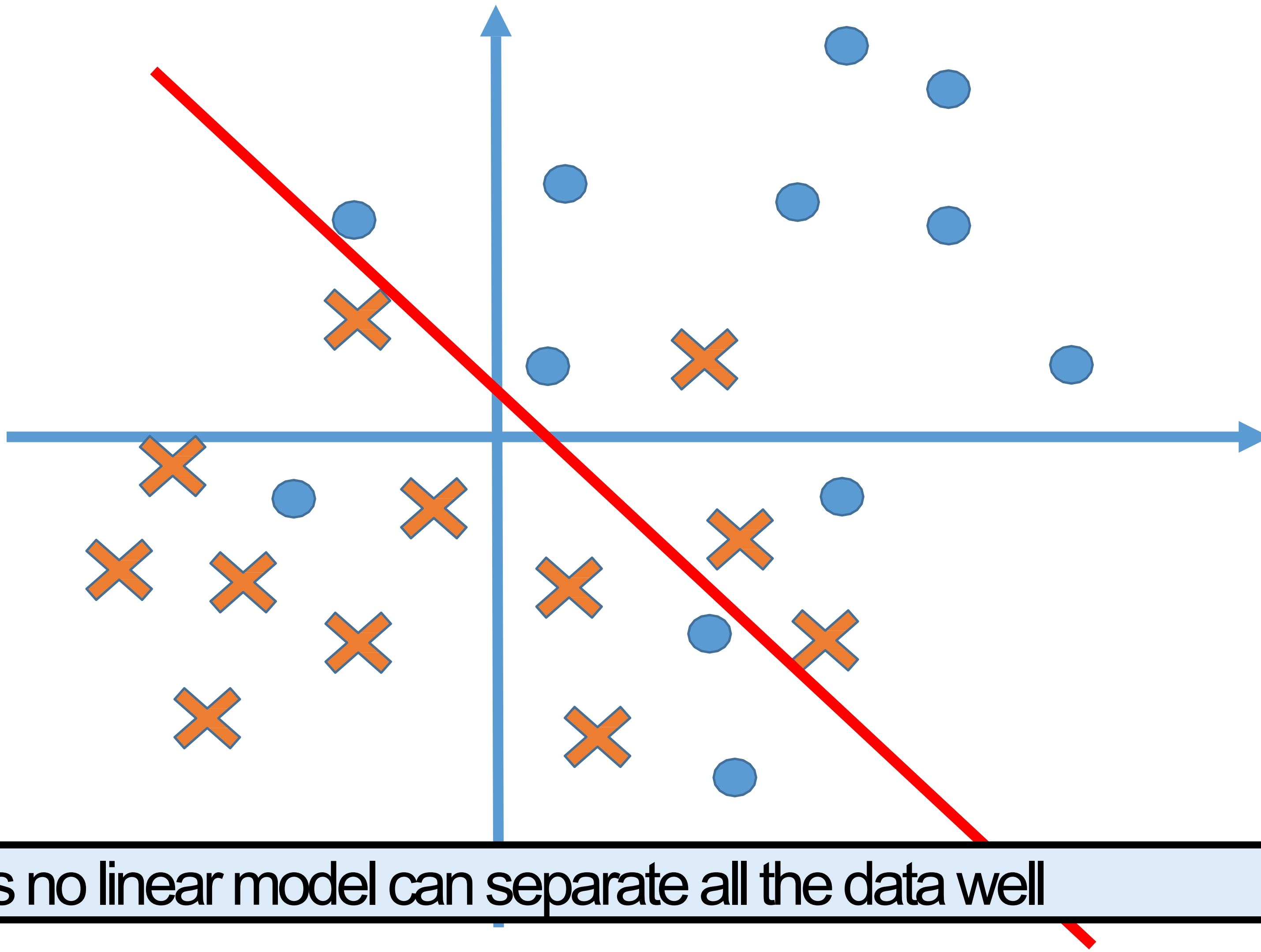
What you will learn today

- Logistic regression assumption
- Sigmoid function
- Maximum likelihood principle
- Optimization in ML
    - Stochastic gradient decent

What if data is not linearly separable?

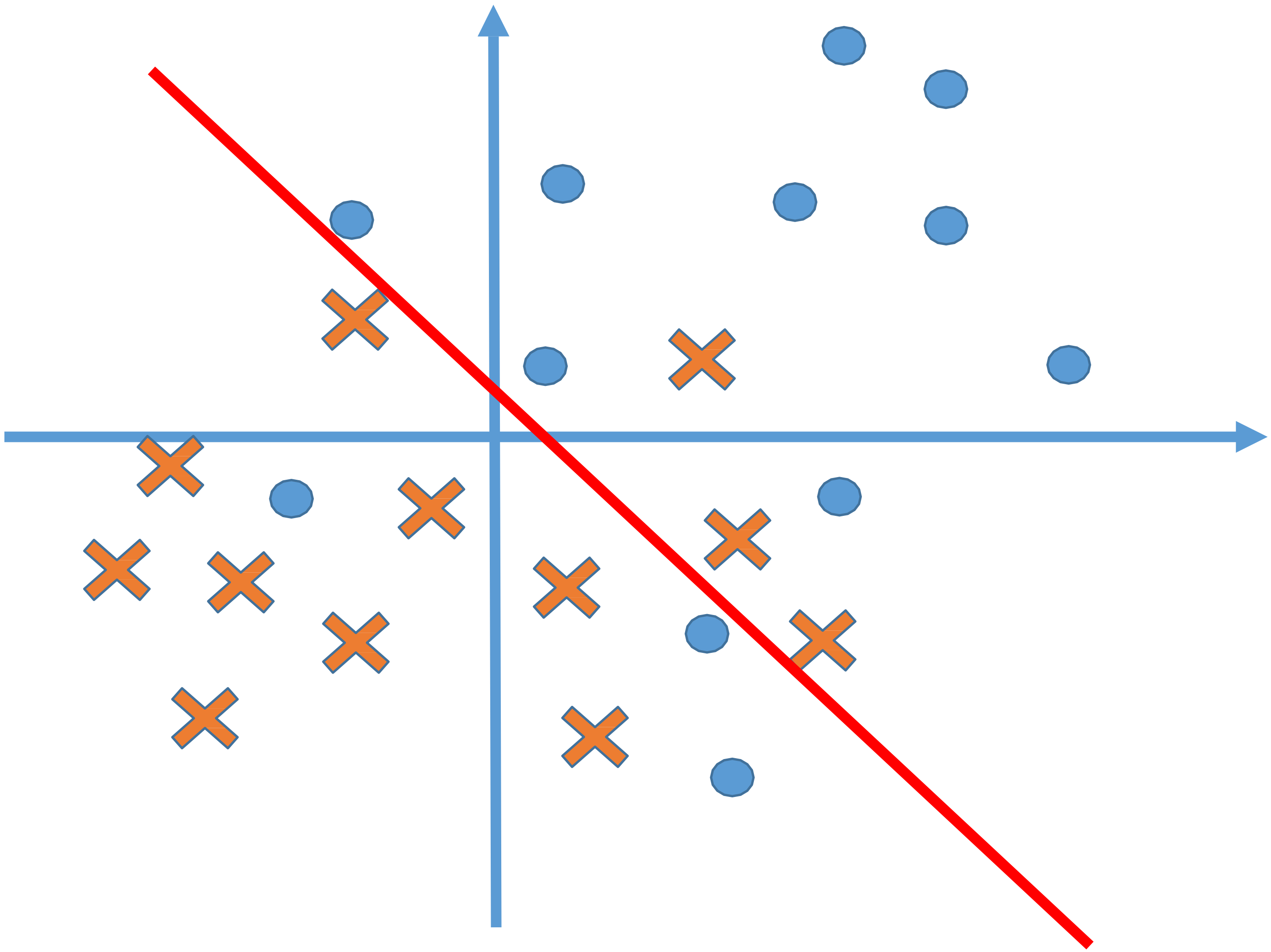


What if data is not linearly separable?



There is no linear model can separate all the data well

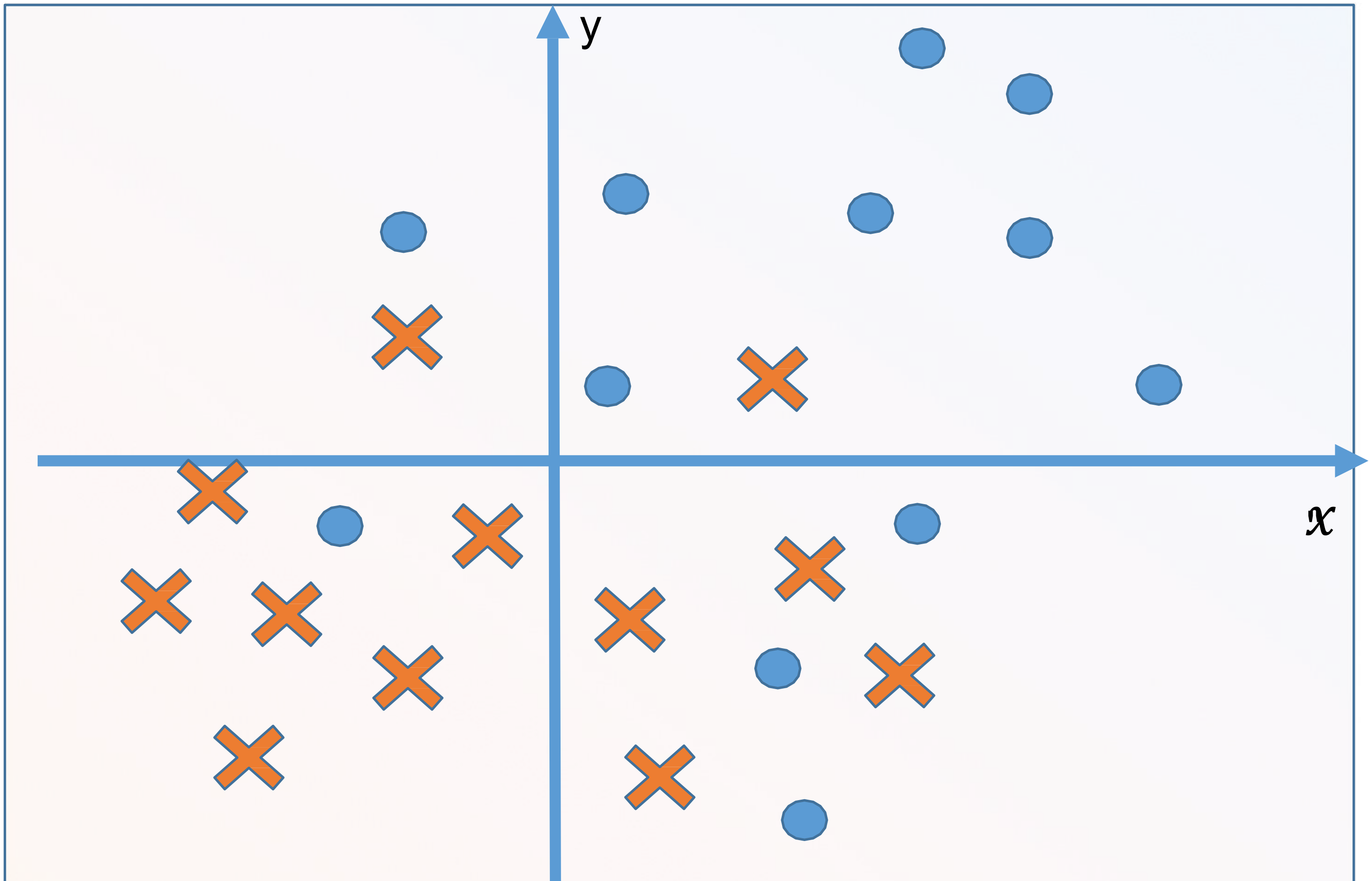
What Making data not linearly separable? [Discussion]



What Making data not linearly separable?

- Decision boundary is nonlinear
  - E.g., XOR example
- Noise in the training data
  - Outlier due to annotation errors
- Not enough features
- The natural of the prediction task
  - Patients with the same lab test results may not always have the same disease

What if data is not linearly separable?

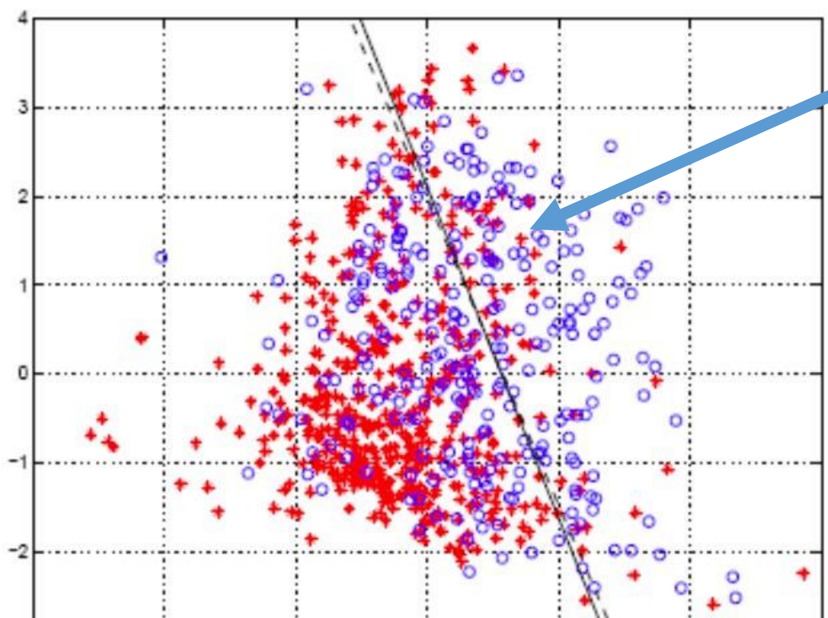


We consider a probabilistic model (today's lecture)

Modeling the Probability

Classification, but…

- The output $\mathbf{y}$ is a discrete value
- Instead of predicting the output label, let's predict $P(y = 1 \mid \mathbf{x})$

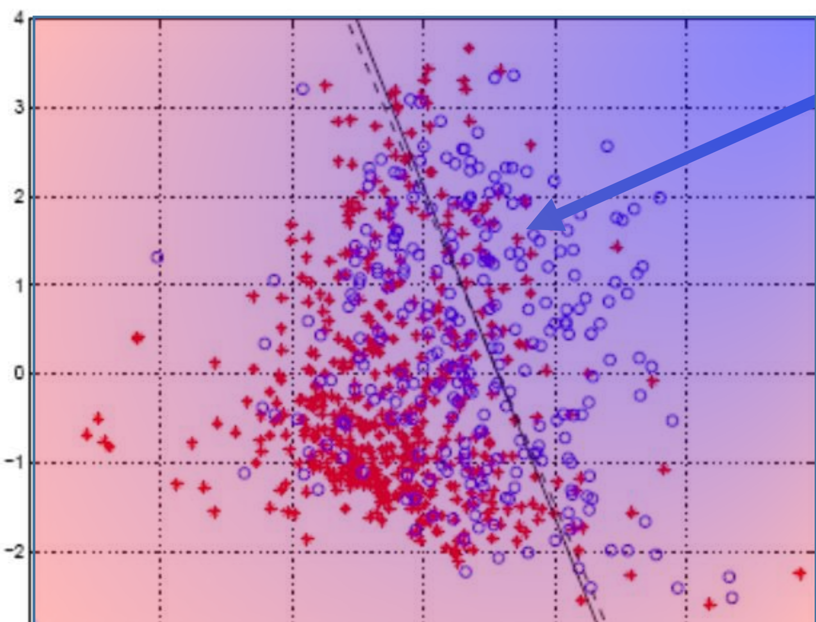


How likely the label y = 1 if my feature vector is in this region

Perceptron does not produce probability estimates

Classification, but…

- The output $\mathbf{y}$ is a discrete value
- Instead of predicting the output label, let us predict $P(y = 1 \mid \mathbf{x})$



How likely the label y = 1 if my feature vector is in this region

Perceptron does not produce probability estimates

Predict P(y = 1 | **x**)

