Lecture 4: Logistic Regression

Instructor: Swabha Swayamdipta

USC CSCI 544 Applied NLP

Sep 5, Fall 2024

Some slides adapted from Dan Jurafsky and Chris Manning and Xuezhe Ma

Lecture Outline

- Recap
  - Smoothing
  - Basics of Supervised Machine Learning
    - Data: Preprocessing and Feature Extraction
- Quiz
- Announcements
- Basics of Supervised Machine Learning
  I. Data: Preprocessing and Feature Extraction
  II. Model:
      I. Logistic Regression
  III. Loss
  IV. Optimization Algorithm
  V. Inference

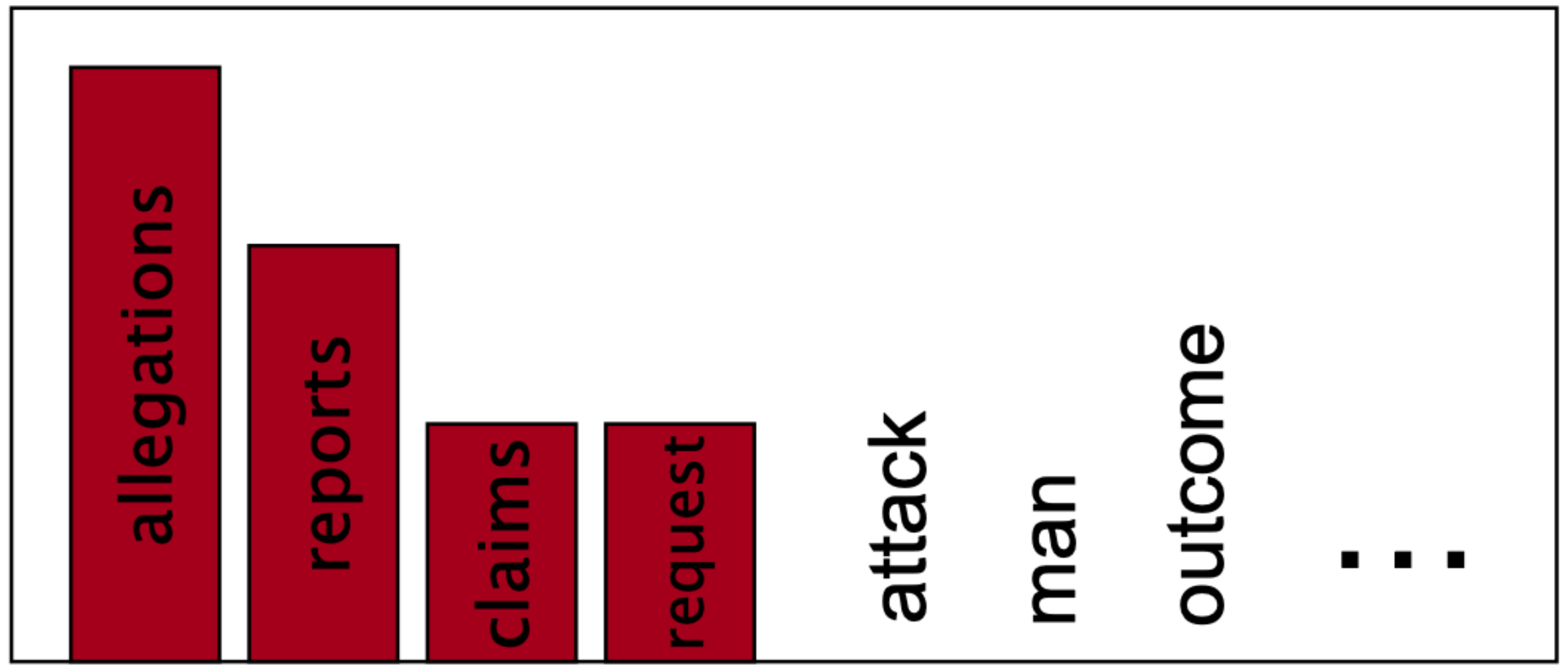
Recap: Smoothing

USC Viterbi

Smoothing ~ Massaging Probability Masses

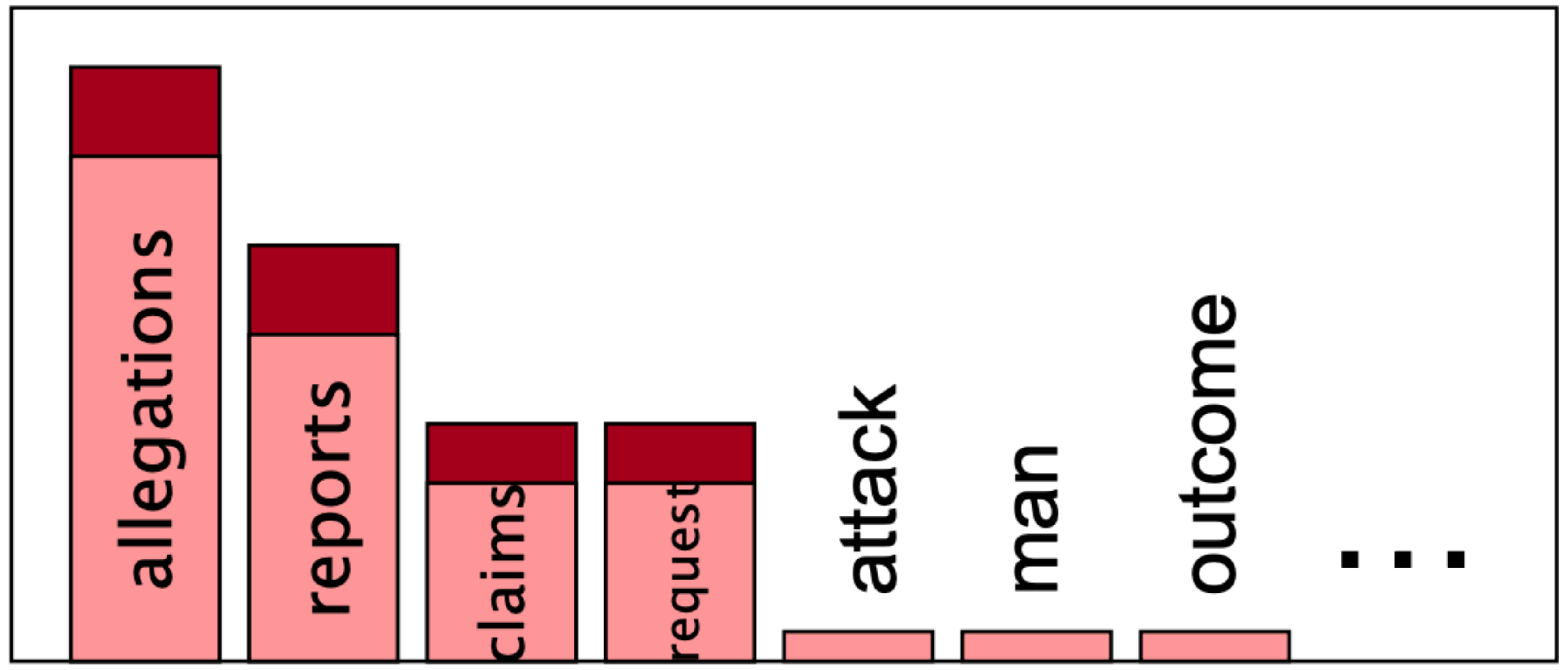
When we have sparse statistics: $Count(w \mid \text{denied the})$

3 allegations
2 reports
1 claims
1 request
7 total



Steal probability mass to generalize better: $Count(w \mid \text{denied the})$

2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



Add-One Estimation

Laplace smoothing

1. Pretend we saw each n-gram one more time than we did
2. Just add one to all the n-gram counts!
3. All the counts that used to be zero will now have a count of 1…

Add-1 estimate for Unigrams

$$P_{Add-1}(w_i) = \frac{c(w_i) + 1}{\sum_w (c(w) + 1)} = \frac{c(w_i) + 1}{V + \sum_w c(w)}$$

Add-One Estimation

Laplace smoothing

1. Pretend we saw each n-gram one more time than we did
2. Just add one to all the n-gram counts!
3. All the counts that used to be zero will now have a count of 1…

Add-1 estimate for Unigrams

$$P_{Add-1}(w_i) = \frac{c(w_i) + 1}{\sum_w (c(w) + 1)} = \frac{c(w_i) + 1}{V + \sum_w c(w)}$$

Add-1 estimate for Bigrams

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1} w_i) + 1}{c(w_{i-1}) + V}$$

Original vs Add-1 smoothed bigram counts

Original, Raw

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Reconstructed

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Big change to the counts!

Original vs Add-1 smoothed bigram counts

Original, Raw

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Reconstructed

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Big change to the counts!

Perhaps 1 is too much, add a fraction?

Add-k smoothing

k is a hyperparameter

Linear Interpolation

Simple Interpolation

$$\hat{P}(w_i | w_{i-2}w_{i-1}) = \lambda_1 P(w_i) + \lambda_2 P(w_i | w_{i-1}) + \lambda_3 P(w_i | w_{i-2}w_{i-1})$$

$$\sum_k \boxed{\lambda_k} = 1$$

Hyperparameters!

Context-Conditional Interpolation

$$\hat{P}(w_i | w_{i-2}w_{i-1}) = \lambda_3(w_{i-2}^{i-1}) P(w_i | w_{i-2}w_{i-1}) + \lambda_2(w_{i-2}^{i-1}) P(w_i | w_{i-1}) + \lambda_1(w_{i-2}^{i-1}) P(w_i)$$

Different hyperparameters for different bigrams (context conditional)!

Recap: Basics of Supervised Machine Learning

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1...N\}$
   - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, ..., x_d]$,
     - e.g. word embeddings

II. **Model**
   - A classification function that computes $\hat{y}$, the estimated class, via $p(y \mid x)$
     - e.g. logistic regression, naïve Bayes

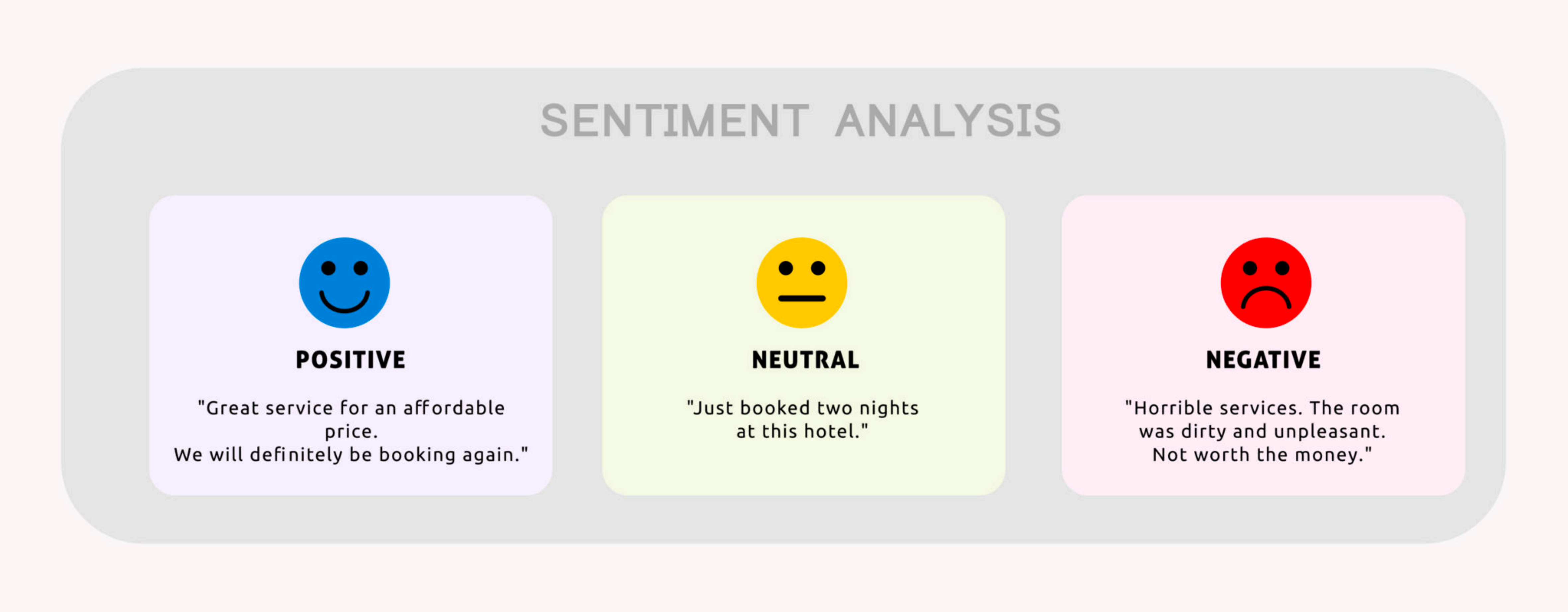
III. **Loss**
   - An objective function for learning
     - e.g. cross-entropy loss, L_{CE}