Input: $x \in \mathbb{R}^d$

Output: $\mathrm{y} \in \{1, -1\}$

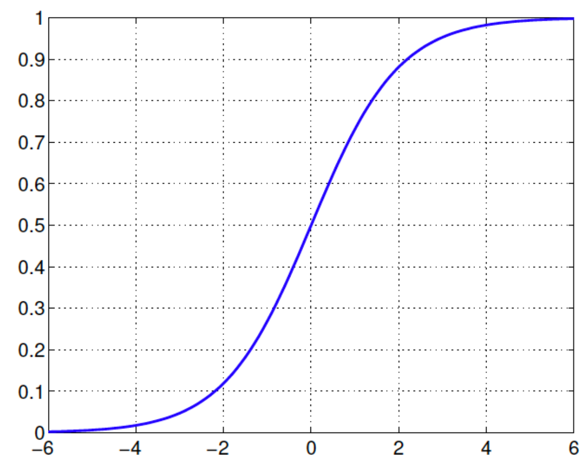
Build a model $h(x)$ such that

$$h(x) = \sigma(w^T x + b) \approx P(y = 1|x)$$

a regression problem

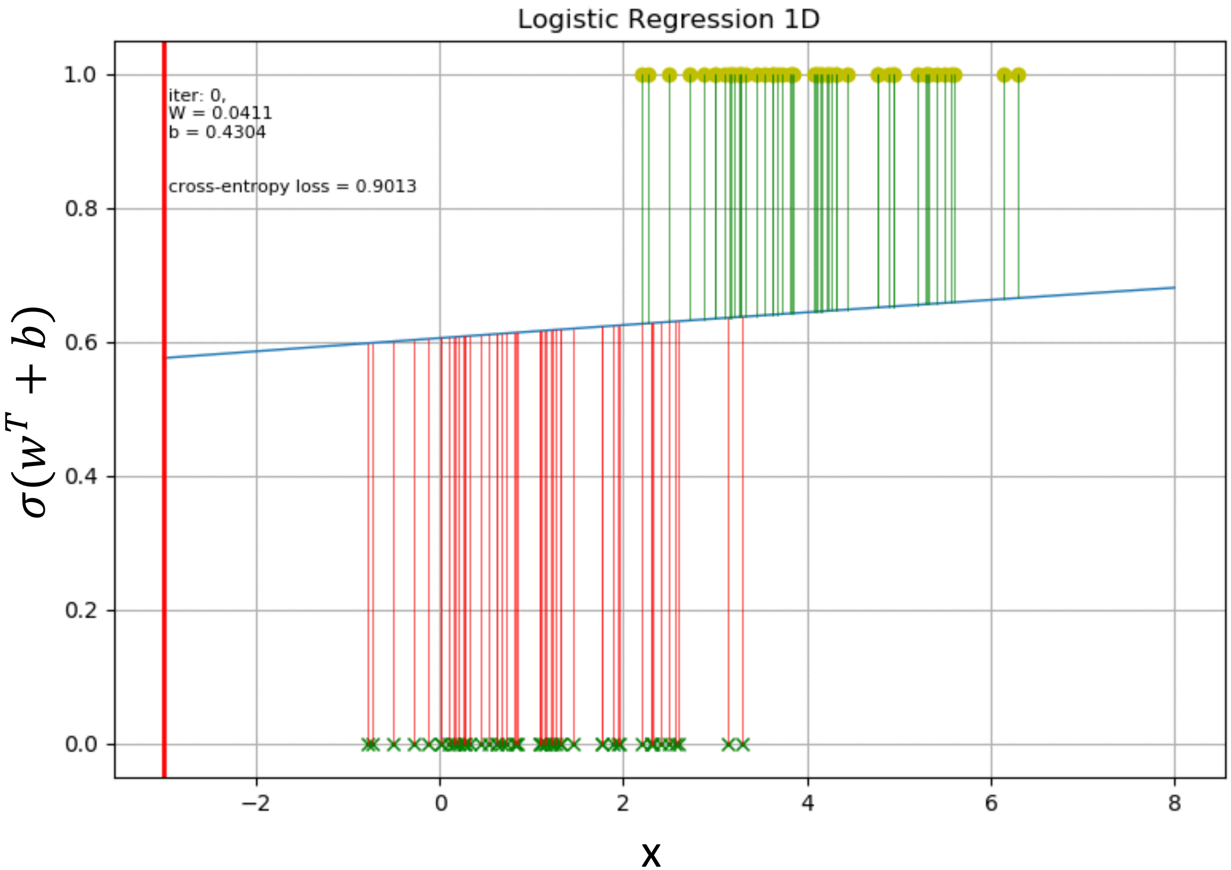
σ is a sigmoid function

$$\sigma(z) = \frac{\exp(z)}{1 + \exp(z)}$$
$$= \frac{1}{1+\exp(-z)}$$



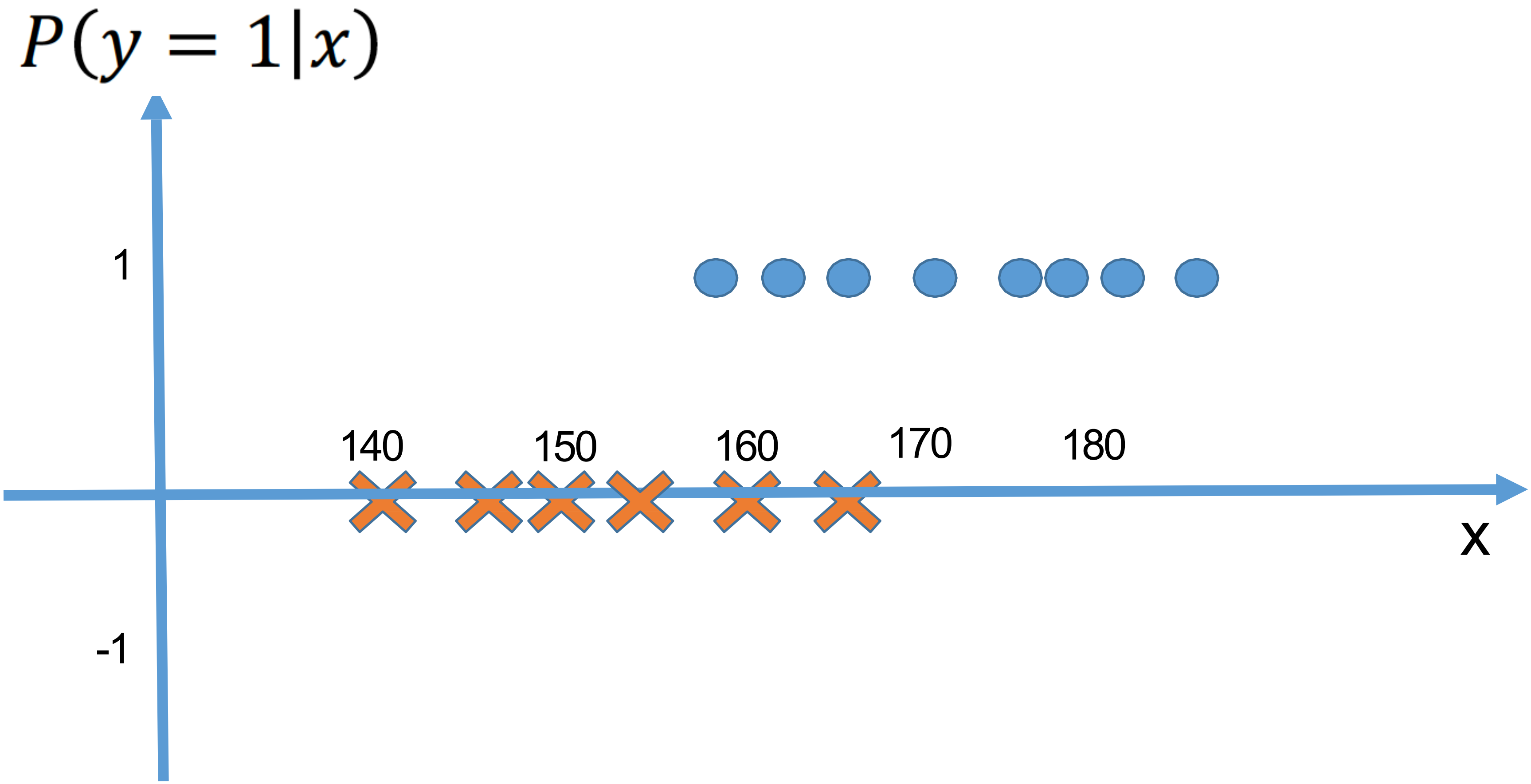
Logistic Regression in Action

$$\sigma(w^T x + b) \approx P(y = 1|x)$$



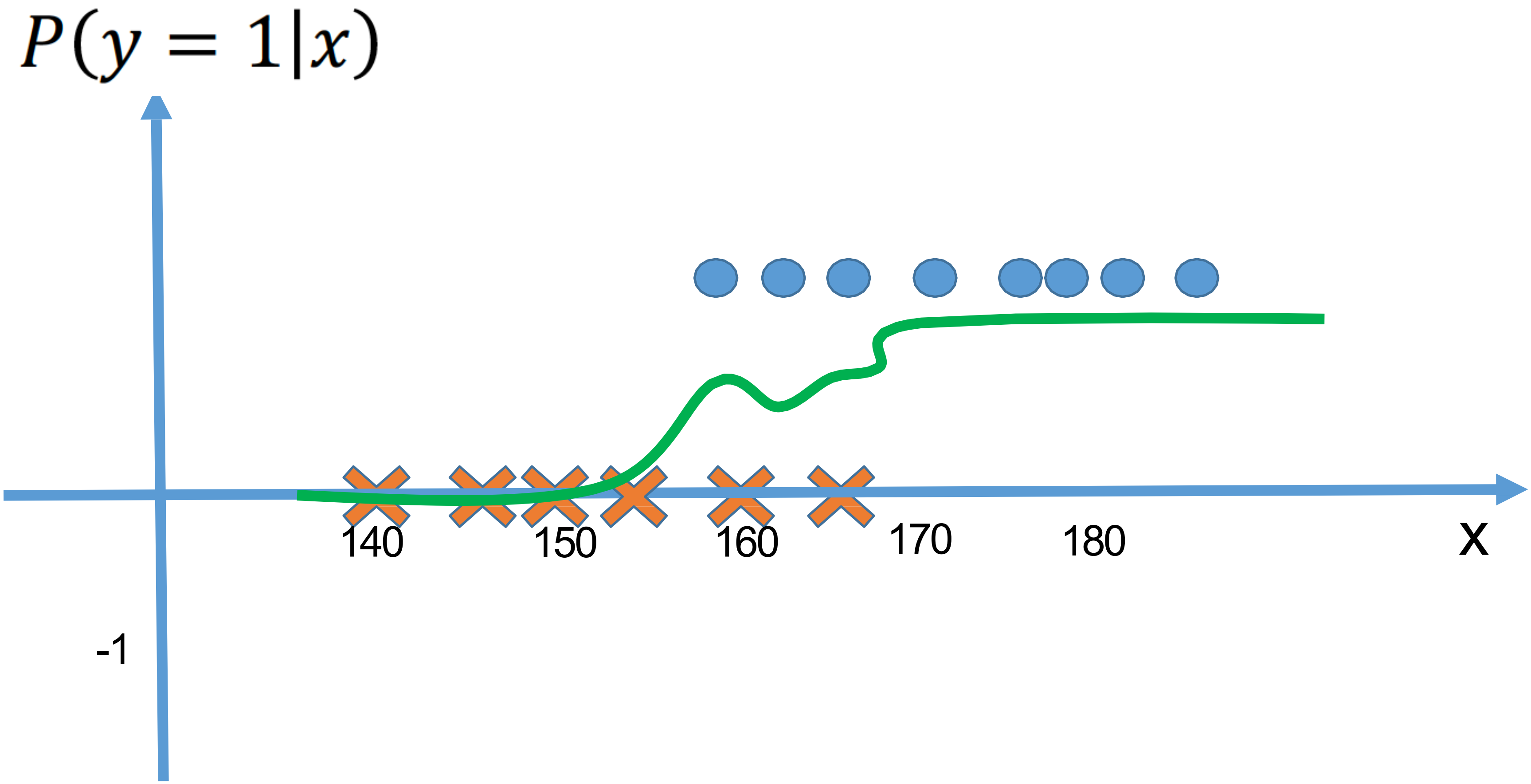
https://medium.com/swlh/from-animation-to-intuition-linear-regression-and-logistic-regression-f641a31e1caf

Why sigmoid function?



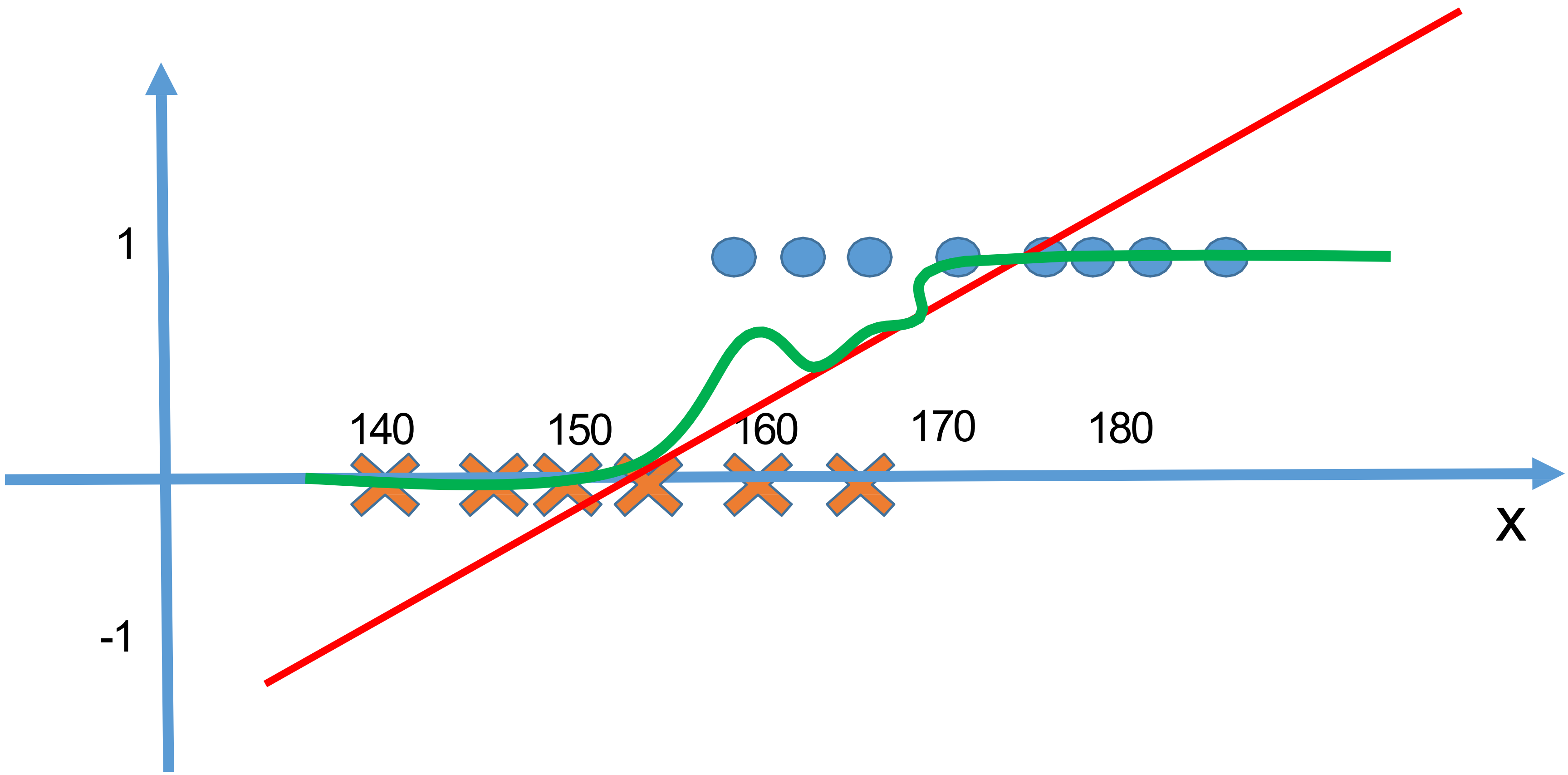
What is the σ function

What is the underlying target function?