IV. **Optimization**
   - An algorithm for optimizing the objective function
     - e.g. stochastic gradient descent

V. **Inference** / Evaluation

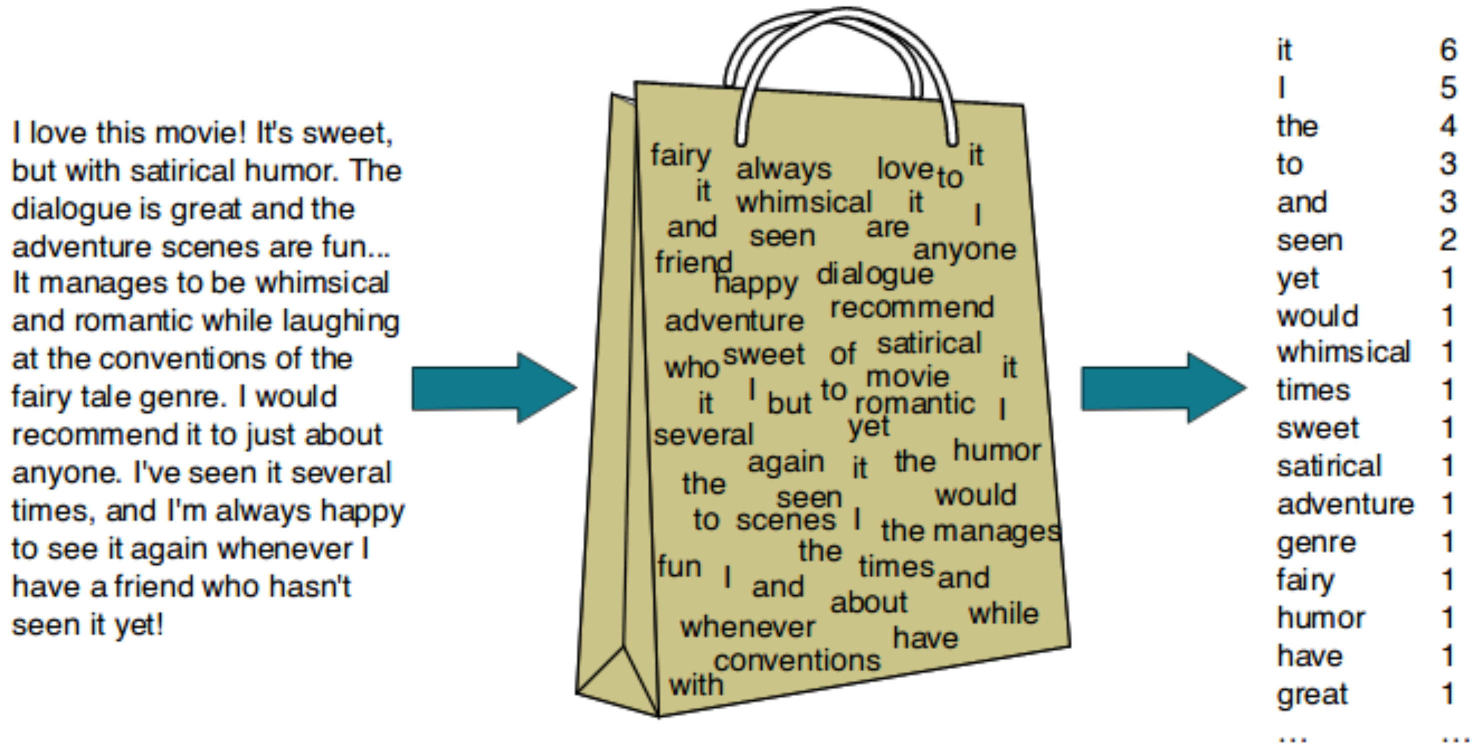
Learning Phase

Features in Classification



- Examples of feature x_i
  - x_i ="review contains 'awesome'"; $w_i = +10$
  - x_j ="review contains 'abysmal'"; $w_j = -10$
  - x_k ="review contains 'mediocre'"; $w_k = -2$
- Each x_i is associated with a weight w_i which determines how important x_i is
  - (For predicting the positive class)
- May be
  - manually configured or
  - automatically inferred, as in modern architectures

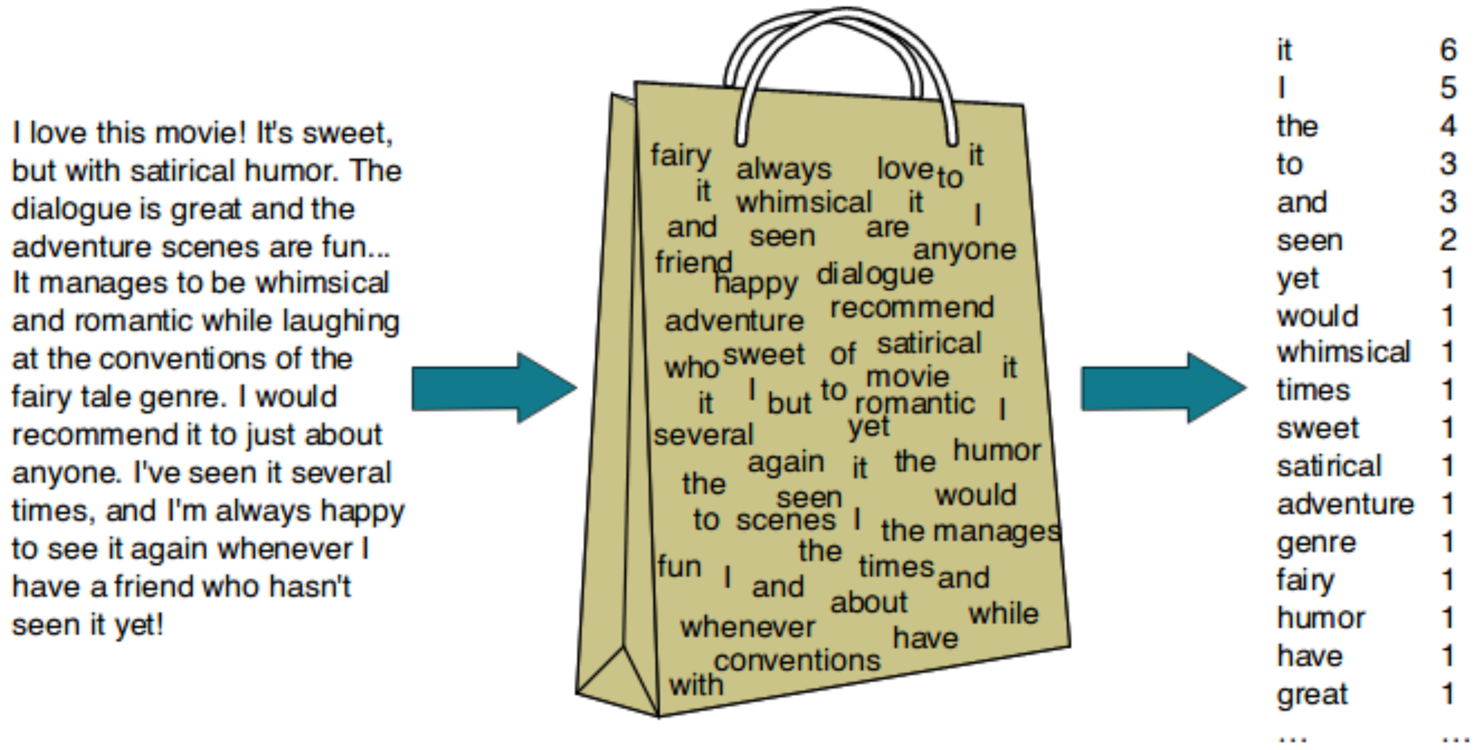
Another type of feature representation: Bag of Words



- With a word vocabulary of k words, BoW represents each doc (e.g., review) into a vector of integers
  - You may choose which k words, depending on the application
- $\mathbf{x} = [x_1, \ldots, x_k], \quad x_i \in 0,1,2,\ldots$
  - $x_i = j$ indicates that word i appears j times in the doc (e.g., review)

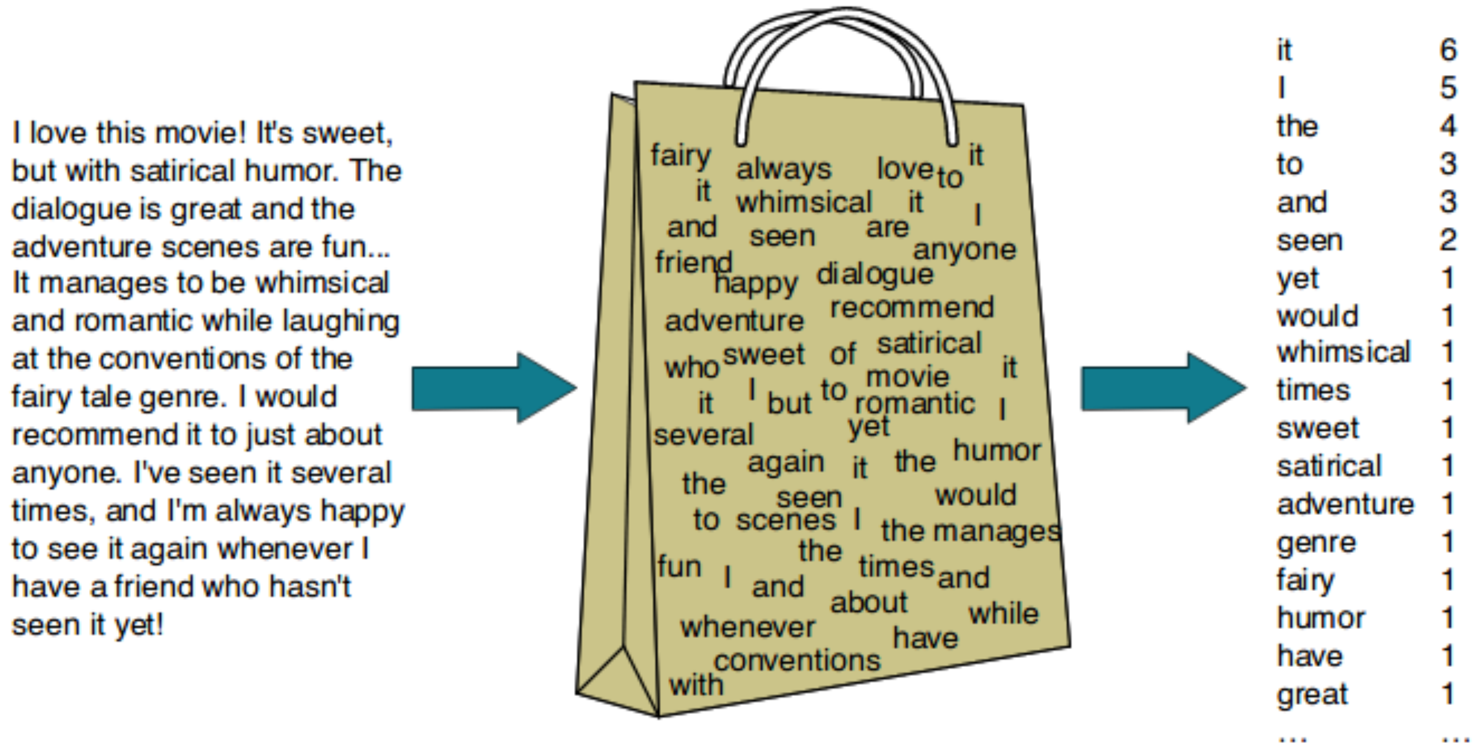
Another type of feature representation: Bag of Words

- With a word vocabulary of k words, BoW represents each doc (e.g., review) into a vector of integers
  - You may choose which k words, depending on the application
- $\mathbf{x} = [x_1, \ldots, x_k], \quad x_i \in 0,1,2,\ldots$
  - $x_i = j$ indicates that word i appears j times in the doc (e.g., review)



"I love this shirt because it is nice and warm. The fabric is also nice and the color complements my skin tone."

Another type of feature representation: Bag of Words



- With a word vocabulary of k words, BoW represents each doc (e.g., review) into a vector of integers
  - You may choose which k words, depending on the application
- $\mathbf{x} = [x_1, \ldots, x_k], \quad x_i \in 0,1,2,\ldots$
  - $x_i = j$ indicates that word i appears j times in the doc (e.g., review)

Feature Definition = [good, bad, nice, ugly, love, hate, complements, coarse, itchy]

"I love this shirt because it is nice and warm. The fabric is also nice and the color complements my skin tone."

$$\mathbf{x} = [0, \quad 0, \quad 2, \quad 0, \quad 1, \quad 0, \quad 1, \quad 0, \quad 0]$$

Bag of Words: Pros and Cons

Bag of Words: Pros and Cons

- Limitations:
  - Insensitive to language structure: all contextual information has been discarded
  - Information in word dependencies is overlooked: new york vs new book
  - The resulting vectors are highly sparse
  - Dominated by common words

Bag of Words: Pros and Cons

- Limitations:
  - Insensitive to language structure: all contextual information has been discarded
  - Information in word dependencies is overlooked: new york vs new book
  - The resulting vectors are highly sparse
  - Dominated by common words

- Pros:
  - Simple!
  - Leads to acceptable performance in quite a few settings

Lecture Outline

- Recap
  - Smoothing
  - Basics of Supervised Machine Learning
    - Data: Preprocessing and Feature Extraction
- Quiz
- Announcements
- Basics of Supervised Machine Learning
  I. Data: Preprocessing and Feature Extraction
  II. Model:
    I. Logistic Regression
  III. Loss
  IV. Optimization Algorithm
  V. Inference

Quiz 1 on Brightspace

Logistics and Announcements

Logistics and Announcements

- Sep 6: Registration Closes. Materials are going to get more complicated…

Logistics and Announcements

- Sep 6: Registration Closes. Materials are going to get more complicated…
- Next Week:
  - Tue, Sep 10: Group Formation Deadline
  - Failure to sign up: You will be randomly assigned to a group

Logistics and Announcements

- Sep 6: Registration Closes. Materials are going to get more complicated…
- Next Week:
  - Tue, Sep 10: Group Formation Deadline
  - Failure to sign up: You will be randomly assigned to a group
- Final Exam on 12/5 (in person)
  - Conflicts in exam times will NOT be supported: Consider Dropping
  - You're NOT allowed to register for overlapping classes without explicit permission from instructors

Logistics and Announcements

- Sep 6: Registration Closes. Materials are going to get more complicated…
- Next Week:
  - Tue, Sep 10: Group Formation Deadline
  - Failure to sign up: You will be randomly assigned to a group
- Final Exam on 12/5 (in person)
  - Conflicts in exam times will NOT be supported: Consider Dropping
  - You're NOT allowed to register for overlapping classes without explicit permission from instructors
- Brightspace - Subscribe to Discussions etc. if you would like to receive notifications

II. Model: Logistic Regression

Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \ldots N\}$
   - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \ldots, x_d]$,
     - e.g. word embeddings

II. **Model**
   - A classification function that computes $\hat{y}$, the estimated class, via $p(y|x)$
     - e.g. logistic regression, naïve Bayes

III. **Loss**
   - An objective function for learning
     - e.g. cross-entropy loss, L_{CE}

IV. **Optimization**
   - An algorithm for optimizing the objective function
     - e.g. stochastic gradient descent

V. **Inference** / Evaluation

Learning Phase

How to get the right y?

How to get the right y?

- For each feature x_i, introduce a weight w_i, which determines the importance of x_i

How to get the right y?

- For each feature x_i, introduce a weight w_i, which determines the importance of x_i
  - Sometimes we have a bias term, b or w_0, which is just another weight not associated to any feature

How to get the right y?

- For each feature x_i, introduce a weight w_i, which determines the importance of x_i
  - Sometimes we have a bias term, b or w_0, which is just another weight not associated to any feature
  - Together, all parameters can be termed as $\theta = [w; b]$

How to get the right y?

- For each feature x_i, introduce a weight w_i, which determines the importance of x_i
  - Sometimes we have a bias term, b or w_0, which is just another weight not associated to any feature
  - Together, all parameters can be termed as $\theta = [w; b]$
- We consider the weighted sum of all features and the bias

How to get the right y?

- For each feature x_i, introduce a weight w_i, which determines the importance of x_i
  - Sometimes we have a bias term, b or w_0, which is just another weight not associated to any feature
  - Together, all parameters can be termed as $\theta = [w; b]$
- We consider the weighted sum of all features and the bias

$$z = \left(\sum_d w_d x_d + b \right)$$

How to get the right y?

- For each feature x_i, introduce a weight w_i, which determines the importance of x_i
  - Sometimes we have a bias term, b or w_0, which is just another weight not associated to any feature
  - Together, all parameters can be termed as $\theta = [w; b]$
- We consider the weighted sum of all features and the bias

$$z = \left(\sum_d w_d x_d + b \right)$$
$$= \mathbf{w} \cdot \mathbf{x} + b$$

How to get the right y?

- For each feature x_i, introduce a weight w_i, which determines the importance of x_i
  - Sometimes we have a bias term, b or w_0, which is just another weight not associated to any feature
  - Together, all parameters can be termed as $\theta = [w; b]$
- We consider the weighted sum of all features and the bias

$$z = \left(\sum_d w_d x_d + b \right)$$

$$= \mathbf{w} \cdot \mathbf{x} + b$$

If high, $\hat{y} = 1$

How to get the right y?

- For each feature x_i, introduce a weight w_i, which determines the importance of x_i
  - Sometimes we have a bias term, b or w_0, which is just another weight not associated to any feature
  - Together, all parameters can be termed as $\theta = [w; b]$
- We consider the weighted sum of all features and the bias

$$z = \left(\sum_d w_d x_d + b \right)$$

$$= \mathbf{w} \cdot \mathbf{x} + b$$

If high, $\hat{y} = 1$

If low, $\hat{y} = 0$

How to get the right y?

- For each feature x_i, introduce a weight w_i, which determines the importance of x_i
  - Sometimes we have a bias term, b or w_0, which is just another weight not associated to any feature
  - Together, all parameters can be termed as $\theta = [w; b]$
- We consider the weighted sum of all features and the bias

$$z = \left(\sum_d w_d x_d + b \right)$$

$$= \mathbf{w} \cdot \mathbf{x} + b$$

If high, $\hat{y} = 1$

If low, $\hat{y} = 0$

But how to determine the threshold?

How to get the right y?

- For each feature x_i, introduce a weight w_i, which determines the importance of x_i
  - Sometimes we have a bias term, b or w_0, which is just another weight not associated to any feature
  - Together, all parameters can be termed as $\theta = [w; b]$
- We consider the weighted sum of all features and the bias

$$z = \left(\sum_d w_d x_d + b \right)$$
$$= \mathbf{w} \cdot \mathbf{x} + b$$

If high, $\hat{y} = 1$

If low, $\hat{y} = 0$

But how to determine the threshold?

We need probabilistic models!