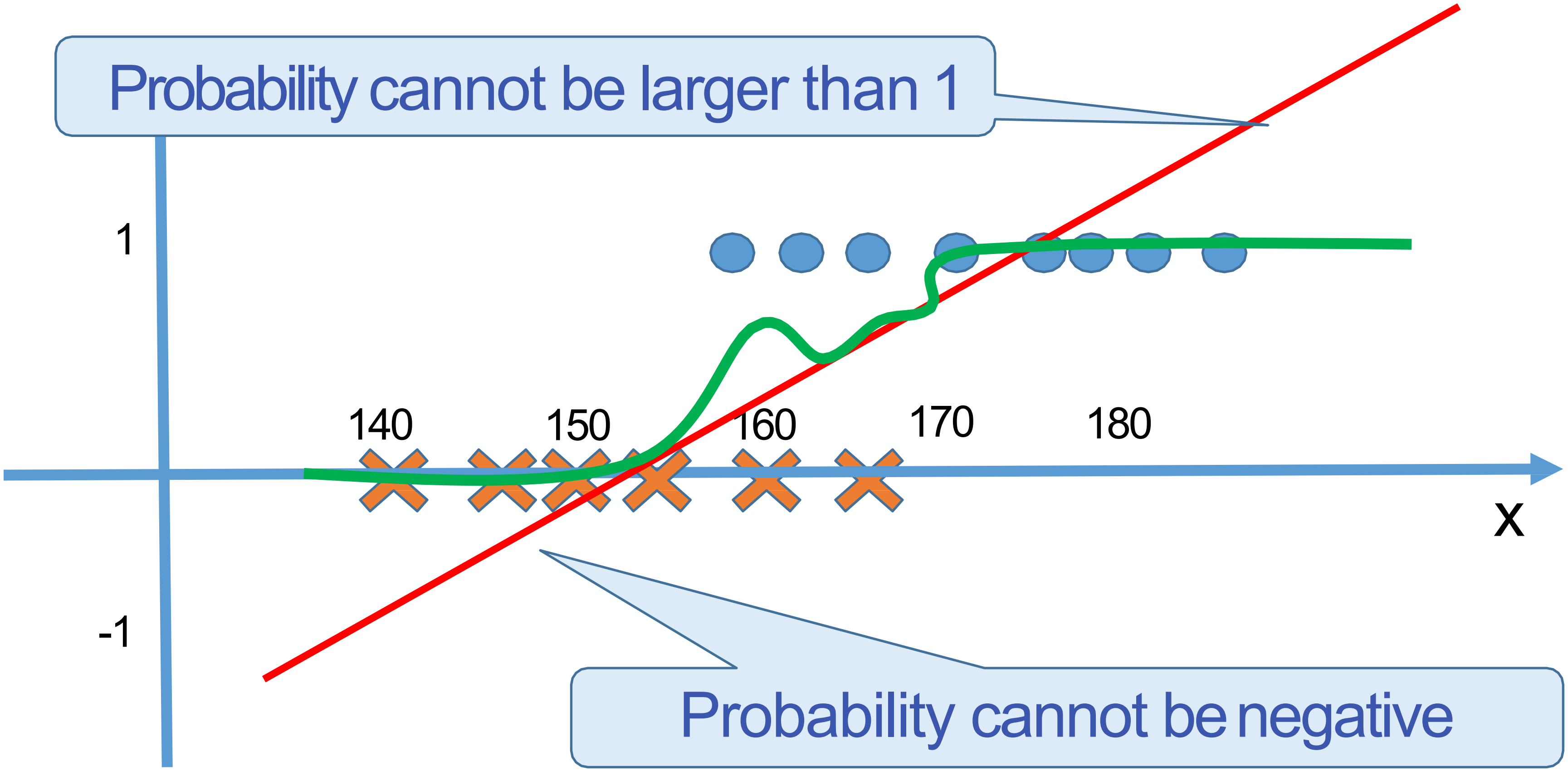


How to fit $P(y = 1|x)$

Can we fit it with a linear function?
$y = \mathbf{w}^T\mathbf{x} + b$



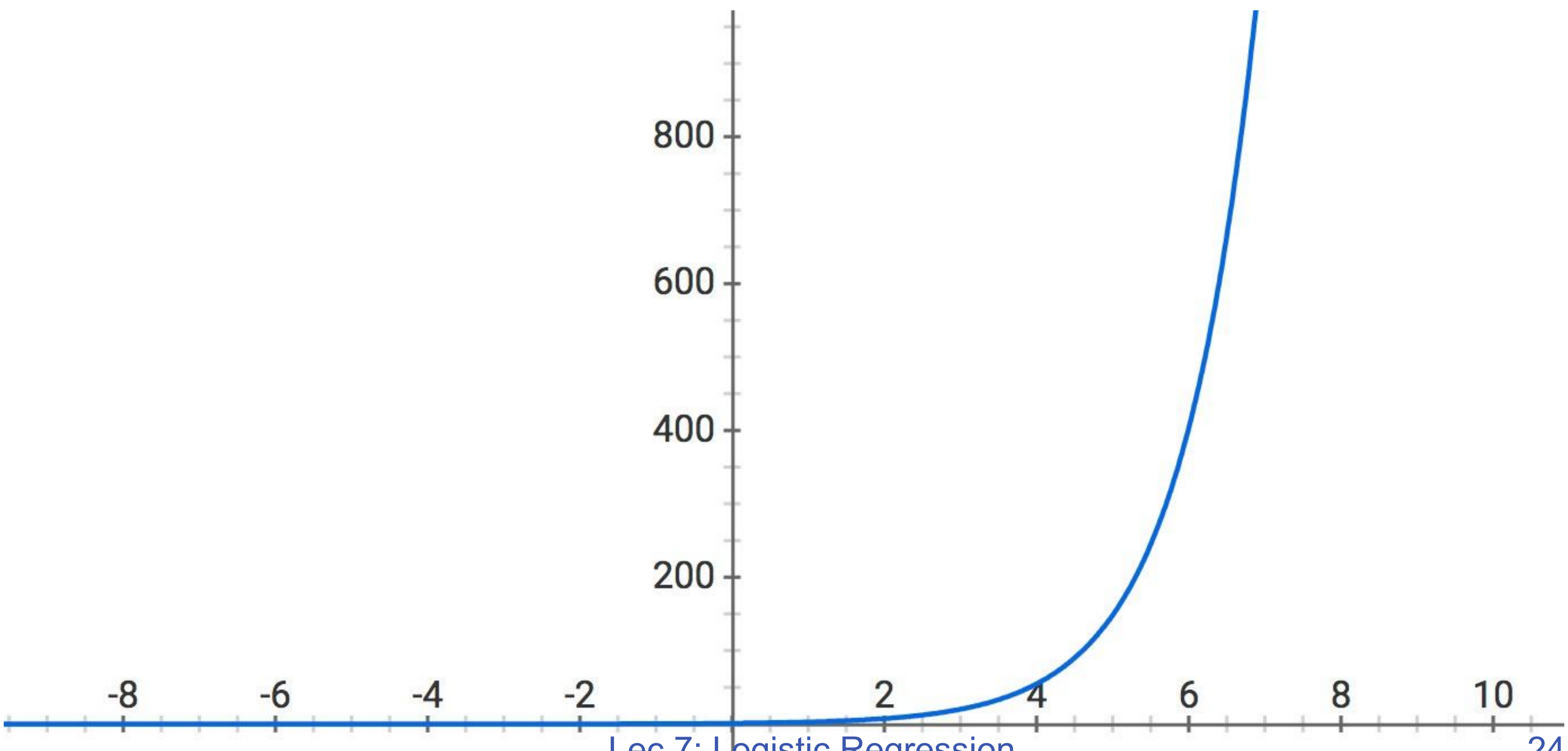
How to fit $P(y = 1|x)$



How can we design such a transformation function?

Idea 1: function always output positive value

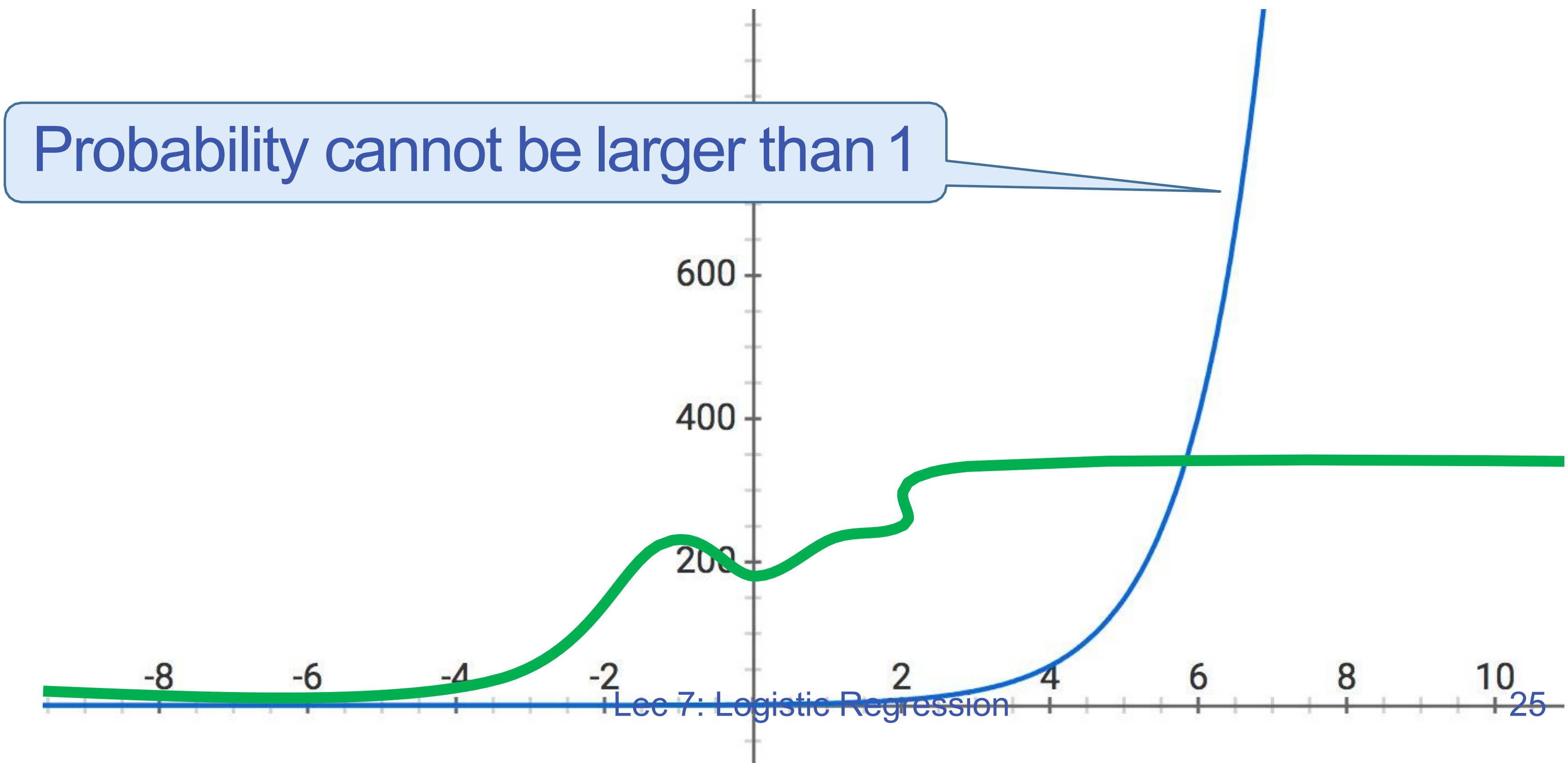
$\exp(\cdot)$ is always positive



Lec 6: Perceptron 22

How can we design such a transformation function?

- Idea 1: function always output positive value
  - $\exp(\cdot)$ is always positive
  - $\exp(w^T x + b)$ always return positive value



How can we design such a transformation function?

Idea 2: normalize the value such that it is less than 1

- $\exp(w^T x + b) \in (0, \infty)$ grows very fast, so we need to use exp to normalize itself
- Let's use

$$\sigma(w^T x + b) = \frac{\exp(w^T x + b)}{1 + \exp(w^T x + b)}$$

- When $w^T x + b \to \infty$,

$$\sigma(w^T x + b) \to 1$$

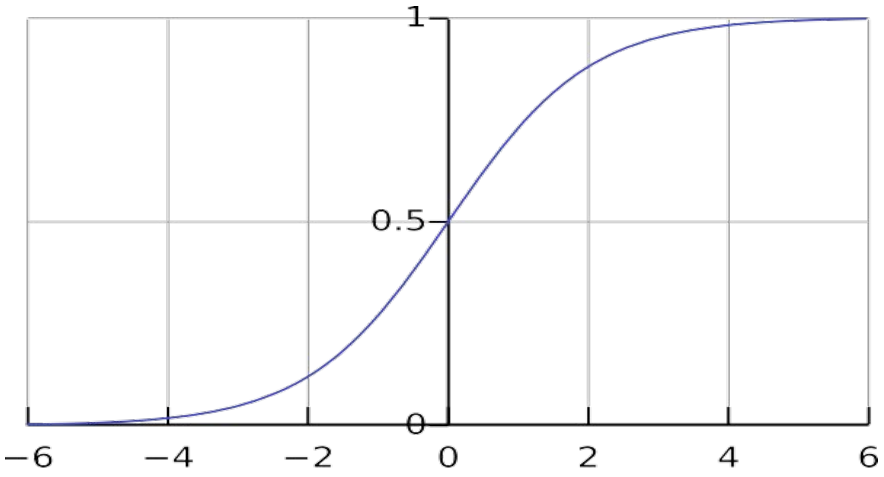
- When $w^T x + b \to -\infty$,

$$\sigma(w^T x + b) \to 0$$

The Sigmoid function

The $\sigma(z)$ function is called sigmoid function (or logistic function)

$$\sigma(z) = \frac{\exp(z)}{1+\exp(z)} = \frac{1}{1+\exp(-z)}$$



Summary (Modeling)

- What is the goal of logistic regression?
  - Model $P(y=1|x)$
- What is the hypothesis space?

$$H = \{\ \mathrm{h}\ |\ h : X \rightarrow P(Y\ |\ X), h(x) = \sigma\ (\ w^T x + b)\ \}$$

$$\sigma(z) = \frac{1}{1+\exp(-z)}$$

We want to find $h(x)$ such that

$$\boldsymbol{h(x)} \approx P(y = 1\ |x)$$

Decision Boundary of Logistic Regression

Predicting a label

$$P(y = 1|\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

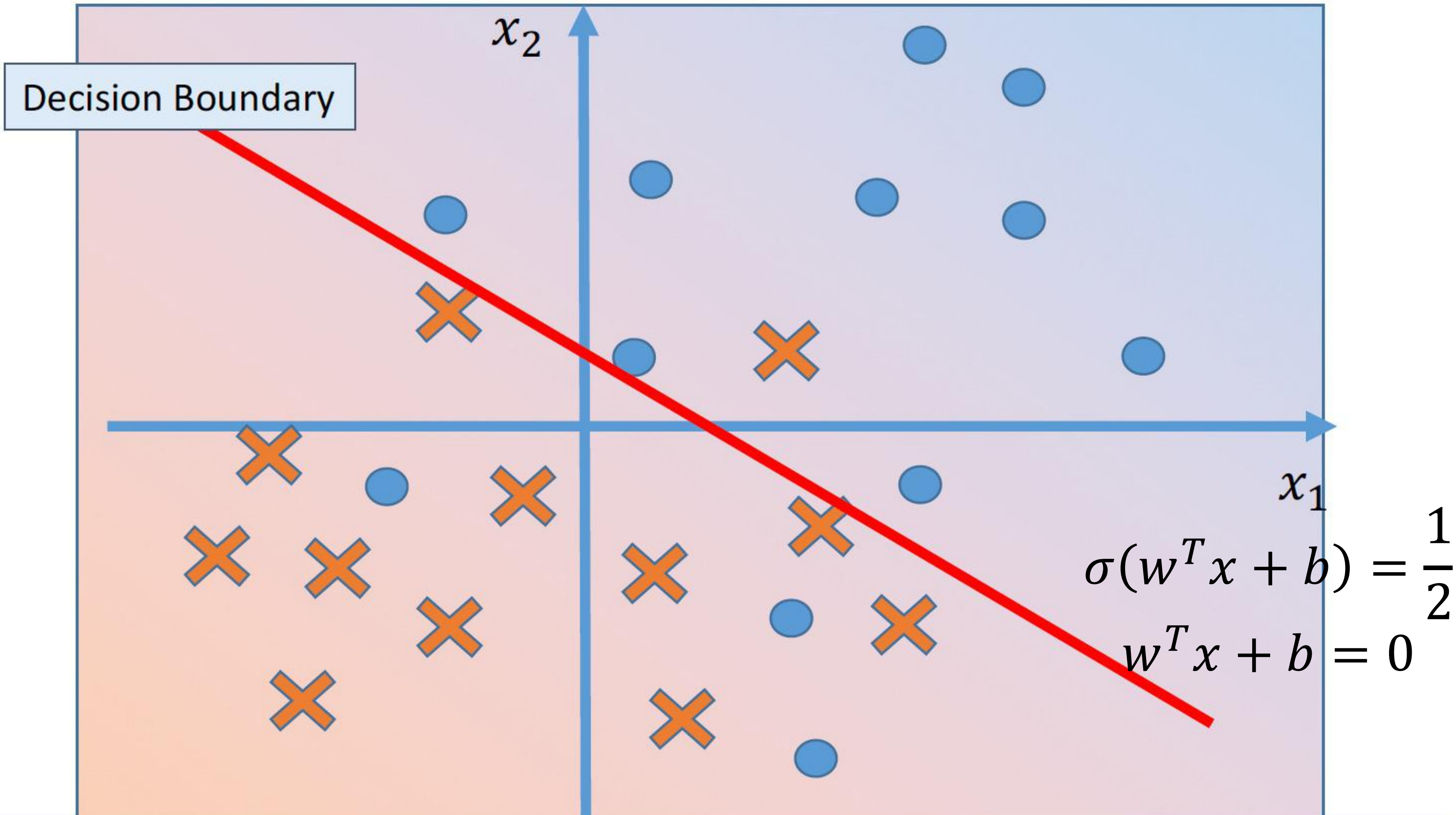
Compute $\sigma(w^T x)$;

If this is greater than half, predict 1
else predict -1

What does this correspond to in terms of $\mathbf{w}^T\mathbf{x}$?

$$w^T x = 0$$

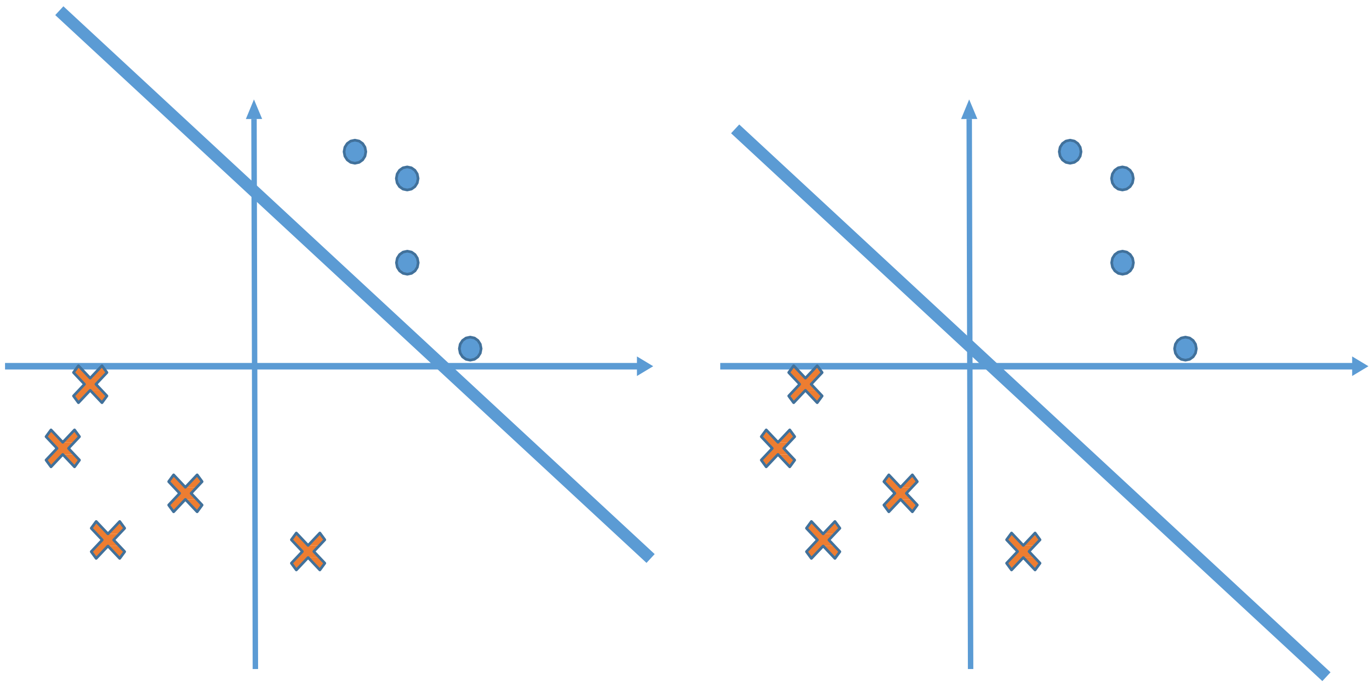
Prediction by logistic regression



A probabilistic model of modeling $\sigma(w^T x + b) = P(y = 1|x)$

Exercise

- Which function(s) are likely to be a decision function of logistic regression

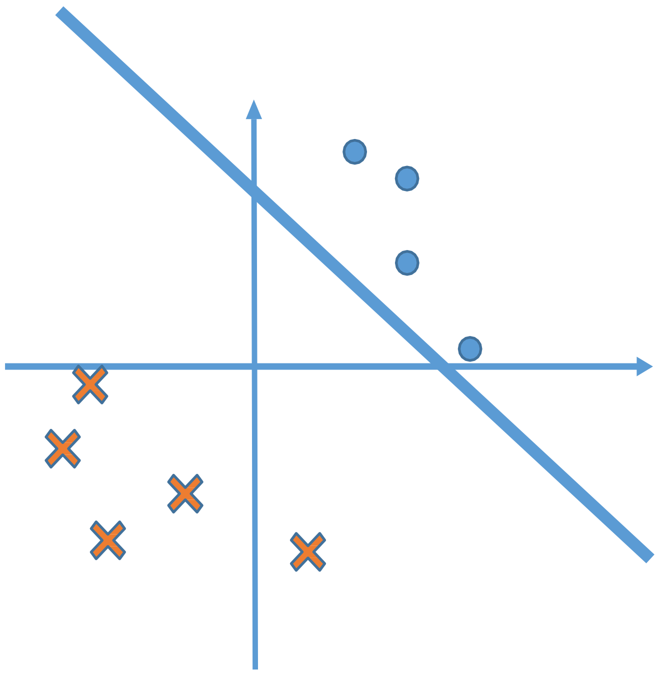


(A)

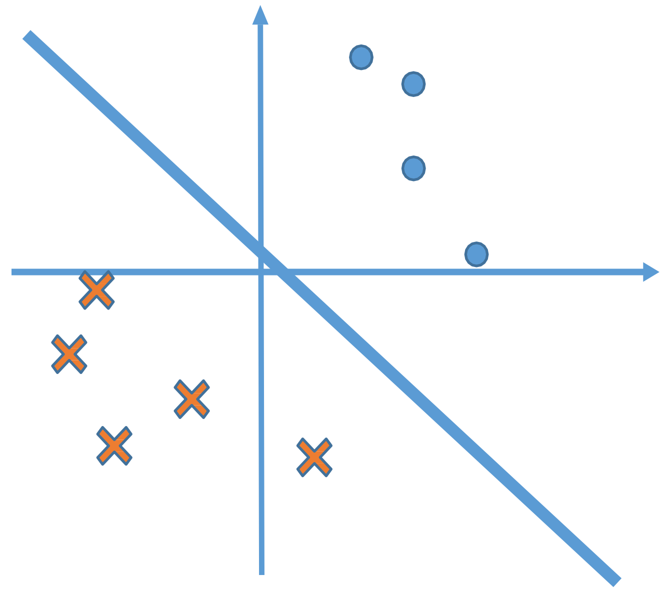
(B)

Exercise

- Which function(s) are likely to be a decision function of Perceptron



(A)



(B)

How to Train a Logistic Regression Model?

Logistic Regression: Setup

- The setting
  - Binary classification
  - Inputs: Feature vectors $x \in R^N$
  - Labels: $\mathbf{y} \in \{\mathbf{-1}, \mathbf{+1}\}$
- Training data
  - S = $\{(\mathbf{x}_i, \mathbf{y}_i)\}$, m examples
- Hypothesis space

$$H = \{ \text{ h} \mid h : X \to P(Y \mid X), h(x) = \sigma (w^T x + b) \}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Logistic Regression: Setup

- Training data
  - S = $\{(\mathbf{x}_i, \mathbf{y}_i)\}$, m examples
- Hypothesis space

$$H = \{\ \mathrm{h} \mid h : X \to Y, h(x) = \sigma\,(\,w^T x + b)\,\}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

- Learning Goal:
  - Find an $\mathrm{h} \in H$, such that $\mathrm{h(x)} \approx P(y = 1 \mid x)$
  - How to?

Maximum Likelihood

Which bag of words more likely generate:

aDaaa

Maximum Likelihood

Which bag of words more likely generate:

aDaaa

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

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

Drawing color cards from the envelope

- Let say we have several cards in the envelope
- Assume

$$P(card = yellow) = \theta$$

$$P(card = purple) = 1 - \theta$$

Drawing color cards from the envelope

- Sample with replacement n times
- k times we get yellow card, and n-k times, we get purple card
- The joint probability (likelihood)

$$C_K^N \ \theta^k(1-\theta)^{n-k}$$

- What is the best θ making the joint probability maximal?

Drawing color cards from the envelope

$$\cancel{C_K^N}\ \theta^k(1-\theta)^{n-k}$$

The usual trick: Convert products to sums by taking log

Recall that this works only because log is an increasing function and the maximizer will not change

- Solving $\max_\theta \theta^k(1-\theta)^{n-k}$
- Equivalently, we can solve

$$\max_\theta \log(\theta^k(1-\theta)^{n-k})$$

$$\max_\theta \ k \log \theta + (n-k)\log(1-\theta)$$

- At the optimum,

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

Drawing color cards from the envelope

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

$$\Rightarrow \frac{k}{\theta} - \frac{n-k}{1-\theta} = 0$$

For this simple problem, we have a closed-form solution.
We are not always lucky like this

$$\Rightarrow \theta(n-k) = (1-\theta)k$$

$$\Rightarrow \theta = \frac{k}{n}$$

Maximum Likelihood Estimator (formal definition)

Likelihood function of parameters

Let $X_1, \ldots, X_N$ be IID (independent and identically distributed) 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 Estimation

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.

Back to Logistic regression

Training data

S={($\mathbf{x}_i$, $\mathbf{y}_i$)}, m examples

Hypothesis space

$$H = \{\ \mathrm{h} \mid h : X \rightarrow Y, h(x) = \sigma\,(\,w^T x + b)\,\}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Find an h $\in H$, such that $\mathrm{h(x)} \approx P(y = 1 \mid x)$

Back to Logistic regression

Training data

S={($\mathbf{x}_i$, $\mathbf{y}_i$)}, m examples

Hypothesis space

$$H = \{ \text{ h } | \, h : X \rightarrow Y, h(x) = \sigma\,(\,w^T x + b)\,\}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Find an h $\in H$, such that $\mathrm{h(x)} \approx P(y = 1|\,x)$

How to? Maximum Likelihood Estimator

$$\mathrm{P(y = 1|x;\ w, b)}$$

Likelihood function

$P(card = yellow) = \theta$

$P(card = purple) = 1 - \theta$

Find θ by maximizing $L(D;\theta)$

Logistic regression

Find w, b by maximizing $P(S; w, b)$

Likelihood function

$P(card = yellow) = \theta$

$P(card = purple) = 1 - \theta$

Find θ by maximizing $L(D;\theta)$

Logistic regression

Find w, b by maximizing $P(S; w, b)$

$$\text{argmax}_{w,b} P(S; w, b) = \text{argmax}_{w,b} \prod_{i=1}^{m} P(y_i | x_i; w, b)$$

Maximum likelihood estimator for logistic regression

$$\text{argmax}_{w,b} P(S; w, b) = \text{argmax}_{w,b} \prod_{i=1}^{m} P(y_i | x_i; w, b)$$

$\boxed{\text{Equivalent to solve}}$

$$\text{argmax}_{w,b} \sum_{i=1}^{m} \log P(y_i | x_i; w, b)$$

Remember our assumption:

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

$$P(y = -1 | x; w, b) = 1 - \sigma(w^T x + b) = 1 - \frac{1}{1 + \exp\left(-(w^T x + b)\right)}$$

$$= \frac{\exp\left(-(w^T x + b)\right)}{1 + \exp\left(-(w^T x + b)\right)} = \frac{1}{1 + \exp\left((w^T x + b)\right)} = \sigma(-(w^T x + b))$$

Maximum likelihood estimator for logistic regression

$$\text{argmax}_{w,b} P(S; w, b) = \text{argmax}_{w,b} \prod_{i=1}^{m} P(y_i | x_i; w, b)$$

$$\boxed{\text{Equivalent to solve}}$$

$$\text{argmax}_{w,b} \sum_{i=1}^{m} \log P(y_i | x_i; w, b)$$

Remember our assumption:

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

$$P(y_i | x_i; w, b) = \begin{cases} \sigma(w^T x_i + b) \\ \sigma(-(w^T x_i + b)) \end{cases} \quad \Rightarrow P(y_i | x_i; w, b) = \sigma(y_i(w^T x_i + b))$$

Maximum likelihood estimator for logistic regression

$$\text{argmax}_{w,b} P(S; w, b) = \text{argmax}_{w,b} \prod_{i=1}^{m} P(y_i | x_i; w, b)$$

Equivalent to solve

$$\text{argmax}_{w,b} \sum_{i=1}^{m} \log P(y_i | x_i; w, b)$$

Using $P(y_i | x_i; w, b) = \sigma(y_i(w^T x_i + b))$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

$$\text{argmax}_{w,b} \sum_{i=1}^{m} \log \sigma(y_i(w^T x_i + b))$$

$$= -\sum_{i=1}^{m} \log(1 + \exp(-y_i(w^T x_i + b)))$$

How to Optimize the Loss?

How to minimizing the loss

- Optimization methods
  - Gradient Descent
  - Stochastic Gradient Descent
  - Analytic solution

  - …many other approaches

How to solve it?

$$\mathrm{argmax}_{w,b} - \sum_{i=1}^{m} \log(1 + \exp(-y_i(w^T x_i + b)))$$

There is no closed-form solution

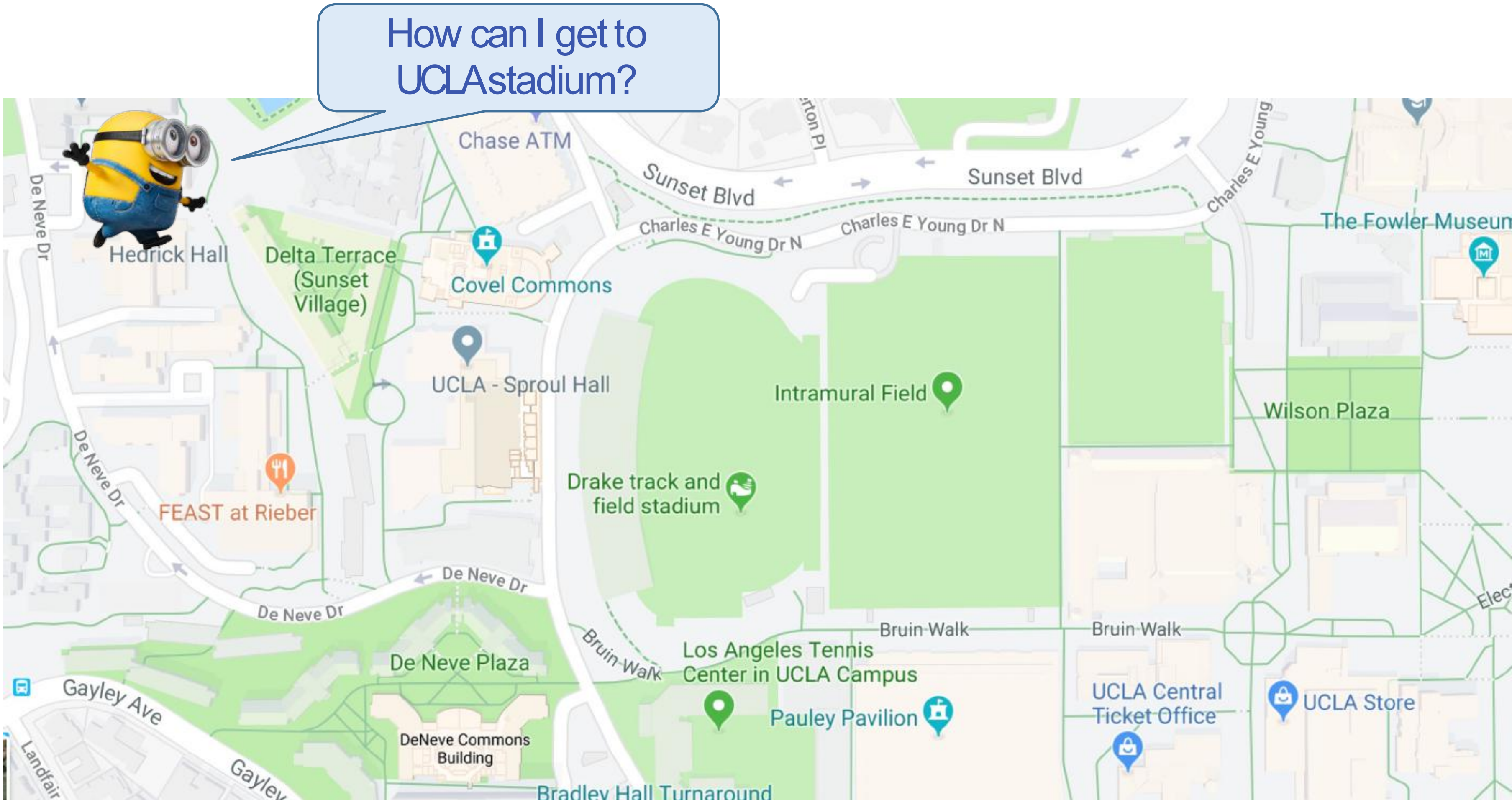
Max f(x) is equivalent to min - f(x)
=>only need to consider minimization problems.