$$P(y = 1 \mid \mathbf{x}; \theta)$$

$$P(y = 0 \mid \mathbf{x}; \theta)$$

$$z = \mathbf{w} \cdot \mathbf{x} + b \qquad z \in \mathbb{R}$$

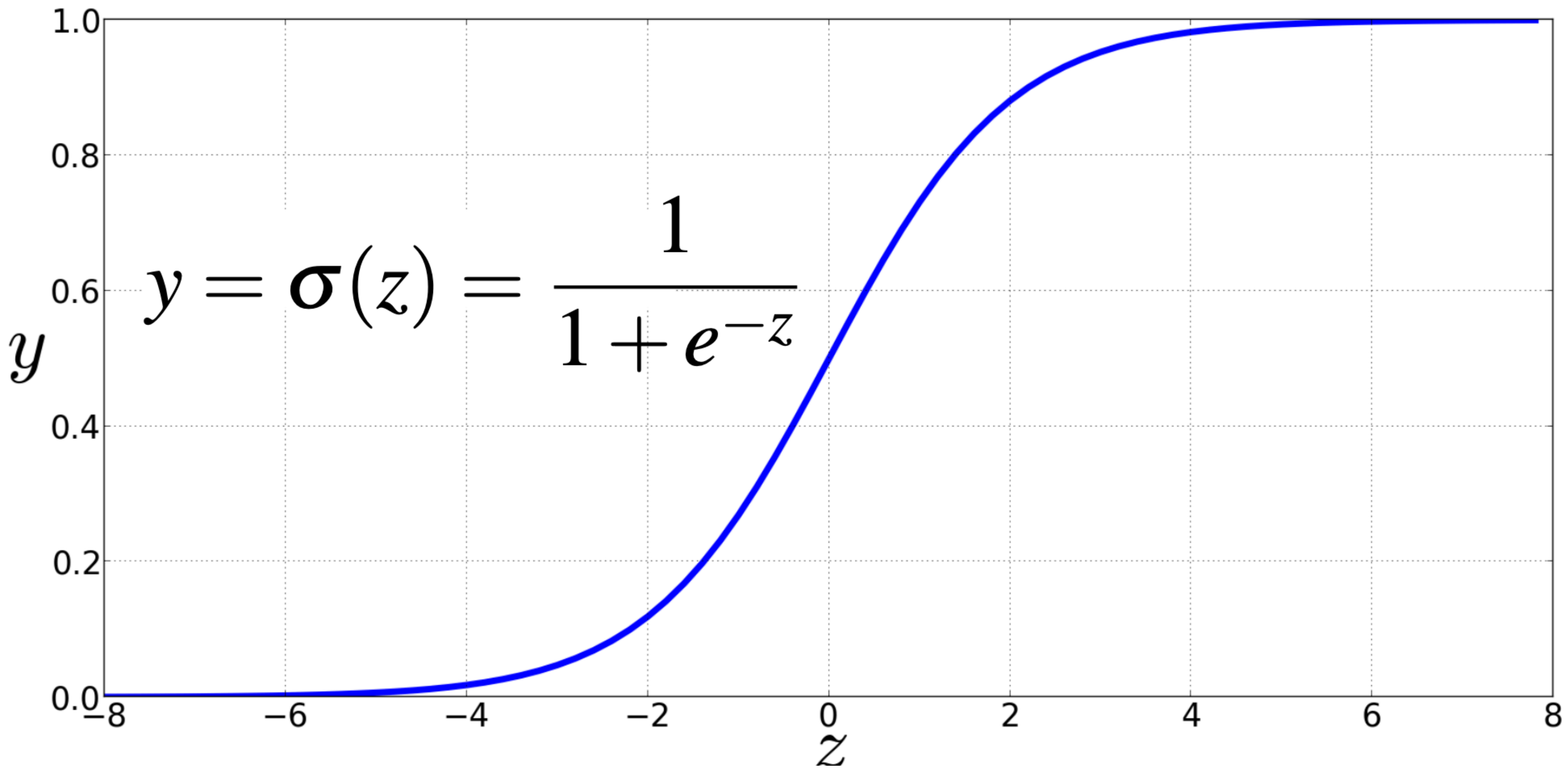
Solution: Squish it into the 0-1 range

$$z = \mathbf{w} \cdot \mathbf{x} + b \qquad z \in \mathbb{R}$$

Solution: Squish it into the 0-1 range

$$z = \mathbf{w} \cdot \mathbf{x} + b \qquad z \in \mathbb{R}$$

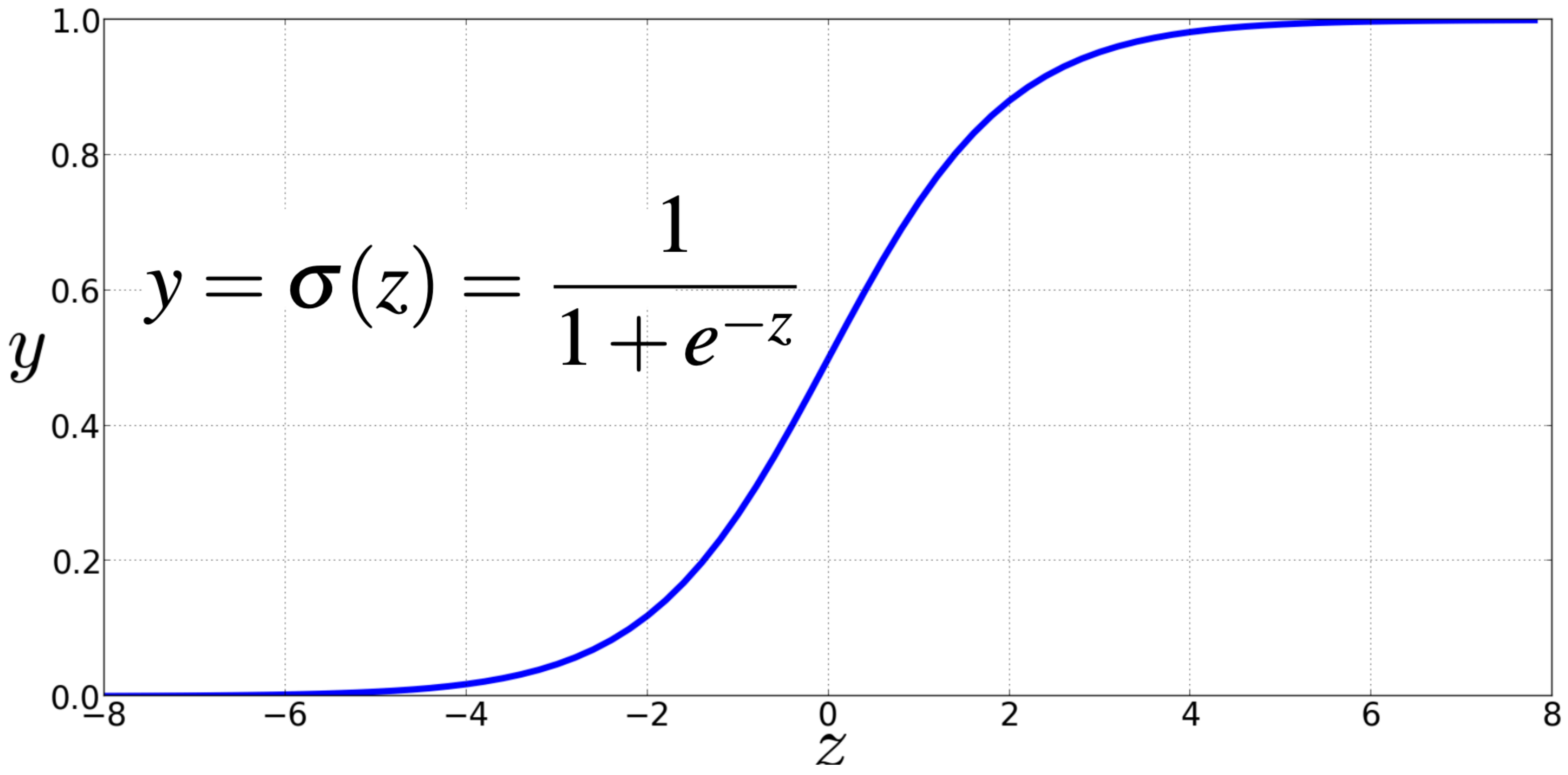
- Sigmoid Function, $\sigma(\cdot)$



Solution: Squish it into the 0-1 range

$$z = \mathbf{w} \cdot \mathbf{x} + b \qquad z \in \mathbb{R}$$

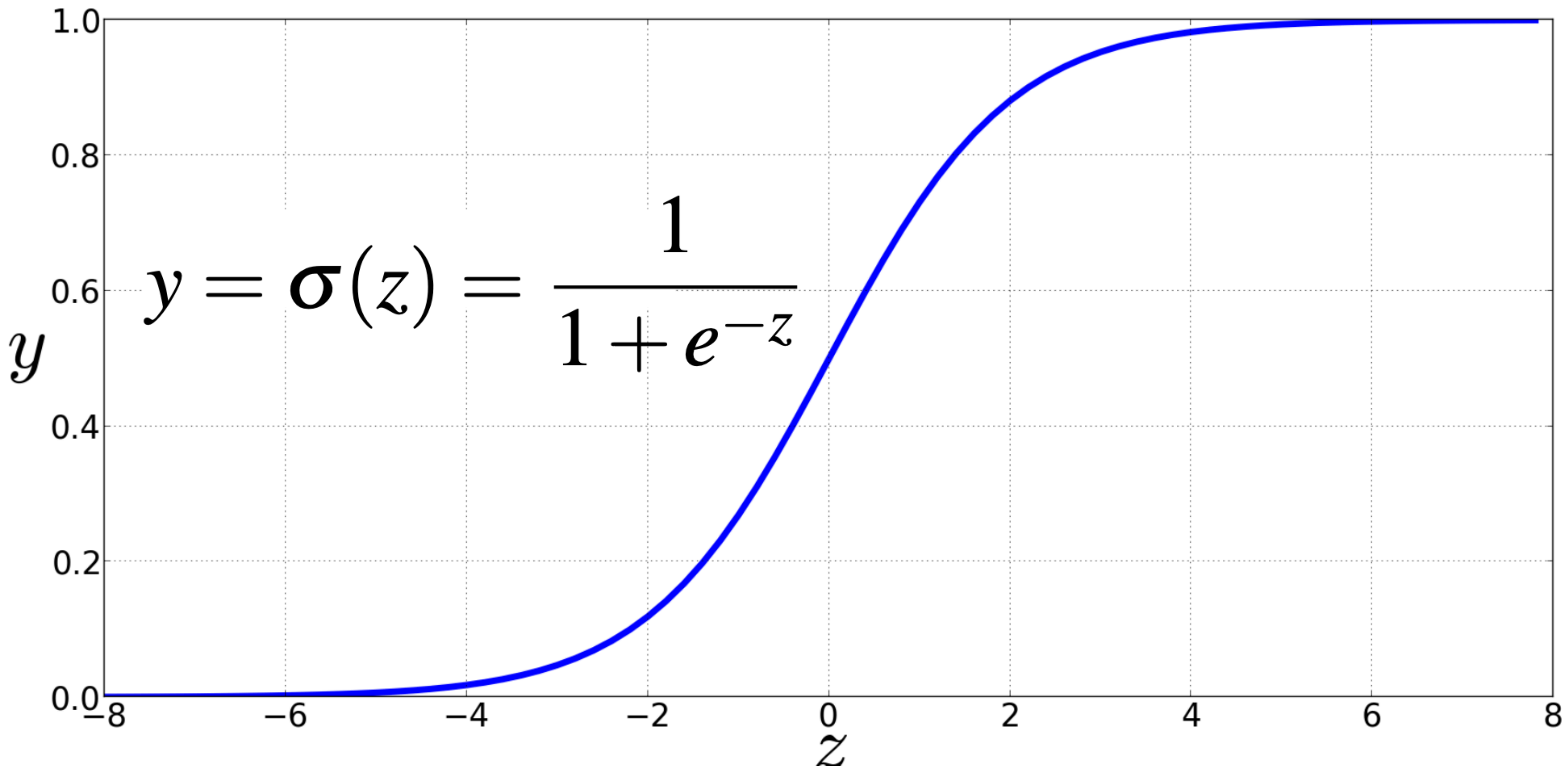
- Sigmoid Function, $\sigma(\cdot)$
  - Non-linear!



Solution: Squish it into the 0-1 range

$$z = \mathbf{w} \cdot \mathbf{x} + b \qquad z \in \mathbb{R}$$

- Sigmoid Function, $\sigma(\cdot)$
  - Non-linear!
- Compute z and then pass it through the sigmoid function

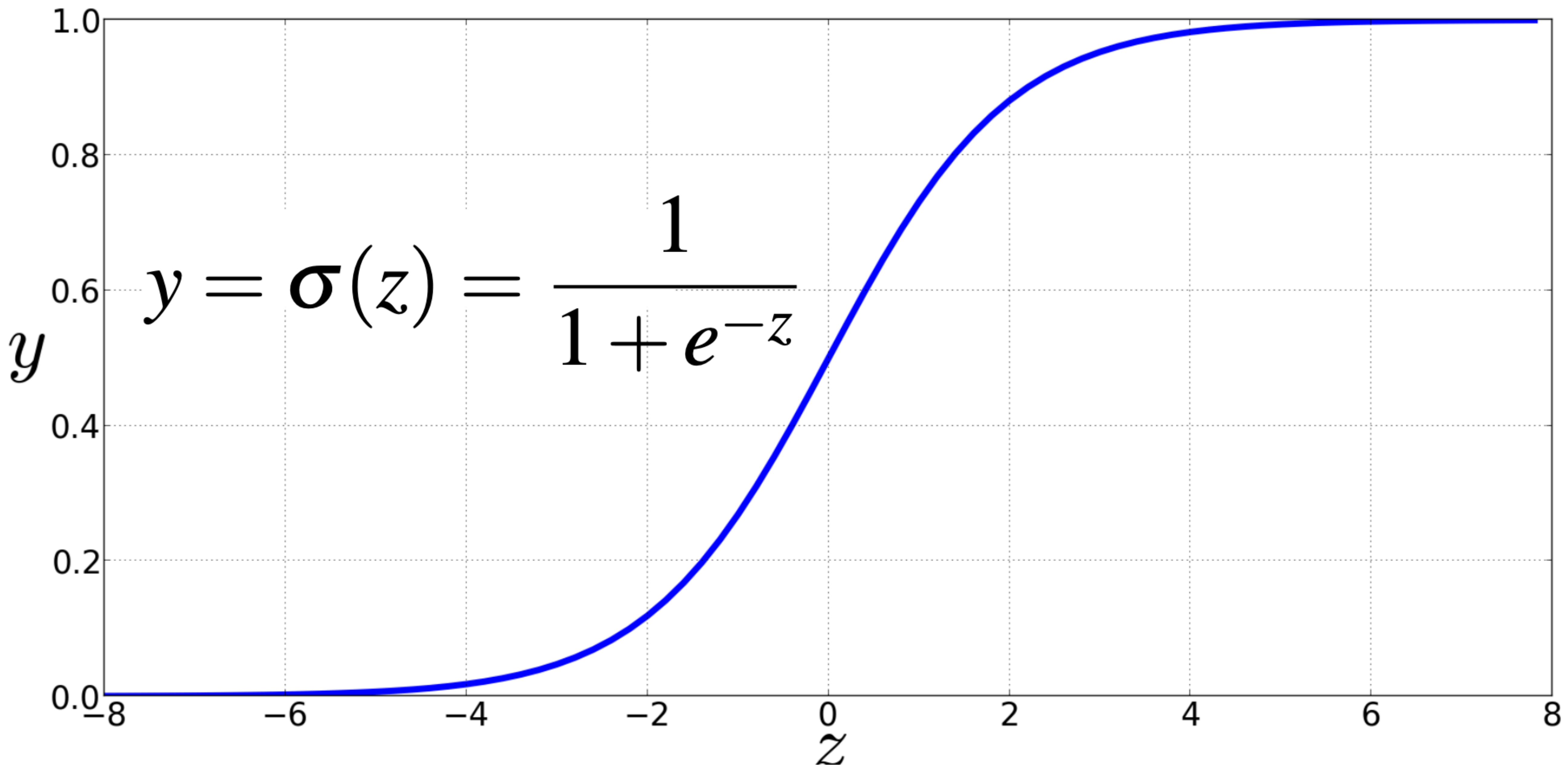


USC Viterbi

Solution: Squish it into the 0-1 range

$$z = \mathbf{w} \cdot \mathbf{x} + b \qquad z \in \mathbb{R}$$

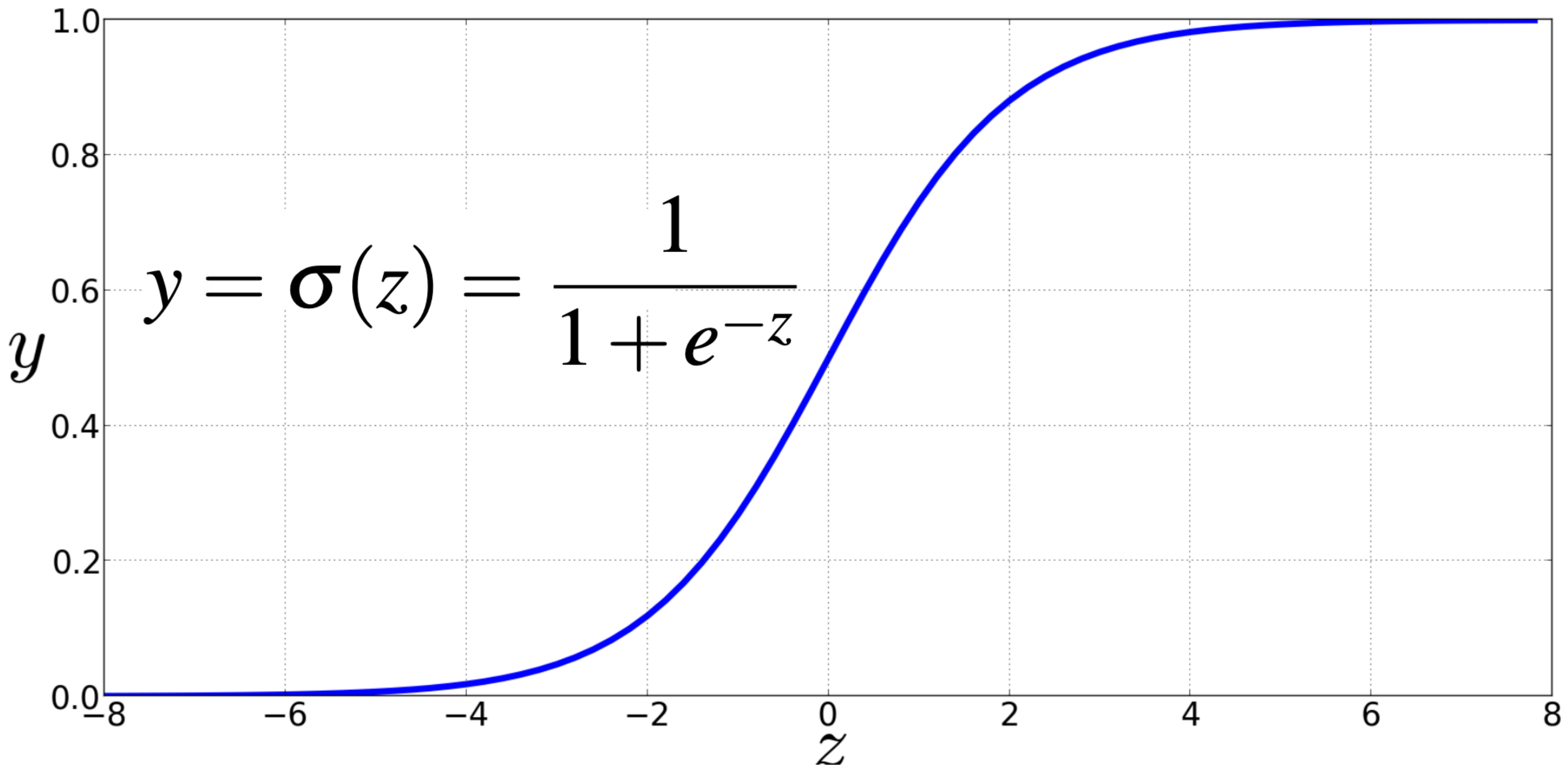
- Sigmoid Function, $\sigma(\cdot)$
  - Non-linear!
- Compute z and then pass it through the sigmoid function
- Treat it as a probability!