One way to solve it is by gradient descent

Gradient Descent

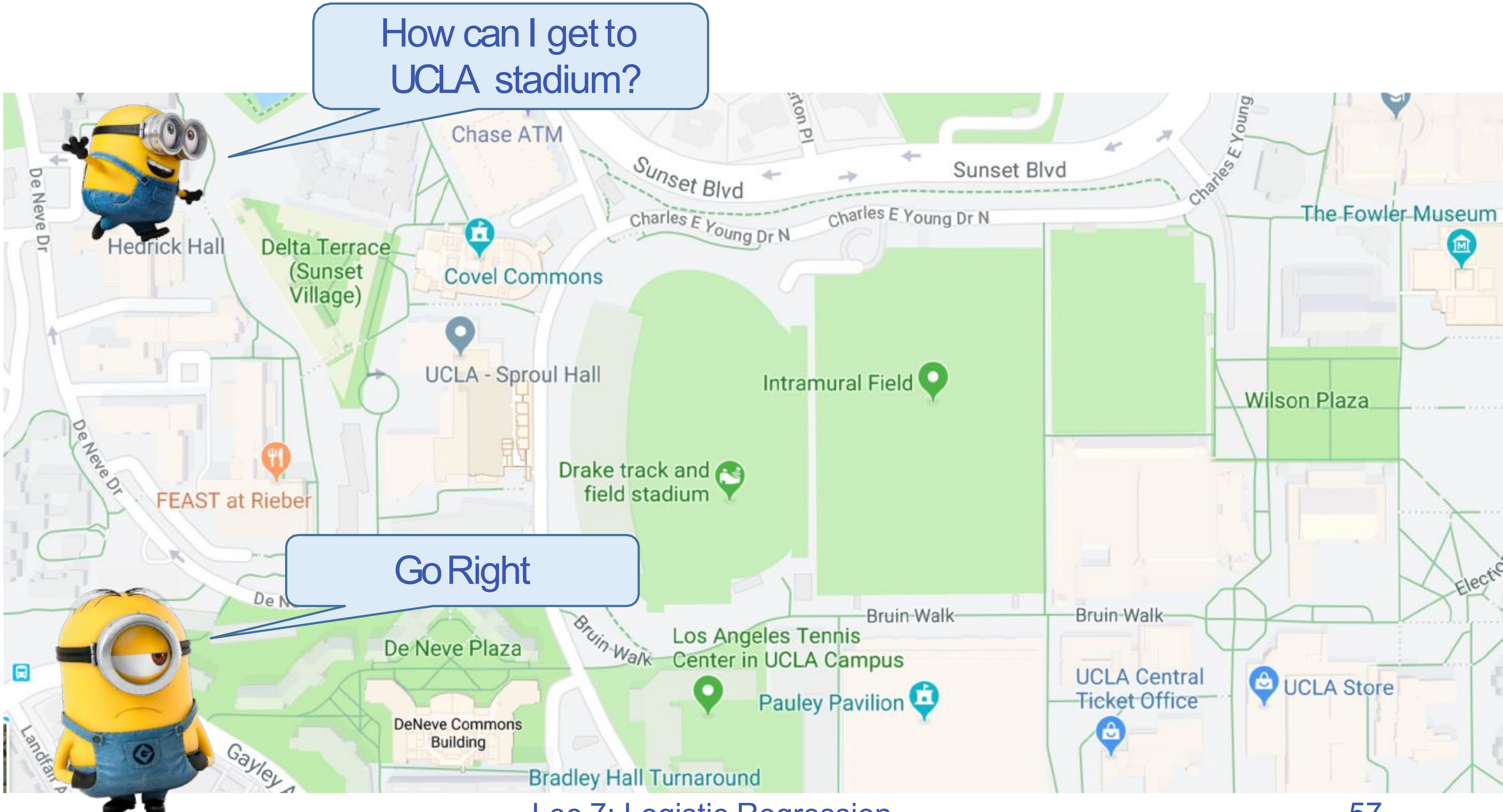
Intuition

Asking direction



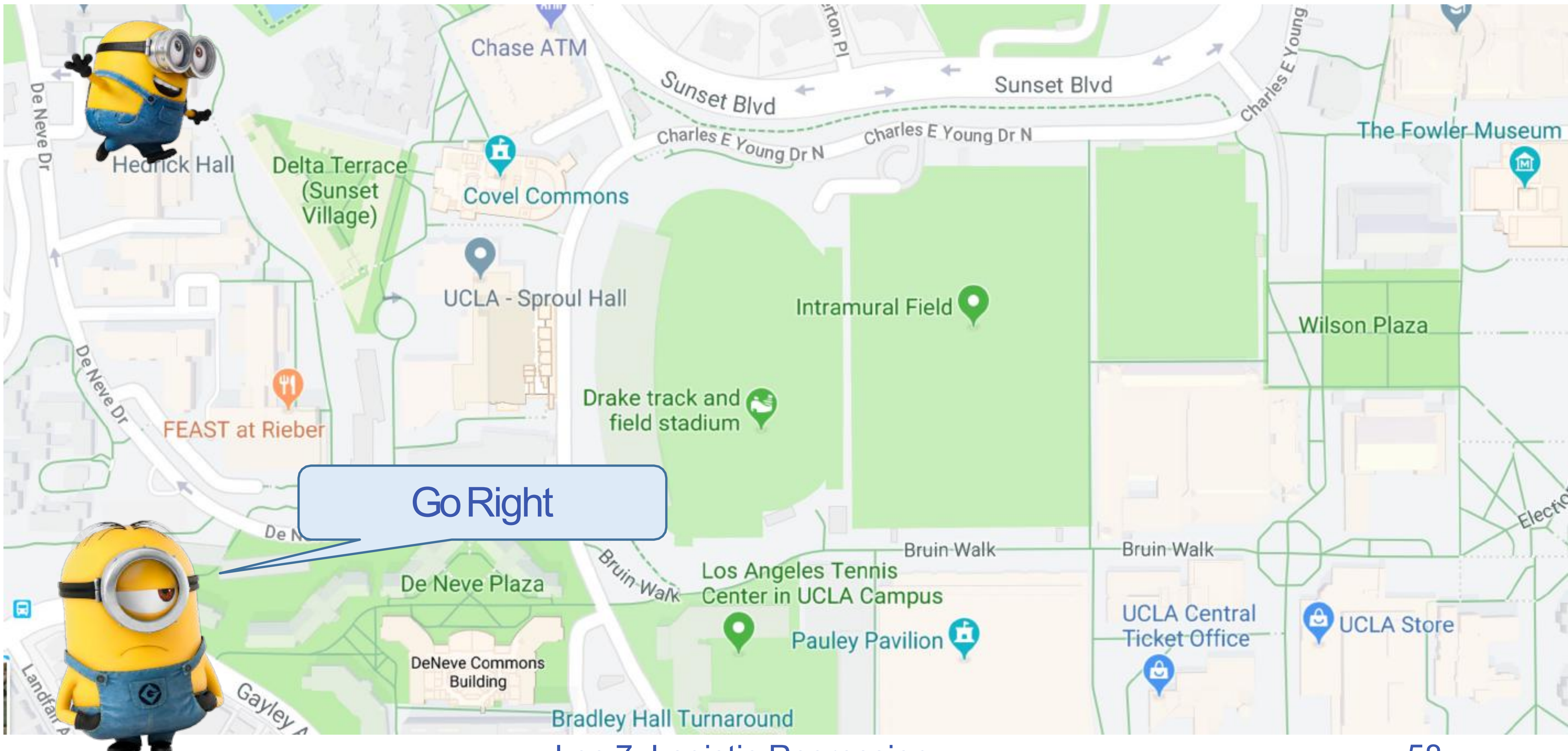
Intuition

Asking direction



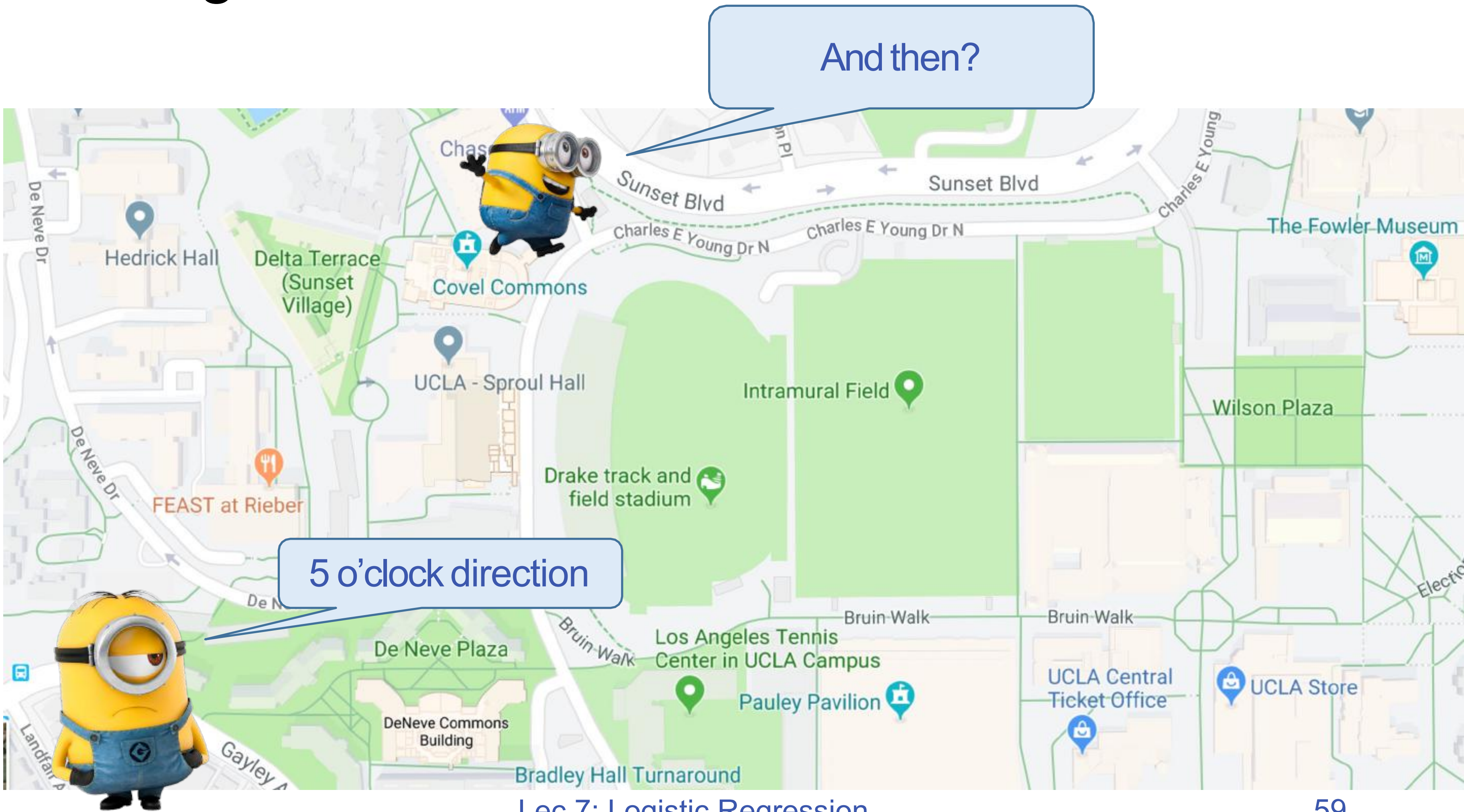
Intuition

Asking direction



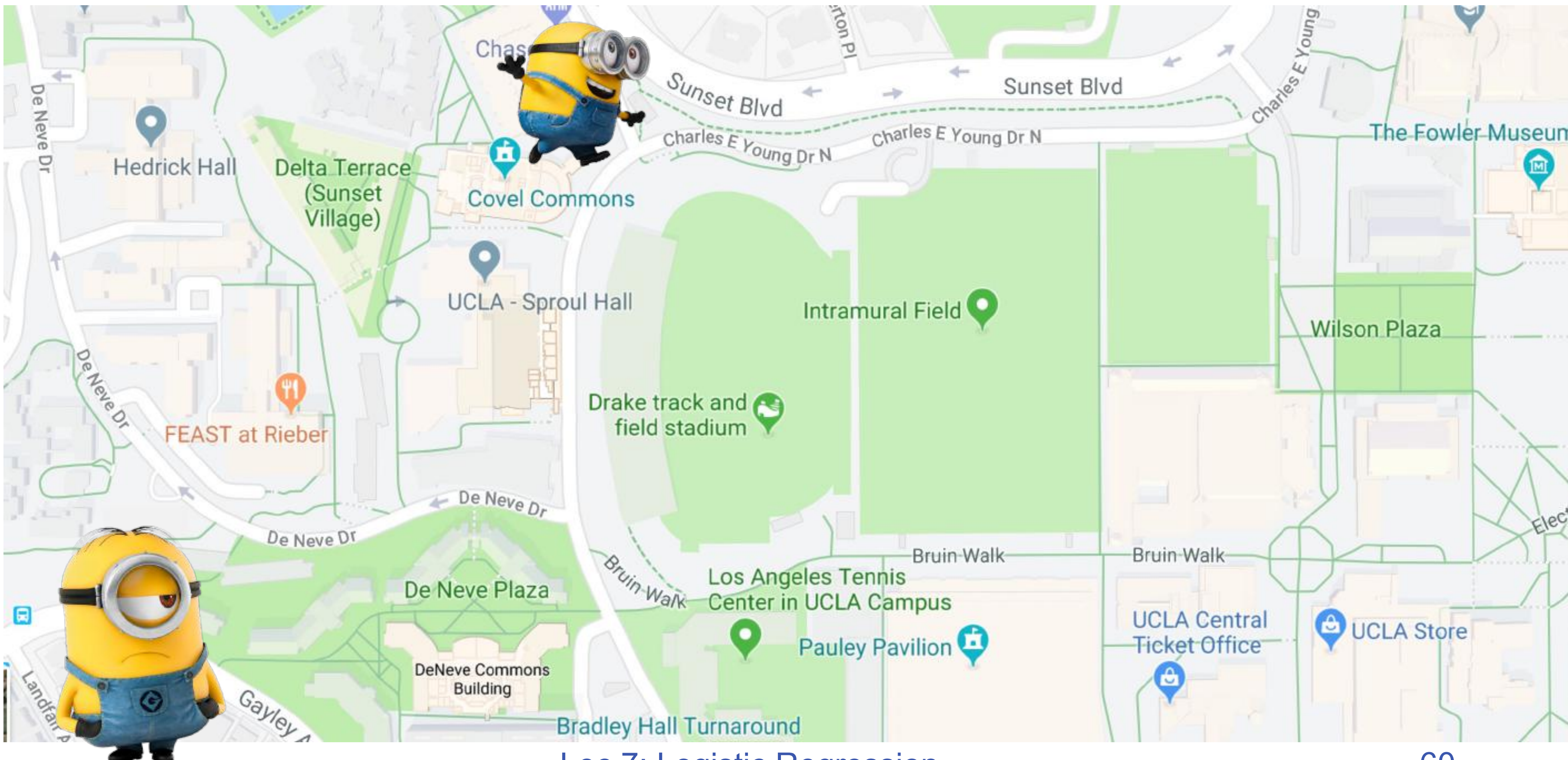
Intuition

Asking direction



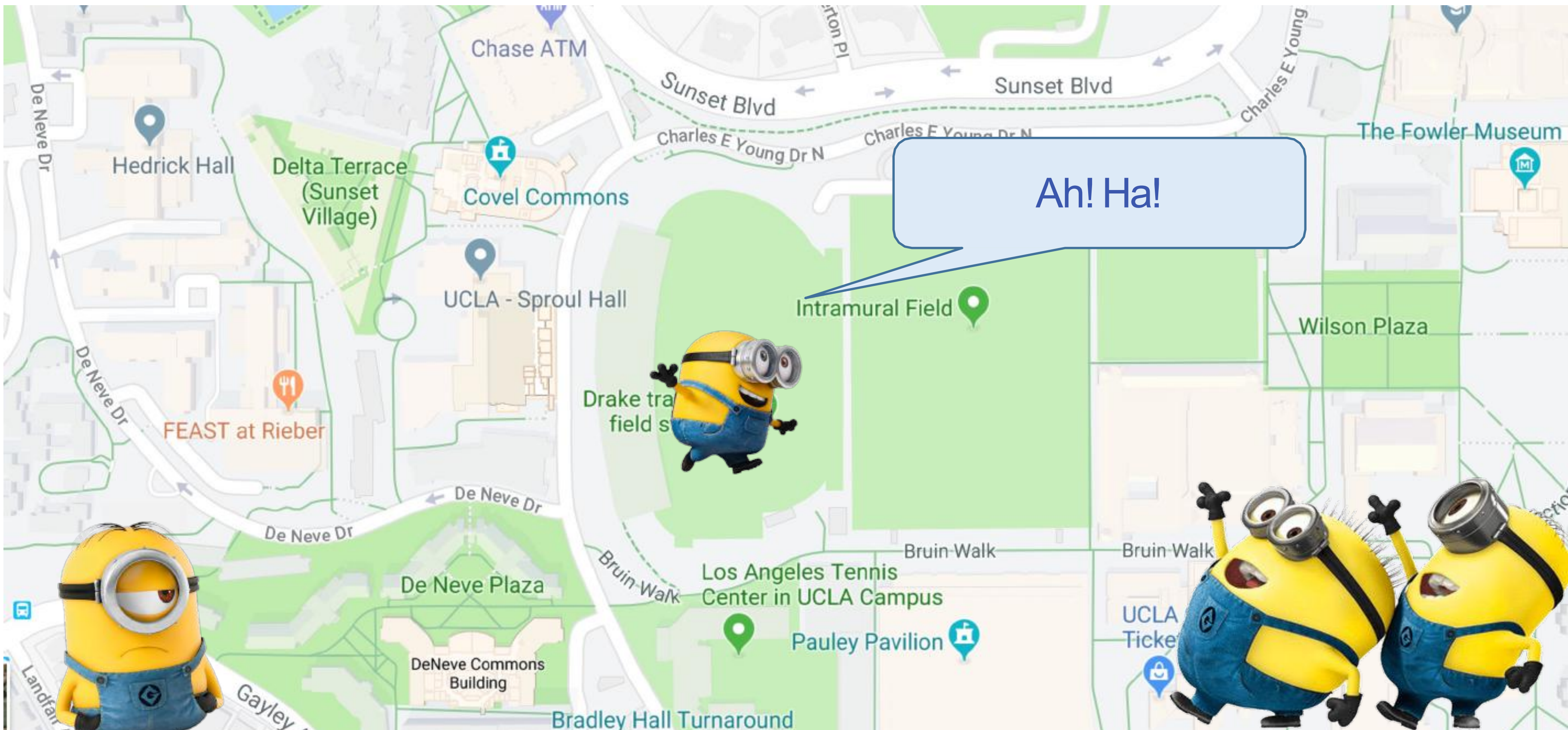
Intuition

Asking direction



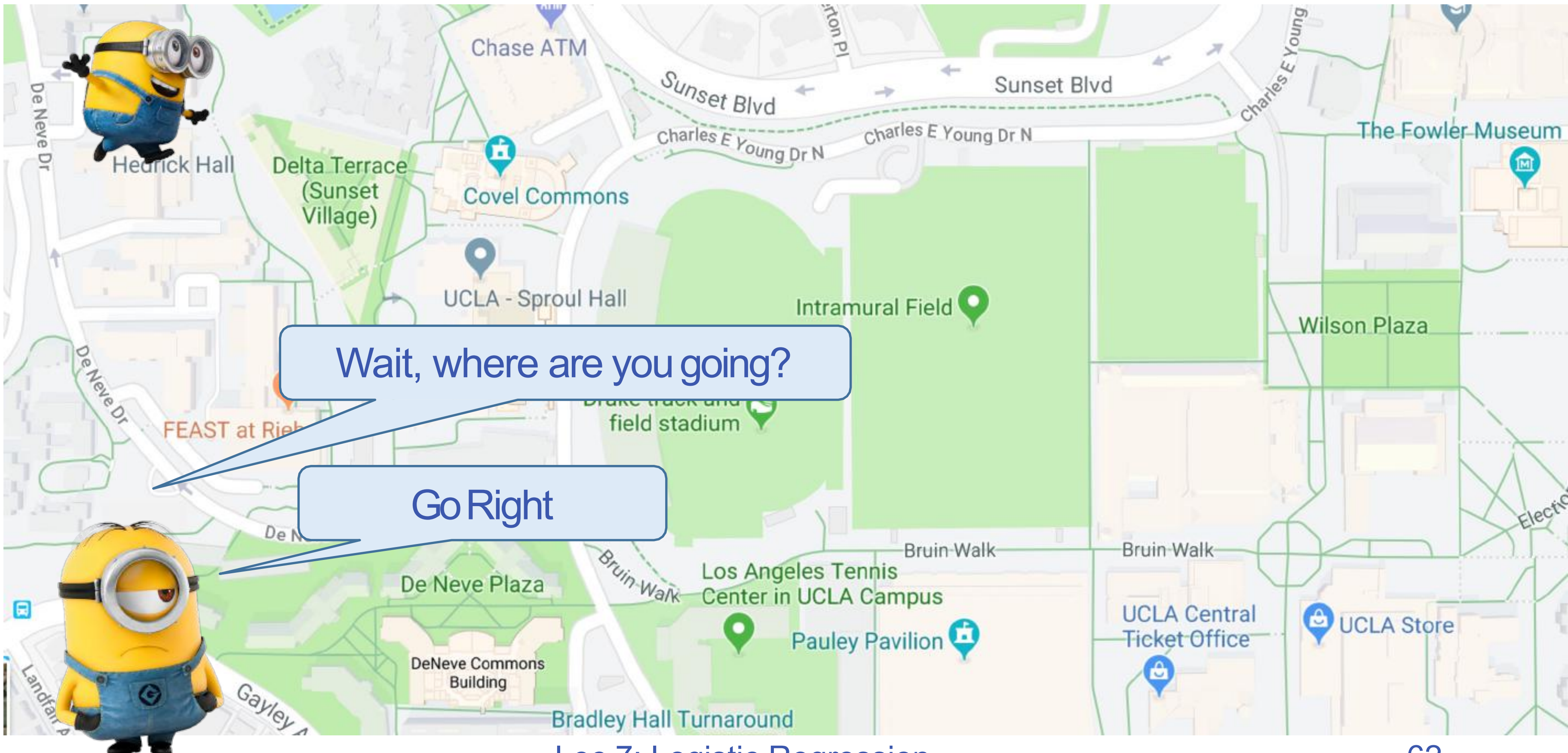
Intuition

Asking direction



Intuition

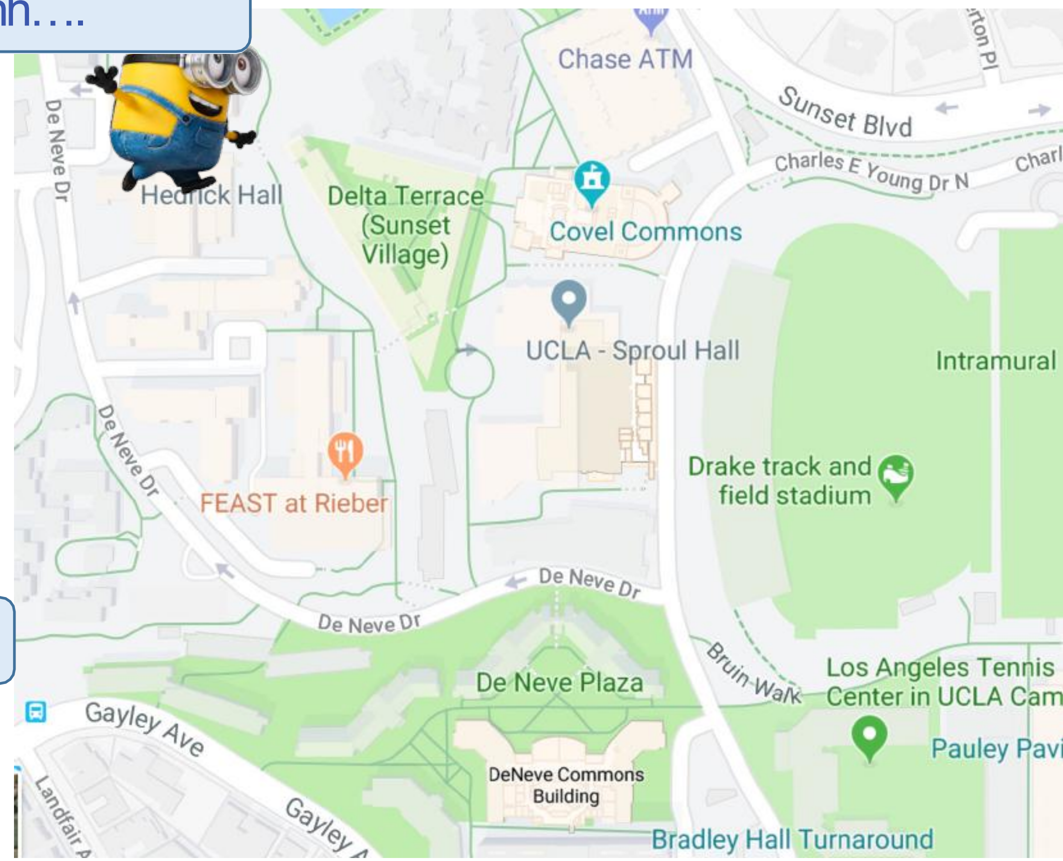
What may go wrong? Incorrect Step-size



Intuition

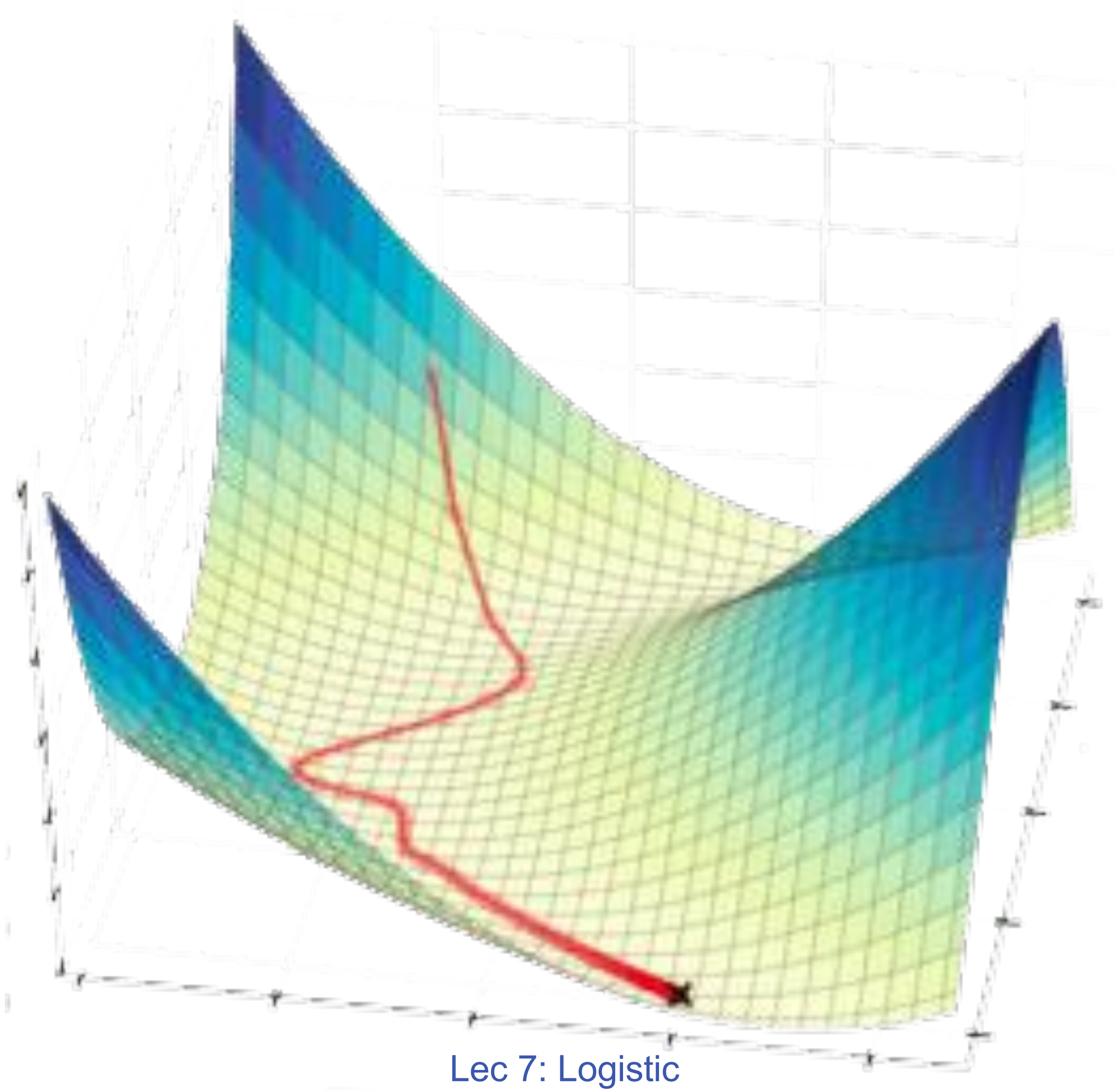
What may go wrong? Incorrect Direction

AHHhhhhhhh….