Solution: Squish it into the 0-1 range

$$z = \mathbf{w} \cdot \mathbf{x} + b \qquad z \in \mathbb{R}$$

- Sigmoid Function, $\sigma(\cdot)$
  - Non-linear!
- Compute z and then pass it through the sigmoid function
- Treat it as a probability!
- Also, a differentiable function, which makes it a good candidate for optimization (more on this later!)



Sigmoids and Probabilities

Sigmoids and Probabilities

$$P(y = 1 \,|\, \mathbf{x}; \theta) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

$$= \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))}$$

Sigmoids and Probabilities

$$P(y = 1 \mid \mathbf{x}; \theta) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

$$= \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))}$$

$$P(y = 0 \mid \mathbf{x}; \theta) = 1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

Sigmoids and Probabilities

$$P(y = 1 \mid \mathbf{x}; \theta) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$
$$= \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))}$$

$$P(y = 0 \mid \mathbf{x}; \theta) = 1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$
$$= 1 - \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))}$$

Sigmoids and Probabilities

$$P(y = 1 \mid \mathbf{x}; \theta) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$
$$= \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))}$$

$$P(y = 0 \mid \mathbf{x}; \theta) = 1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$
$$= 1 - \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))}$$
$$= \frac{\exp(-(\mathbf{w} \cdot \mathbf{x} + b))}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))}$$

Sigmoids and Probabilities

$$P(y = 1 \mid \mathbf{x}; \theta) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$
$$= \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))}$$

$$P(y = 0 \mid \mathbf{x}; \theta) = 1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$
$$= 1 - \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))}$$
$$= \frac{\exp(-(\mathbf{w} \cdot \mathbf{x} + b))}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))}$$
$$= \frac{1}{1 + \exp(\mathbf{w} \cdot \mathbf{x} + b)}$$

Sigmoids and Probabilities

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

$$\begin{aligned}P(y=0\,|\,\mathbf{x};\theta) &= 1-\sigma(\mathbf{w}\cdot\mathbf{x}+b)\\ &= 1-\frac{1}{1+\exp(-(\mathbf{w}\cdot\mathbf{x}+b))}\\ &= \frac{\exp(-(\mathbf{w}\cdot\mathbf{x}+b))}{1+\exp(-(\mathbf{w}\cdot\mathbf{x}+b))}\\ &= \frac{1}{1+\exp(\mathbf{w}\cdot\mathbf{x}+b)}\\ &= \sigma(-(\mathbf{w}\cdot\mathbf{x}+b))\end{aligned}$$

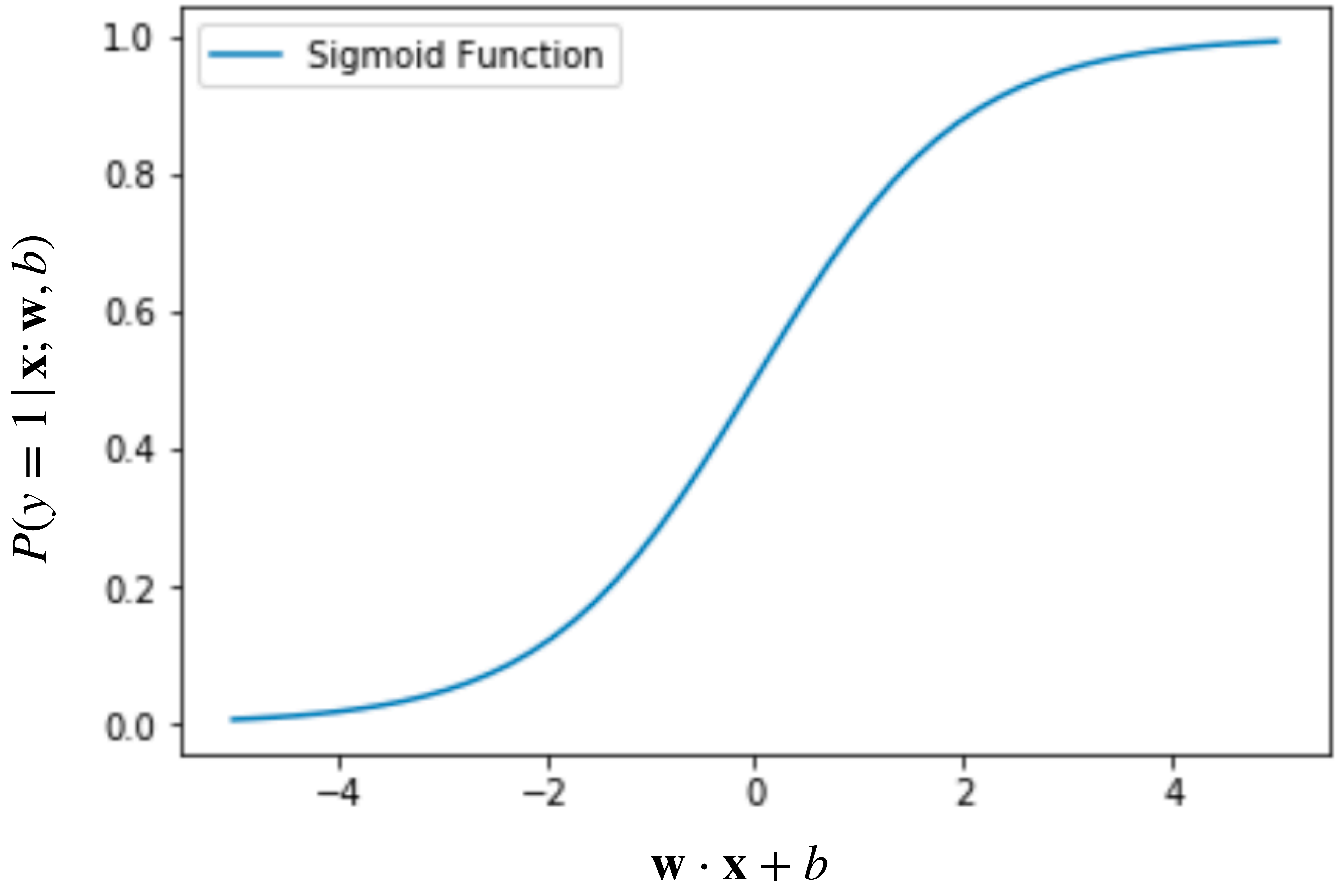
USC Viterbi

Classification Decision

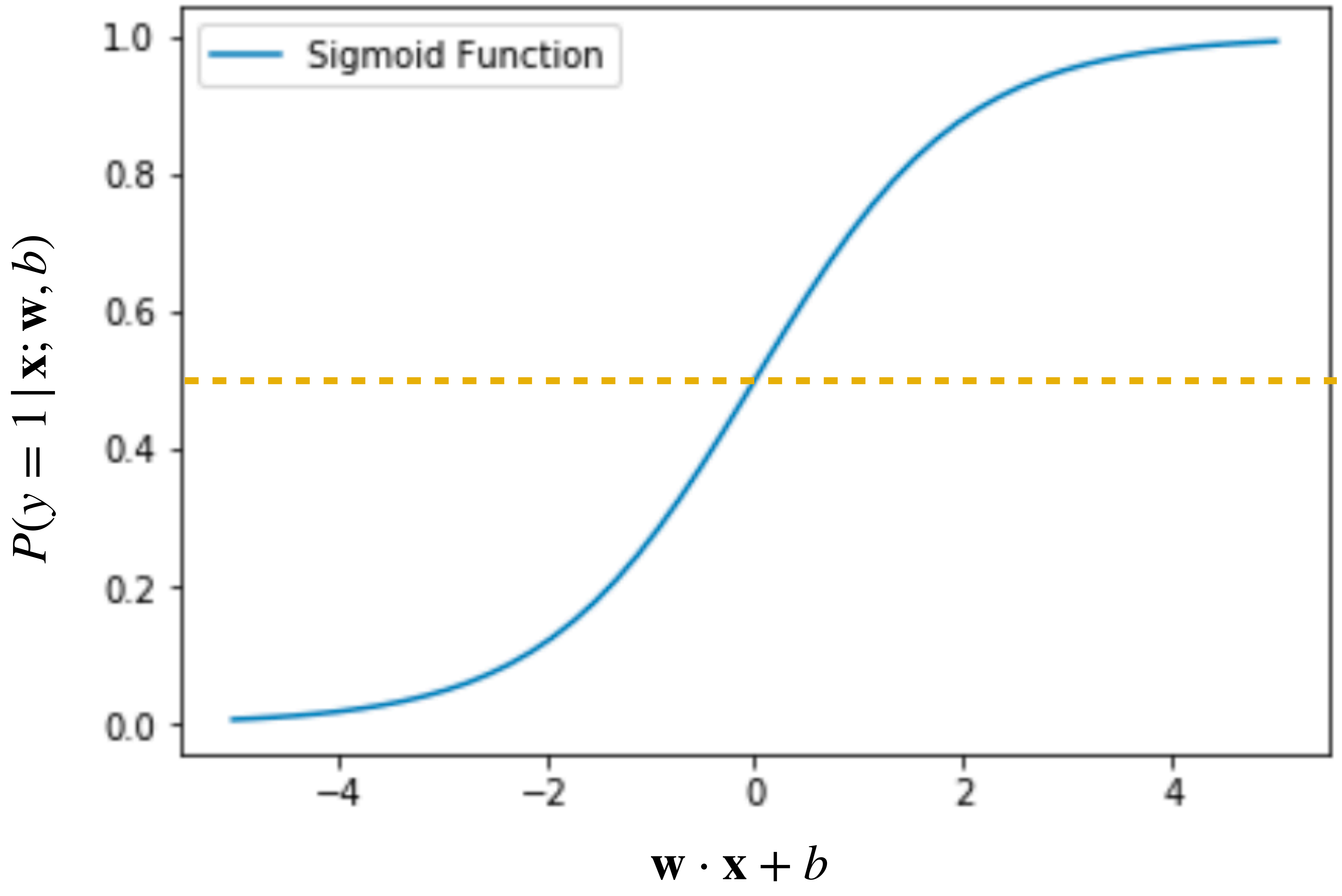
$P(y = 1 \mid \mathbf{x}; \mathbf{w}, b)$