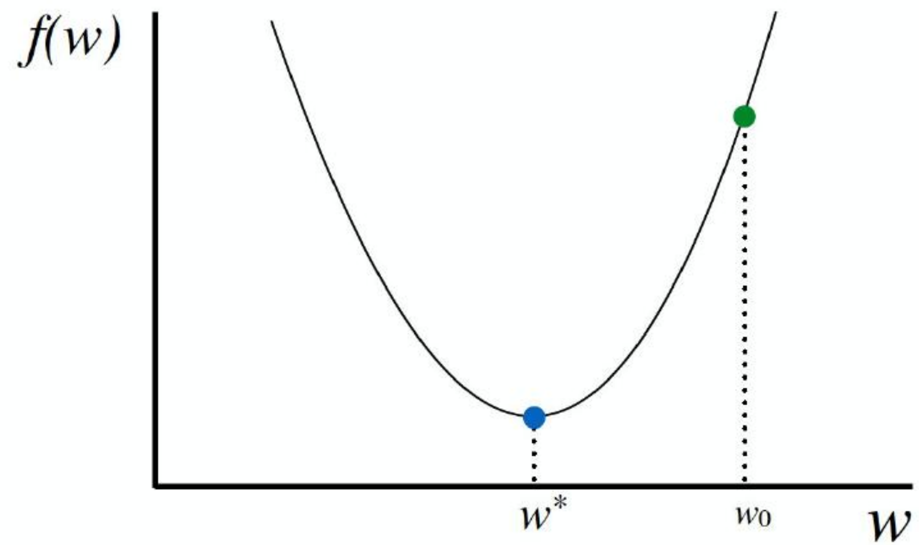
Go left, hehehe…

Gradient Descent



Gradient descent

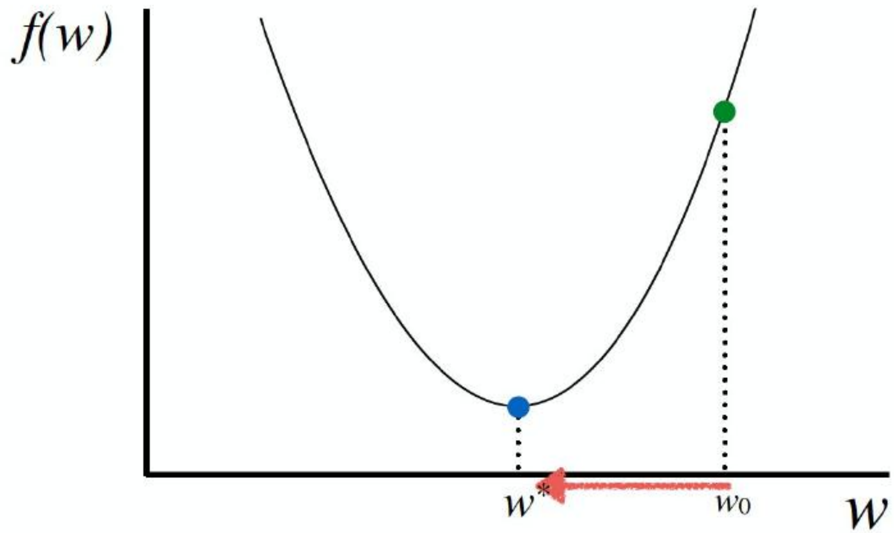
Start at a random point



Gradient descent

Start at a random point

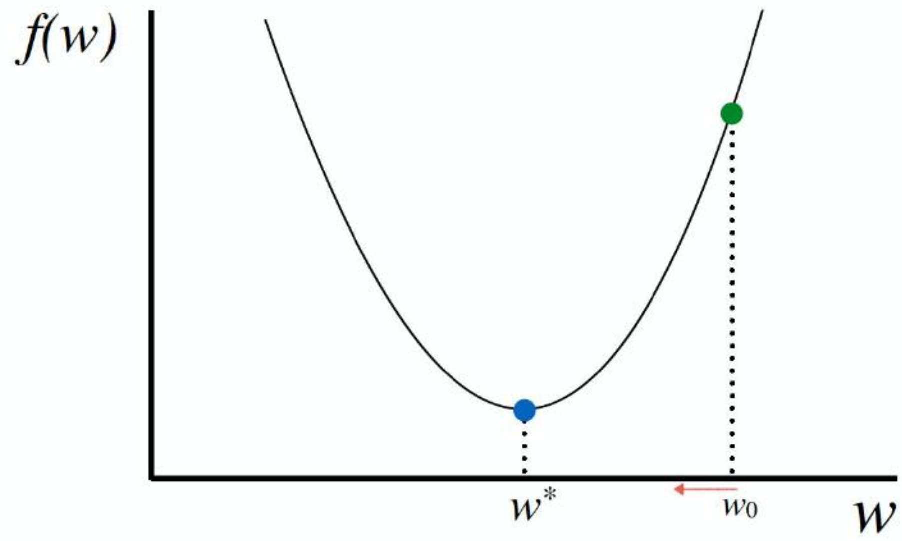
Determine a descent direction



Gradient descent

Start at a random point

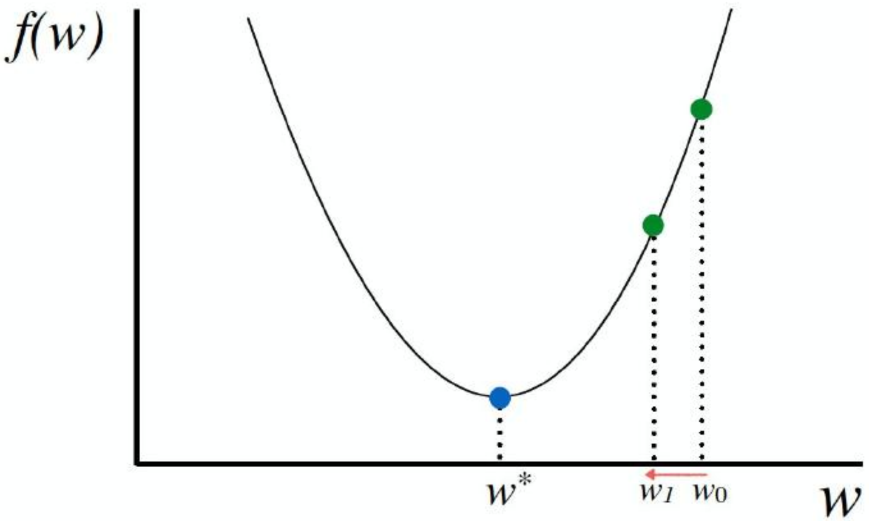
Determine a descent direction
Choose a step size



Gradient descent

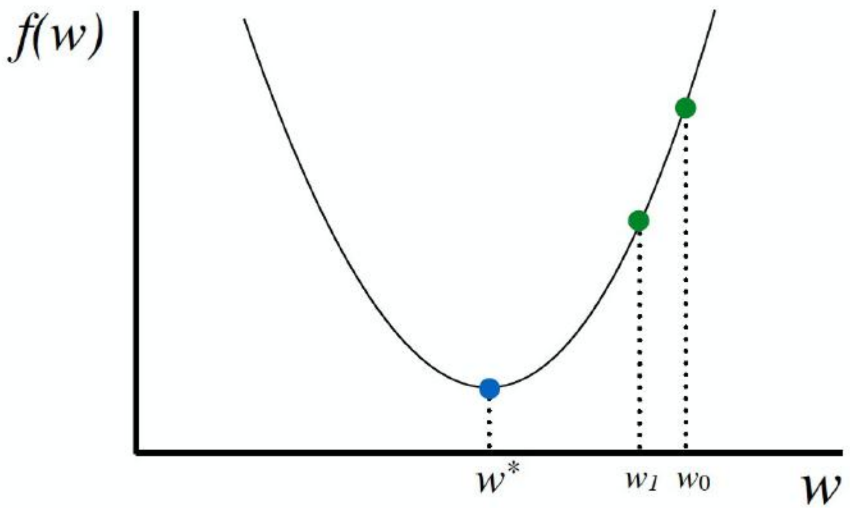
Start at a random point

Determine a descent direction
Choose a step size
Update



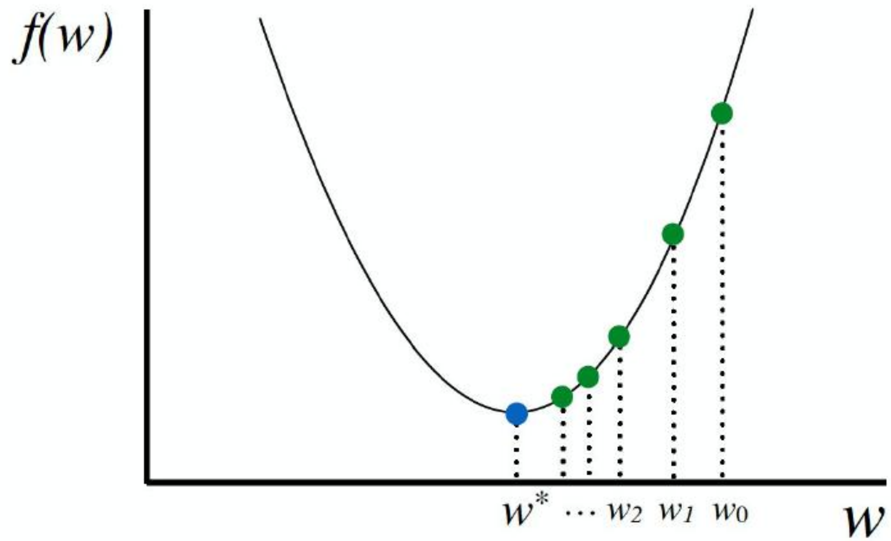
Gradient descent

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



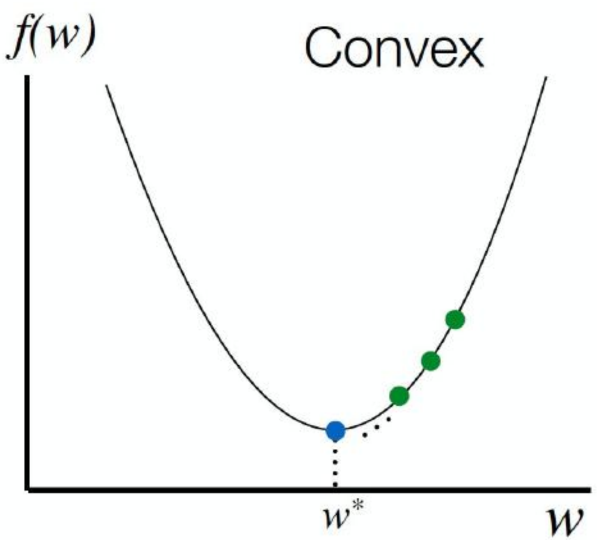
Gradient descent

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

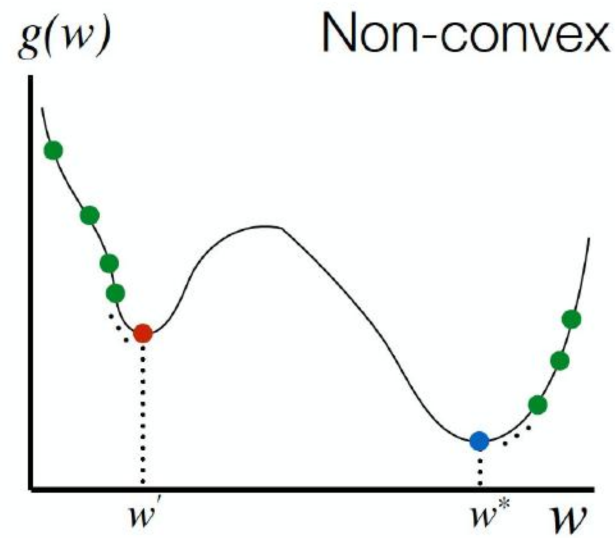


Where will we converge?

If function is convex, it converges to the global optimum (need proper choice of step-size)



Any local minimum is a global minimum



Multiple local minima may exist

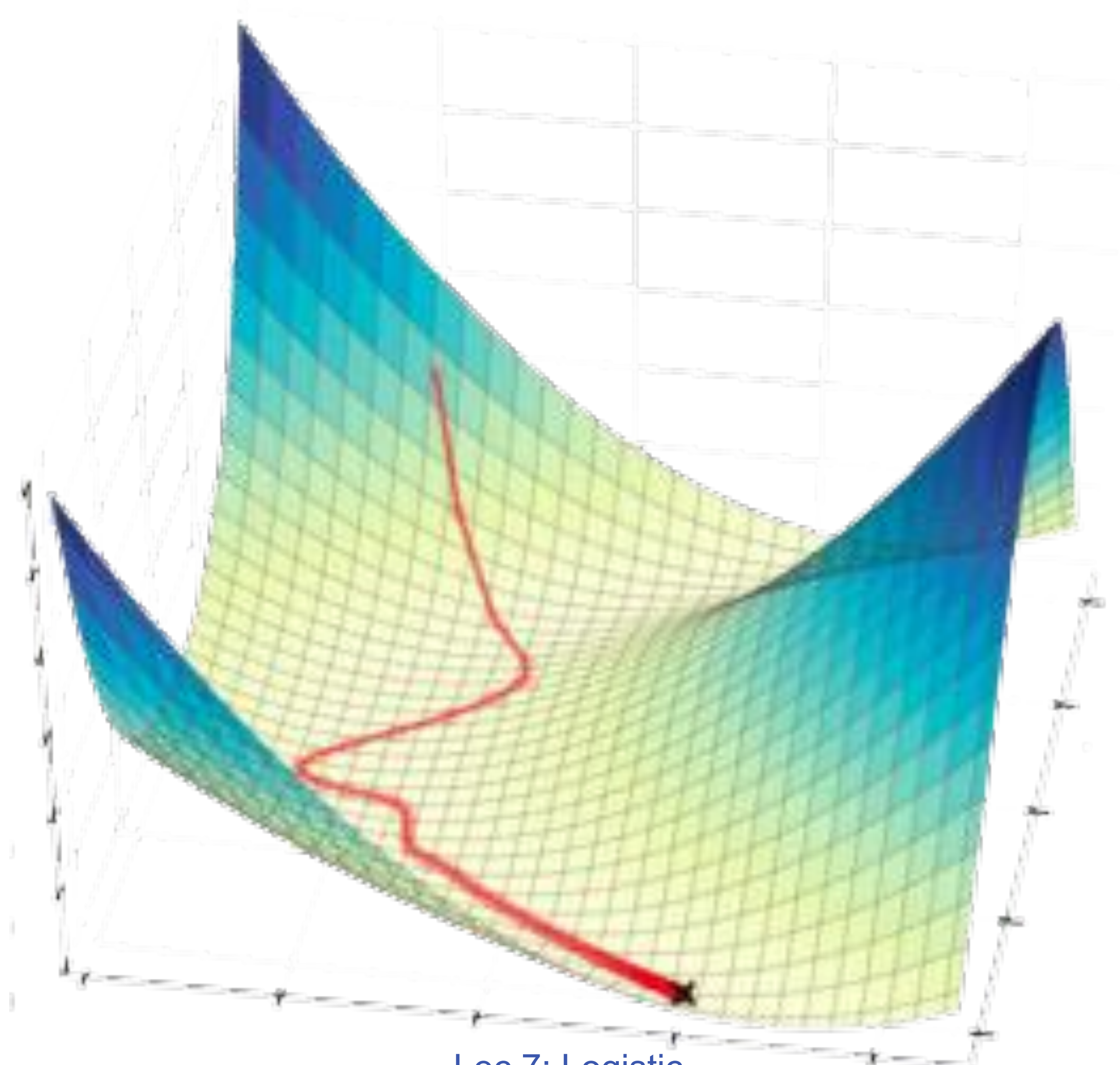
Gradient Descent

Algorithm 1 Gradient Descent (J)

1: $t \leftarrow 0$
2: Initialize $\theta^{(0)}$
3: **repeat**
4: $\quad \boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \nabla J(\boldsymbol{\theta}^{(t)})$
5: $\quad t \leftarrow t + 1$
6: **until** convergence
7: Return final value of $\boldsymbol{\theta}$

Need to compute the gradient for the negative log likelihood

Gradient Descent



Example

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

We compute the gradients

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

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

- Use the following iterative procedure for gradient descent

$$\nabla f(\theta) = \begin{bmatrix} 2(\theta_1^2 - \theta_2)\theta_1 + \theta_1 - 1 \\ -(\theta_1^2 - \theta_2) \end{bmatrix}$$

1. Initialize $\theta_1^{(0)}$ and $\theta_2^{(0)}$, and $t = 0$
2. do

Type equation here.

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

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

$$t \leftarrow t + 1$$

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

Remarks

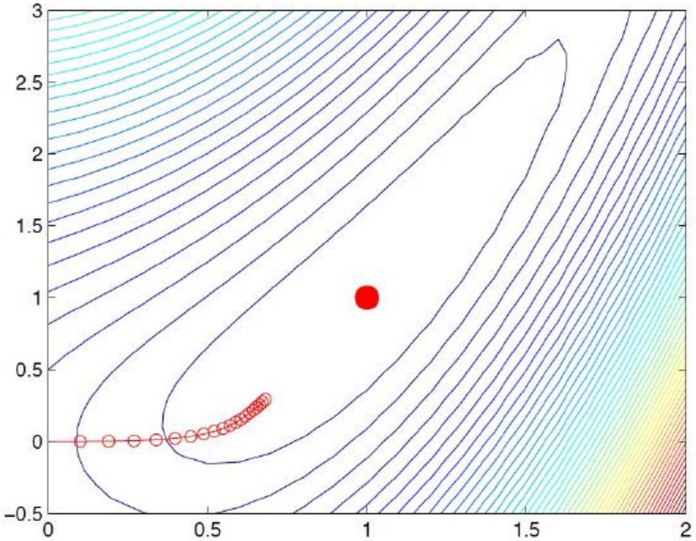
- η is often called step size or learning rate -- how far our update will go along the the direction of the negative gradient
- With a suitable choice of η, the iterative procedure converges to a stationary point where

$$\frac{\partial f}{\partial \boldsymbol{\theta}} = 0$$

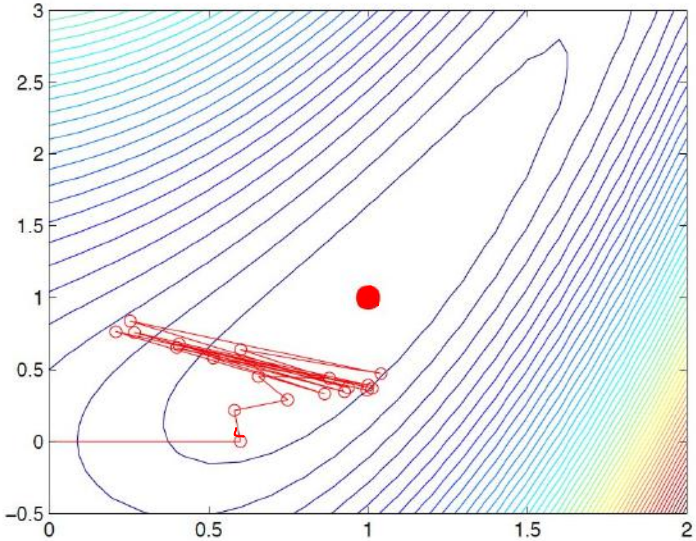
- A stationary point is only necessary for being the minimum

Choosing the right η is important

small η is too slow?

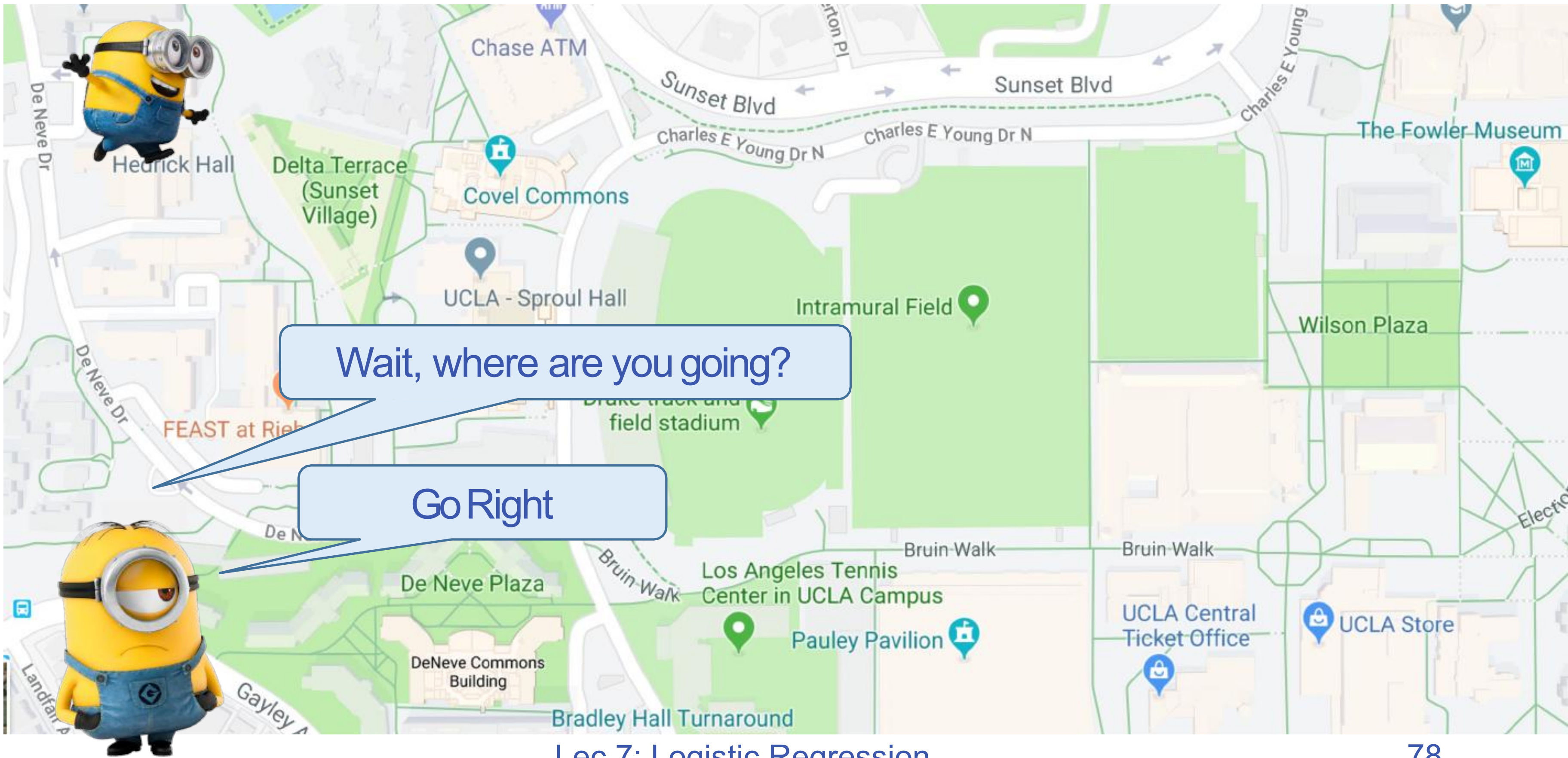


large η is too unstable?



Intuition

What may go wrong? Incorrect Step-size



Iterative optimization

Algorithm 2 Gradient Descent (J)

$t \leftarrow 0$
Initialize $\theta^{(0)}$
repeat
  $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \boxed{\eta \nabla J(\boldsymbol{\theta}^{(t)})}$
  $t \leftarrow t + 1$
until convergence
Return final value of $\boldsymbol{\theta}$

How to compute the gradient?

Need to compute the gradient for the linear regression cost function (residual sum of squares RSS)

Stochastic Gradient Descent

Incremental/Stochastic gradient descent

Repeat for each example ($\mathbf{x}_i$, y_i)

Use this example to calculate the gradient and update the model

Contrast with *batch gradient descent* which makes one update to the weight vector for every pass over the data

Stochastic gradient descent

If $f(w) = \frac{\mathbf{1}}{|\boldsymbol{D}|} \sum_i^{|D|} f_i(w)$

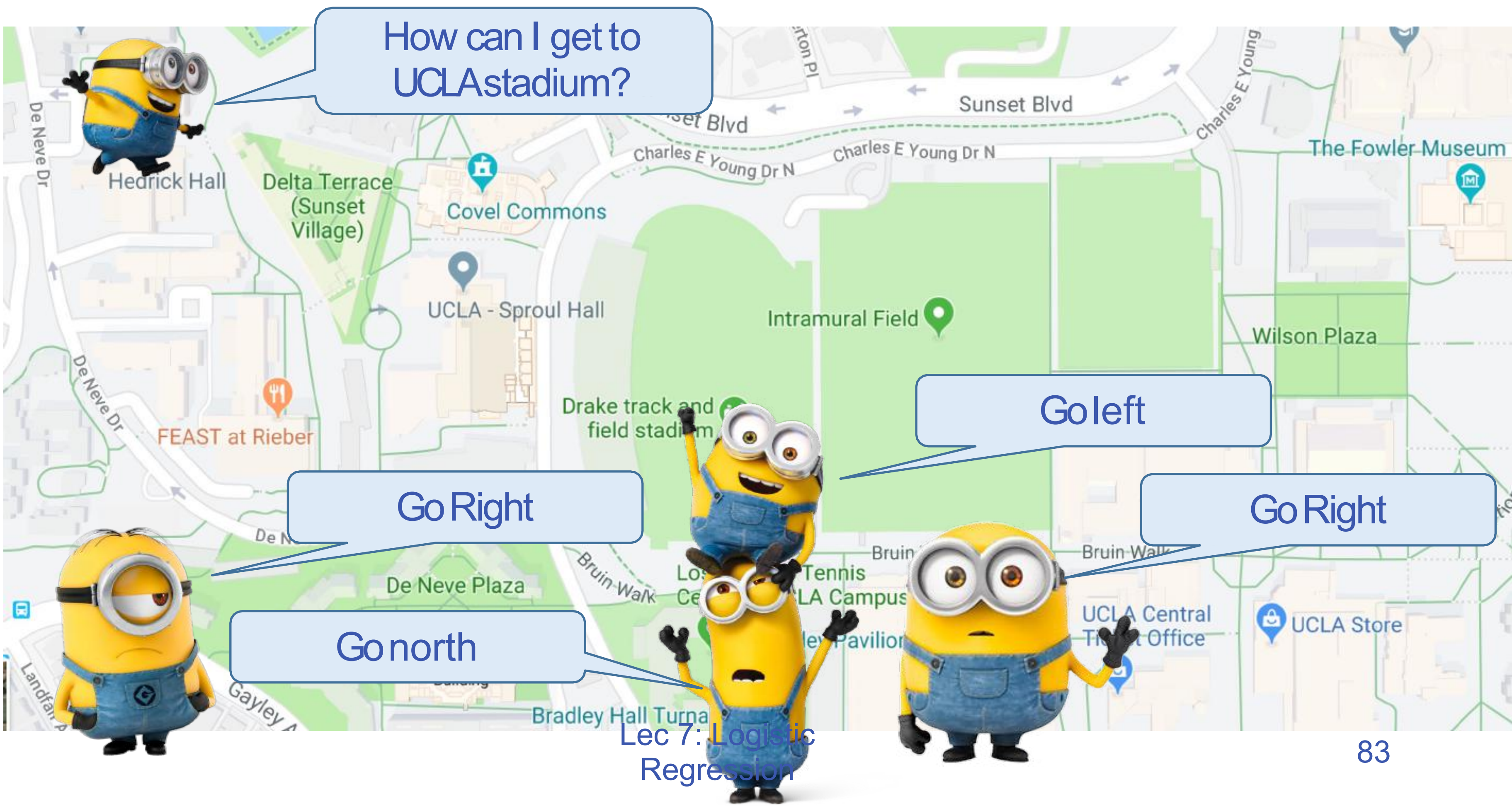
$$\nabla f(w) = \frac{1}{|D|} \sum_i \nabla f_i(w) = E_{i \sim D} \nabla f_i(w)$$

- Approximate the true gradient by a gradient at a single example at a time

> Repeat until converge:
> Randomly pick one sample (x_i, y_i)
> Update $\mathrm{w} \leftarrow \mathrm{w} - \eta \nabla \mathrm{f_i}(\mathrm{w})$

Intuition

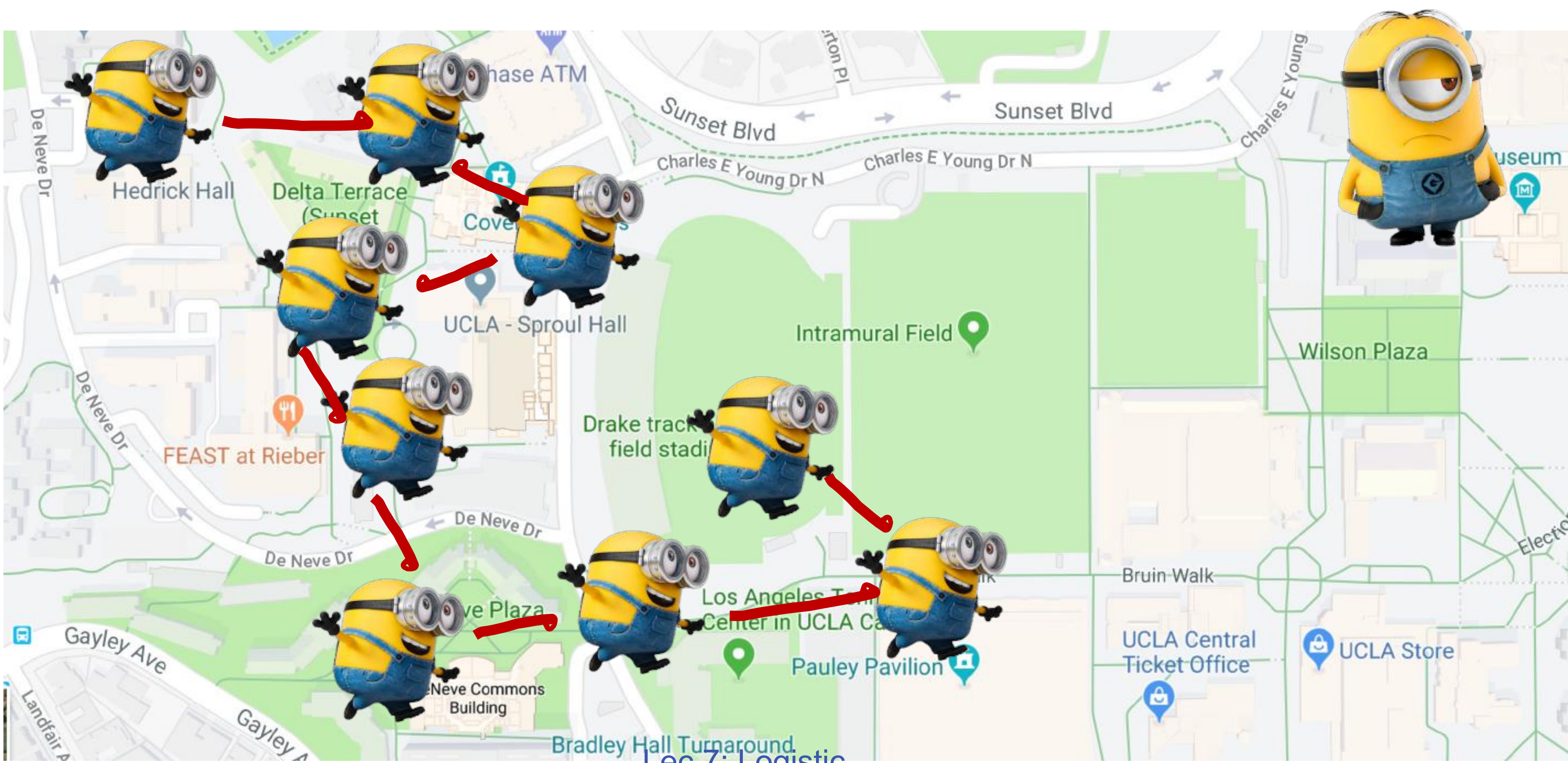
Asking direction. Gradient descent: compute gradient of all instances.



Intuition

Asking direction. Stochastic Gradient descent:

compute approximate gradient by one instance



Stochastic gradient Descent

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

1. Initialize $\boldsymbol{w} \leftarrow \boldsymbol{0} \in \mathbb{R}^n$
2. For epoch $1 \dots T$:
3.   For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
4.   Update $w \leftarrow w - \eta \nabla f(w)$
5. Return $\boldsymbol{w}$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

1. Initialize $\boldsymbol{w} \leftarrow \boldsymbol{0} \in \mathbb{R}^n$
2. For epoch $1 \ldots T$:
3. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
4. if $y(\boldsymbol{w}^\top \boldsymbol{x}) < 0$
5. $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y \boldsymbol{x}$
6. Return $\boldsymbol{w}$

Prediction: $y^{\text{test}} \leftarrow \operatorname{sgn}(\boldsymbol{w}^\top \boldsymbol{x}^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

1. Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \dots T$:
3. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
4. if $y(\boldsymbol{w}^\top \boldsymbol{x}) < \mathbf{0}$
5. $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y \boldsymbol{x}$
6. Return $\boldsymbol{w}$

Prediction: y^{test}

Perceptron effectively minimizing:

$$\sum_i \max(0,\ 1 - \ y_i(\mathbf{w}^T \mathbf{x}_i))$$

Summary

- Maximum Likelihood Estimation
- Gradient Descent
- Stochastic Gradient Descent

- We will see more examples in later lectures