$\mathbf{w} \cdot \mathbf{x} + b$

Classification Decision

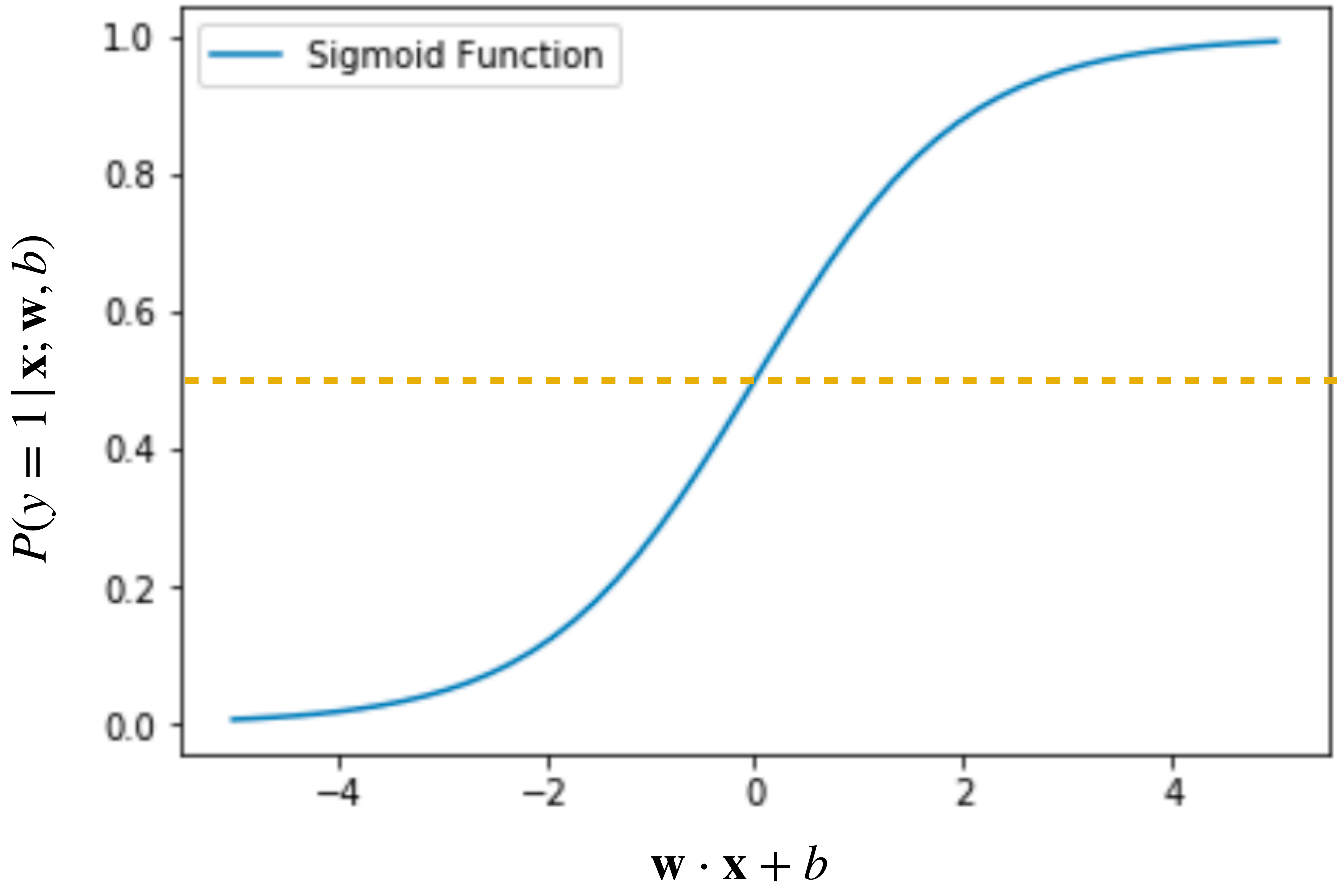


Classification Decision



Classification Decision

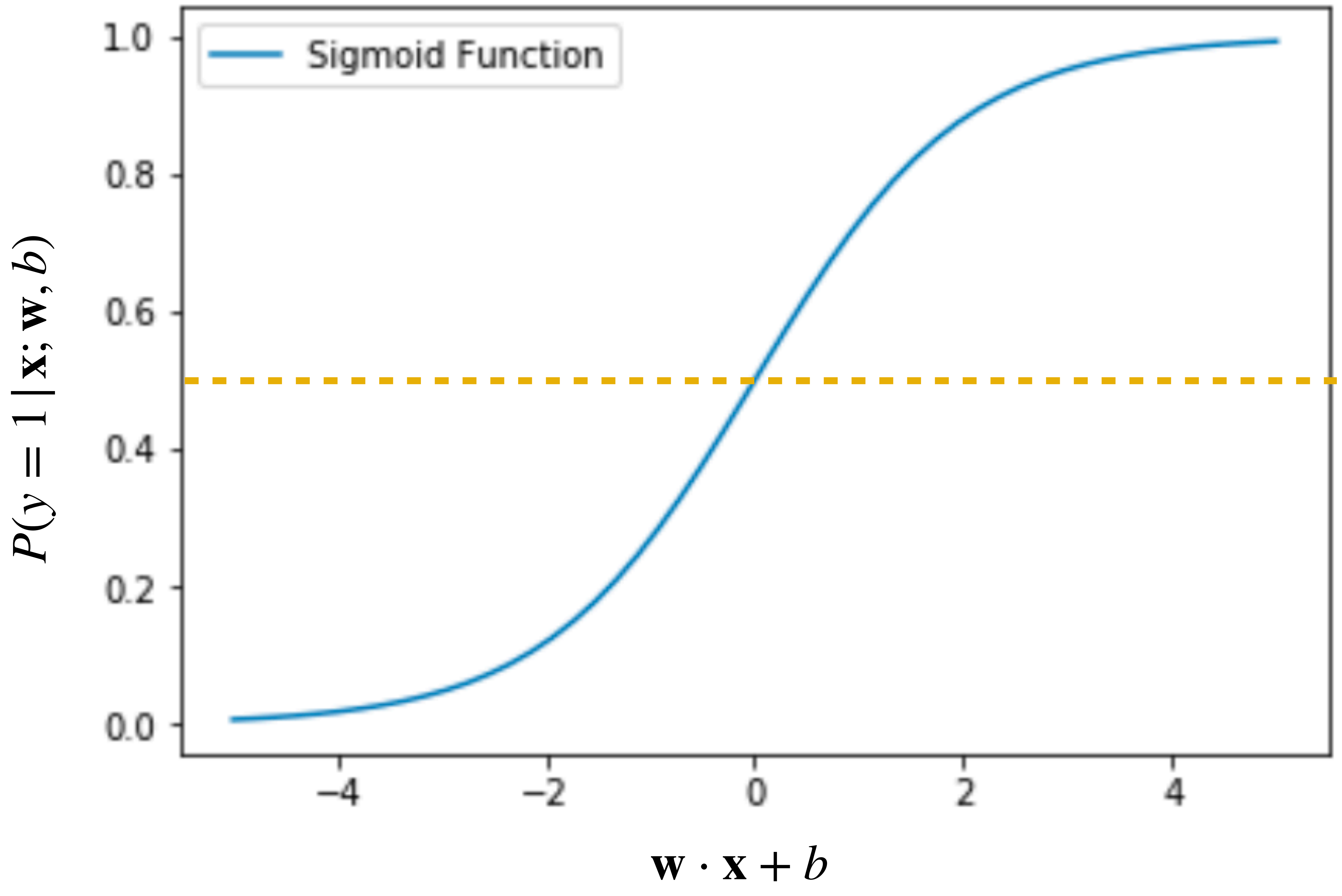
$$\hat{y} = \begin{cases} 1 & \textbf{if } p(y = 1 \mid x) > 0.5 \\ 0 & \textbf{otherwise} \end{cases}$$



Classification Decision

$$\hat{y} = \begin{cases} 1 & \textbf{if } p(y = 1 \mid x) > 0.5 \\ 0 & \textbf{otherwise} \end{cases}$$

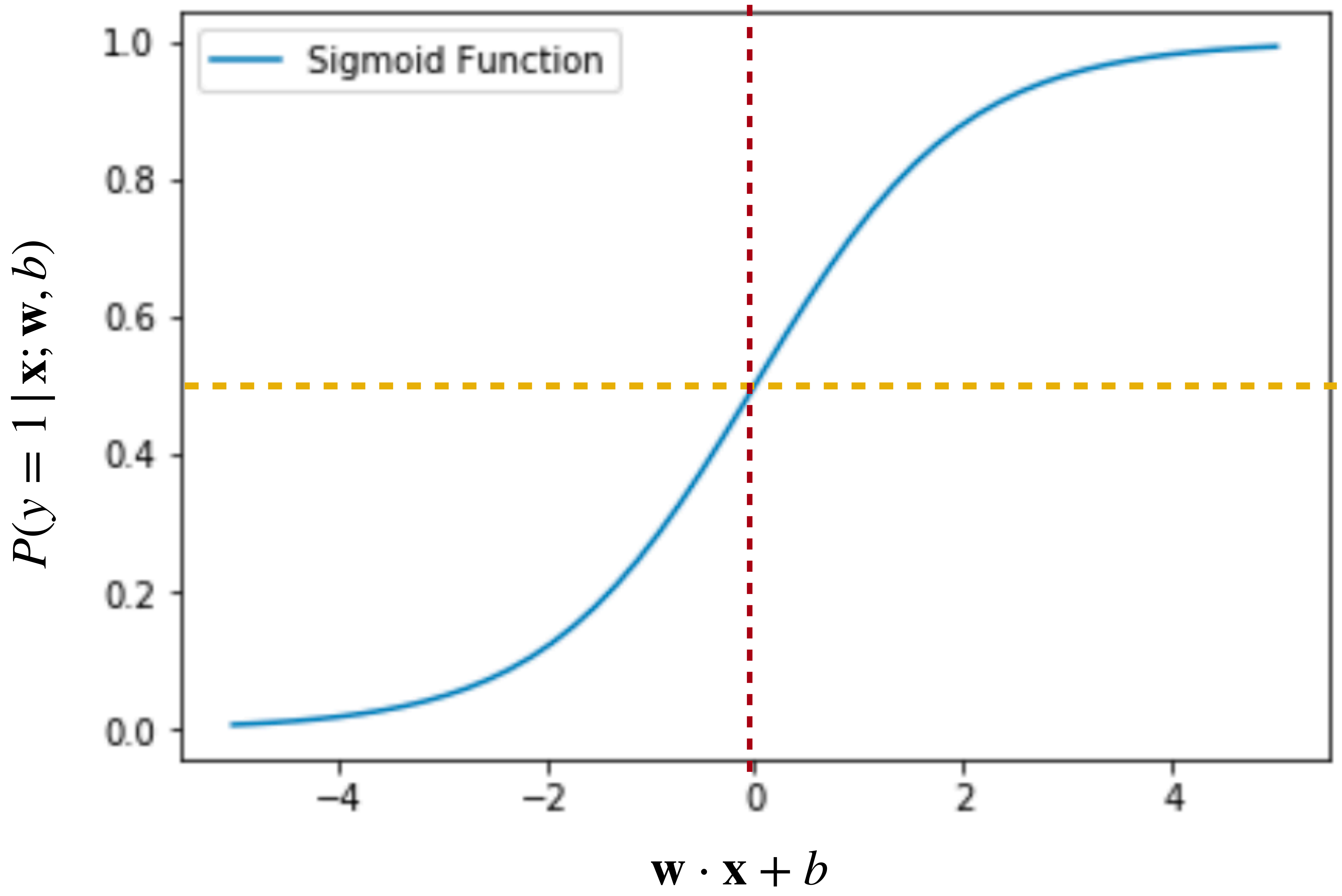
Decision Boundary



Classification Decision

$$\hat{y} = \begin{cases} 1 & \textbf{if } p(y = 1 \mid x) > 0.5 \\ 0 & \textbf{otherwise} \end{cases}$$

Decision Boundary

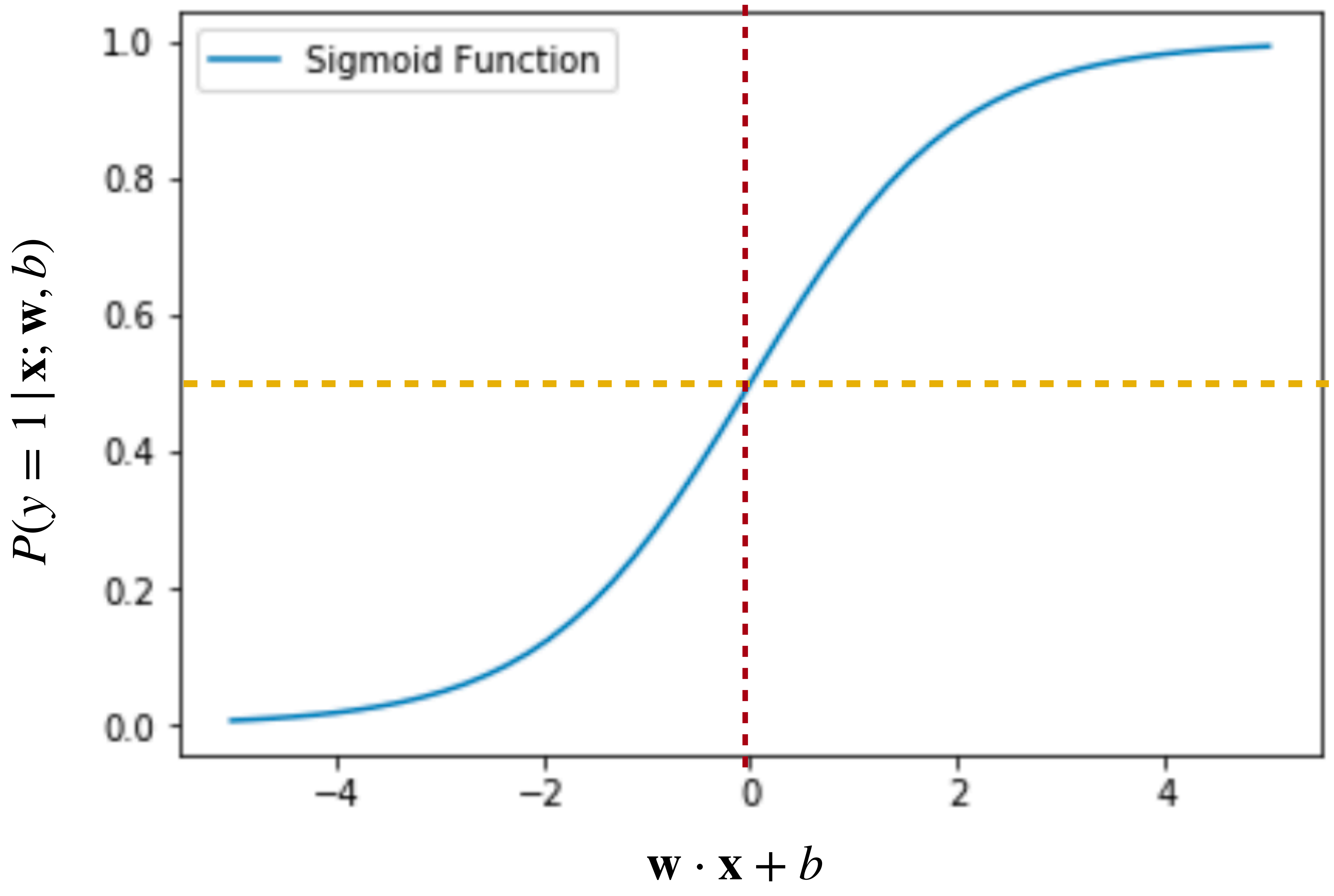


Classification Decision

$$\hat{y} = \begin{cases} 1 & \textbf{if } p(y=1 \mid x) > 0.5 \\ 0 & \textbf{otherwise} \end{cases}$$

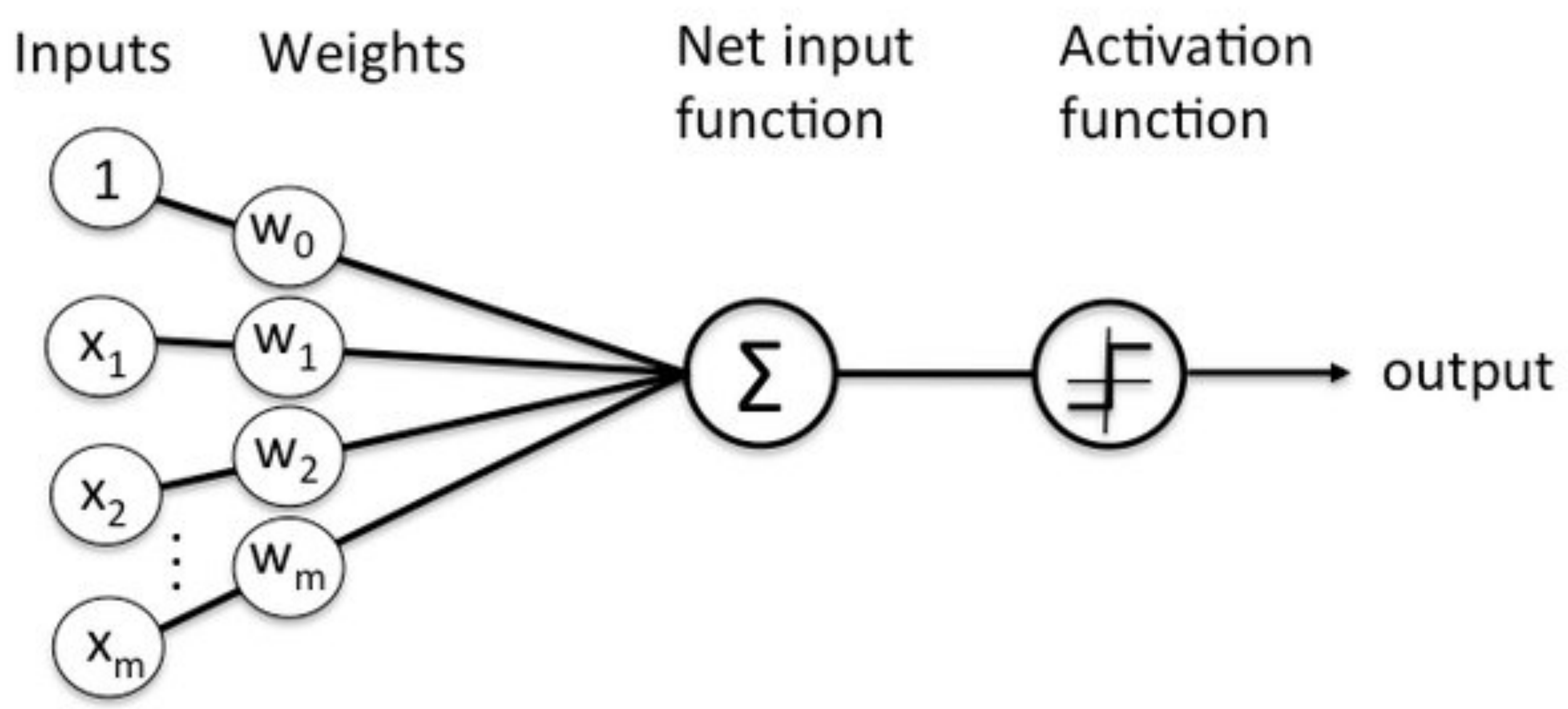
Decision Boundary

$$\hat{y} = \begin{cases} 1 & \textbf{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \textbf{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \end{cases}$$



Another notation

Another notation



But where do the $\mathbf{w}$'s and the b's come from?

But where do the $\mathbf{w}$'s and the b's come from?

- Supervised Classification:
  - We know the correct label y (either 0 or 1) for each x

But where do the $\mathbf{w}$'s and the b's come from?

- Supervised Classification:
  - We know the correct label y (either 0 or 1) for each x
  - But what the system produces is an estimate, $\hat{y}$

But where do the $\mathbf{w}$'s and the b's come from?

- Supervised Classification:
  - We know the correct label y (either 0 or 1) for each x
  - But what the system produces is an estimate, $\hat{y}$
- Set $\mathbf{w}$ and b to minimize the **distance** between our estimate $\hat{y}^{(i)}$ and the true $y^{(i)}$

But where do the $\mathbf{w}$'s and the b's come from?

- Supervised Classification:
  - We know the correct label y (either 0 or 1) for each x
  - But what the system produces is an estimate, $\hat{y}$
- Set $\mathbf{w}$ and b to minimize the **distance** between our estimate $\hat{y}^{(i)}$ and the true $y^{(i)}$
  - We need a distance estimator: a **loss function** or a **cost function**

Loss function

But where do the $\mathbf{w}$'s and the b's come from?

- Supervised Classification:
  - We know the correct label y (either 0 or 1) for each x
  - But what the system produces is an estimate, $\hat{y}$
- Set $\mathbf{w}$ and b to minimize the **distance** between our estimate $\hat{y}^{(i)}$ and the true $y^{(i)}$
  - We need a distance estimator: a **loss function** or a **cost function**
  - We need an **optimization algorithm** to update $\mathbf{w}$ and b to minimize the loss.

Loss function

Optimization Algorithm

Lecture Outline

- Recap
  - Smoothing
  - Basics of Supervised Machine Learning
    - Data: Preprocessing and Feature Extraction
- Quiz
- Announcements
- Basics of Supervised Machine Learning
  - I. Data: Preprocessing and Feature Extraction
  - II. Model:
    - I. Logistic Regression
  - III. Loss
  - IV. Optimization Algorithm
  - V. Inference

III. Loss: Cross-Entropy

The distance between $\hat{y}$ and y

The distance between $\hat{y}$ and y

- We want to know how far is the classifier output:
  - $\hat{y} = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$

The distance between $\hat{y}$ and y

- We want to know how far is the classifier output:
  - $\hat{y} = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$
- From the true (ground truth / gold standard) label:
  - $y \in \{0,1\}$

The distance between $\hat{y}$ and y

- We want to know how far is the classifier output:
  - $\hat{y} = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$
- From the true (ground truth / gold standard) label:
  - $y \in \{0,1\}$
- This difference is called the loss or cost
  - $L(\hat{y}, y)$ = how much $\hat{y}$ differs from y
  - In other words, how much would you lose if you mispredicted
  - Or how much would it cost you to mispredict

Remember maximum likelihood?

Suppose we flip the coin four times and see (H, H, H, T). What is p?

$p = 3/4 = 0.75$ maximizes the probability of data sequence (H,H,H,T)

maximum likelihood estimate

Remember maximum likelihood?

- Here: **conditional** maximum likelihood estimation

Suppose we flip the coin four times and see (H, H, H, T).

What is p?

$p = 3/4 = 0.75$ maximizes the probability of data sequence (H,H,H,T)

maximum likelihood estimate

Remember maximum likelihood?

- Here: **conditional** maximum likelihood estimation
- We choose the parameters $\mathbf{w}, b$ that maximize

Suppose we flip the coin four times and see (H, H, H, T).

What is p?

$p = 3/4 = 0.75$ maximizes the probability of data sequence (H,H,H,T)

maximum likelihood estimate

Remember maximum likelihood?

- Here: **conditional** maximum likelihood estimation
- We choose the parameters $\mathbf{w}, b$ that maximize
  - the log probability

Suppose we flip the coin four times and see (H, H, H, T).

What is p?

$p = 3/4 = 0.75$ maximizes the probability of data sequence (H,H,H,T)

maximum likelihood estimate

USC Viterbi

Remember maximum likelihood?

Suppose we flip the coin four times and see (H, H, H, T).

What is p?

- Here: **conditional** maximum likelihood estimation
- We choose the parameters $\mathbf{w}, b$ that maximize
  - the log probability
    - of the true y labels in the training data

$p = 3/4 = 0.75$ maximizes the probability of data sequence (H,H,H,T)

maximum likelihood estimate

Remember maximum likelihood?

- Here: **conditional** maximum likelihood estimation
- We choose the parameters $\mathbf{w}, b$ that maximize
    - the log probability
        - of the true y labels in the training data
        - **given** the observations x

Suppose we flip the coin four times and see (H, H, H, T). What is p?

$p = 3/4 = 0.75$ maximizes the probability of data sequence (H,H,H,T)

maximum likelihood estimate

Remember maximum likelihood?

Suppose we flip the coin four times and see (H, H, H, T).

What is p?

- Here: **conditional** maximum likelihood estimation
- We choose the parameters $\mathbf{w}, b$ that maximize
  - the log probability
    - of the true y labels in the training data
    - **given** the observations x

$$\max \log p(y \mid x)$$

$p = 3/4 = 0.75$ maximizes the probability of data sequence (H,H,H,T)

maximum likelihood estimate

Maximizing conditional likelihood

Maximizing conditional likelihood

Goal: maximize probability of the correct label $p(y \mid x)$

Maximizing conditional likelihood

For a single observation

Goal: maximize probability of the correct label $p(y|x)$

Maximizing conditional likelihood

For a single observation

Goal: maximize probability of the correct label $p(y\,|\,x)$

Since there are only 2 discrete outcomes (0 or 1) we can express the probability $p(y\,|\,\mathbf{x})$ from our classifier (the thing we want to maximize) as

Maximizing conditional likelihood

For a single observation

Goal: maximize probability of the correct label $p(y \mid x)$

Since there are only 2 discrete outcomes (0 or 1) we can express the probability $p(y \mid \mathbf{x})$ from our classifier (the thing we want to maximize) as

$$p(y \mid x) = \hat{y}^y (1 - \hat{y})^{1-y}$$

Maximizing conditional likelihood

For a single observation

Goal: maximize probability of the correct label $p(y \mid x)$

Since there are only 2 discrete outcomes (0 or 1) we can express the probability $p(y \mid \mathbf{x})$ from our classifier (the thing we want to maximize) as

$$p(y \mid x) = \hat{y}^{y}(1 - \hat{y})^{1-y}$$

	$\hat{y} = 0$	$\hat{y} = 1$

Maximizing conditional likelihood

For a single observation

Goal: maximize probability of the correct label $p(y|x)$

Since there are only 2 discrete outcomes (0 or 1) we can express the probability $p(y|\mathbf{x})$ from our classifier (the thing we want to maximize) as

$$p(y|x) = \hat{y}^y(1-\hat{y})^{1-y}$$

	$\hat{y}=0$	$\hat{y}=1$
$y=0$	1	0

Maximizing conditional likelihood

For a single observation

Goal: maximize probability of the correct label $p(y|x)$

Since there are only 2 discrete outcomes (0 or 1) we can express the probability $p(y|\mathbf{x})$ from our classifier (the thing we want to maximize) as

$$p(y|x) = \hat{y}^y (1 - \hat{y})^{1-y}$$

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	1	0
$y = 1$	0	1

Maximizing conditional likelihood

Maximizing conditional likelihood

Goal: maximize probability of the correct label $p(y \mid \mathbf{x})$

Maximizing conditional likelihood

Goal: maximize probability of the correct label $p(y \mid \mathbf{x})$

Maximize: $p(y \mid x) = \hat{y}^y (1 - \hat{y})^{1-y}$

Maximizing conditional likelihood

Goal: maximize probability of the correct label $p(y \mid \mathbf{x})$

Maximize: $p(y \mid x) = \hat{y}^y(1 - \hat{y})^{1-y}$

Now take the log of both sides

$$\log p(y \mid x) = \log(\hat{y}^y(1 - \hat{y})^{1-y})$$

Maximizing conditional likelihood

Goal: maximize probability of the correct label $p(y \mid \mathbf{x})$

Maximize: $p(y \mid x) = \hat{y}^y (1 - \hat{y})^{1-y}$

Now take the log of both sides

$$\begin{aligned} \log p(y \mid x) &= \log(\hat{y}^y (1 - \hat{y})^{1-y}) \\ &= y \log \hat{y} + (1 - y) \log(1 - \hat{y}) \end{aligned}$$

Maximizing conditional likelihood

Goal: maximize probability of the correct label $p(y \mid \mathbf{x})$

Maximize: $p(y \mid x) = \hat{y}^y (1 - \hat{y})^{1-y}$

Now take the log of both sides

$$\log p(y \mid x) = \log(\hat{y}^y (1 - \hat{y})^{1-y})$$
$$= y \log \hat{y} + (1 - y) \log(1 - \hat{y})$$

Whatever values maximize $\log p(y \mid x)$ will also maximize $p(y \mid x)$

Maximizing conditional likelihood

Goal: maximize probability of the correct label $p(y \mid \mathbf{x})$

Maximize: $p(y \mid x) = \hat{y}^y(1 - \hat{y})^{1-y}$

Now take the log of both sides

$$\begin{aligned}\log p(y \mid x) &= \log(\hat{y}^y(1 - \hat{y})^{1-y}) \\ &= y \log \hat{y} + (1 - y)\log(1 - \hat{y})\end{aligned}$$

Whatever values maximize $\log p(y \mid x)$ will also maximize $p(y \mid x)$

Why does this work?

Minimizing negative log likelihood

$$
\begin{aligned}
\log p(y \mid x) &= \log(\hat{y}^y (1-\hat{y})^{1-y}) \\
&= y \log \hat{y} + (1-y)\log(1-\hat{y})
\end{aligned}
$$

Minimizing negative log likelihood

Goal: maximize probability of the correct label $p(y \mid \mathbf{x})$

Maximize: $$\log p(y \mid x) = \log(\hat{y}^{y}(1 - \hat{y})^{1-y})$$
$$= y \log \hat{y} + (1 - y)\log(1 - \hat{y})$$

Minimizing negative log likelihood

Goal: maximize probability of the correct label $p(y \mid \mathbf{x})$

Maximize: $$\begin{aligned} \log p(y \mid x) &= \log(\hat{y}^y (1-\hat{y})^{1-y}) \\ &= y \log \hat{y} + (1-y) \log(1-\hat{y}) \end{aligned}$$

Now flip the sign for something to minimize (we minimize the loss / cost)

Minimizing negative log likelihood

Goal: maximize probability of the correct label $p(y \mid \mathbf{x})$

Maximize:
$$\log p(y \mid x) = \log(\hat{y}^y (1 - \hat{y})^{1-y})$$
$$= y \log \hat{y} + (1 - y)\log(1 - \hat{y})$$

Now flip the sign for something to minimize (we minimize the loss / cost)

Minimize: $L_{CE}(y, \hat{y}) = -\log p(y \mid x) = -[y \log \hat{y} + (1 - y)\log(1 - \hat{y})]$

Cross-Entropy Loss

Minimizing negative log likelihood

Goal: maximize probability of the correct label $p(y \mid \mathbf{x})$

Maximize: $$\log p(y \mid x) = \log(\hat{y}^y (1 - \hat{y})^{1-y}) = y \log \hat{y} + (1 - y) \log(1 - \hat{y})$$

Now flip the sign for something to minimize (we minimize the loss / cost)

Minimize: $$L_{CE}(y, \hat{y}) = -\log p(y \mid x) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})] = -[y \log \sigma(\mathbf{w} \cdot \mathbf{x} + b) + (1 - y) \log \sigma(-(\mathbf{w} \cdot \mathbf{x} + b))]$$

Cross-Entropy Loss

Minimizing negative log likelihood

Goal: maximize probability of the correct label $p(y \mid \mathbf{x})$

Measures how well the training data matches the proposed model distribution and how good the model distribution is

Maximize: $$\log p(y \mid x) = \log(\hat{y}^y (1 - \hat{y})^{1-y})$$
$$= y \log \hat{y} + (1 - y) \log(1 - \hat{y})$$

Now flip the sign for something to minimize (we minimize the loss / cost)

Minimize: $$L_{CE}(y, \hat{y}) = -\log p(y \mid x) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$
$$= -[y \log \sigma(\mathbf{w} \cdot \mathbf{x} + b) + (1 - y) \log \sigma(-(\mathbf{w} \cdot \mathbf{x} + b))]$$

Cross-Entropy Loss

Lecture Outline

- Recap
  - Smoothing
  - Basics of Supervised Machine Learning
    - Data: Preprocessing and Feature Extraction
- Quiz
- Announcements
- Basics of Supervised Machine Learning
  I. Data: Preprocessing and Feature Extraction
  II. Model:
      I. Logistic Regression
  III. Loss
  IV. Optimization Algorithm
  V. Inference

IV. Optimization:
Stochastic Gradient Descent

Our goal: minimize the loss

- Loss function is parameterized by weights: $\theta = [\mathbf{w}; b]$
- We will represent $\hat{y}$ as $f(x; \theta)$ to make the dependence on θ more obvious

$$L_{CE}(f(\mathbf{x}^{(i)}; \theta), y^{(i)})$$

Our goal: minimize the loss

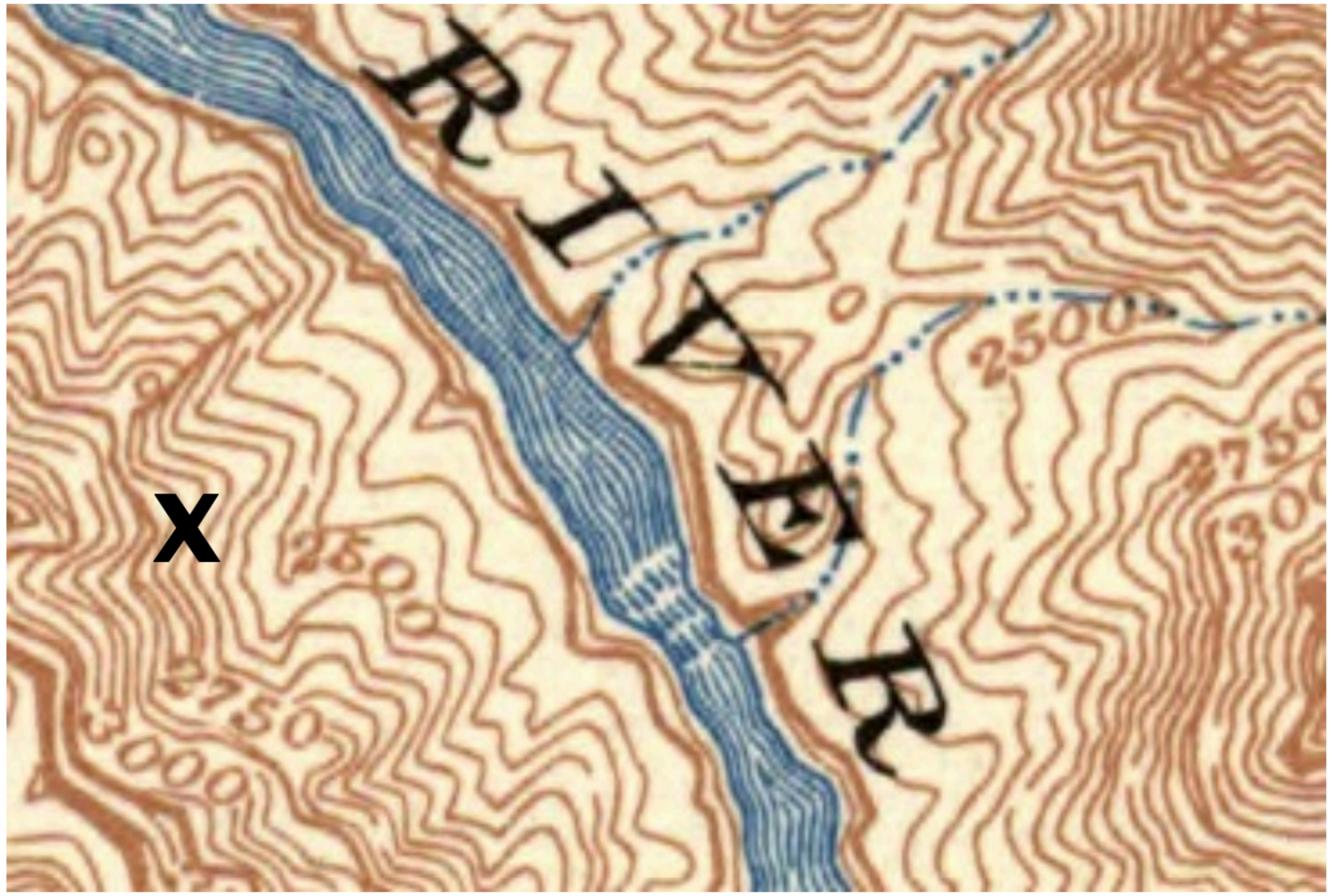
- Loss function is parameterized by weights: $\theta = [\mathbf{w}; b]$
- We will represent $\hat{y}$ as $f(x;\theta)$ to make the dependence on θ more obvious

We want the weights that minimize the loss, averaged over all examples:

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} L_{CE}(f(\mathbf{x}^{(i)};\theta), y^{(i)})$$

Intuition for gradient descent

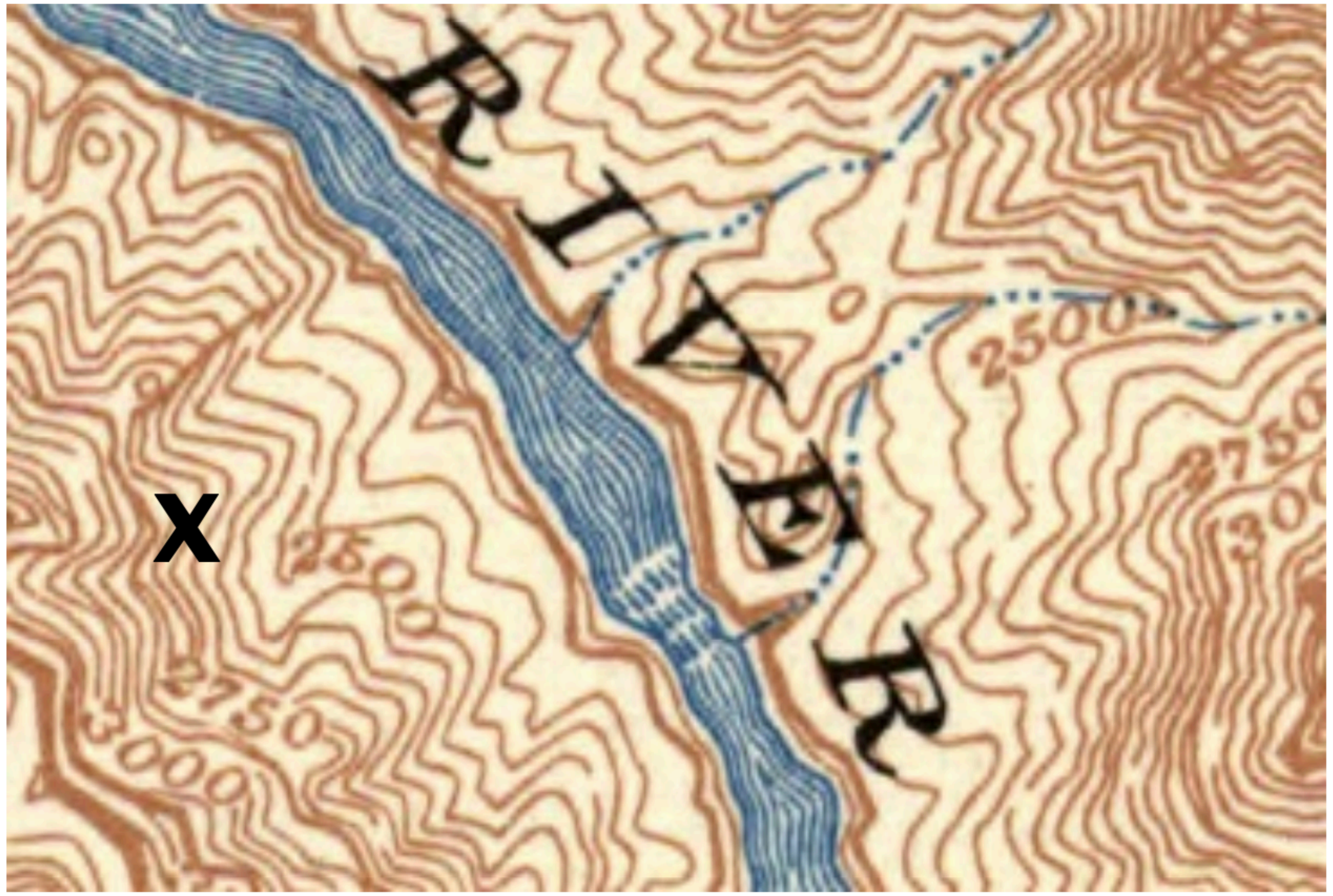
How to get to the bottom of the river canyon?



Intuition for gradient descent

How to get to the bottom of the river canyon?

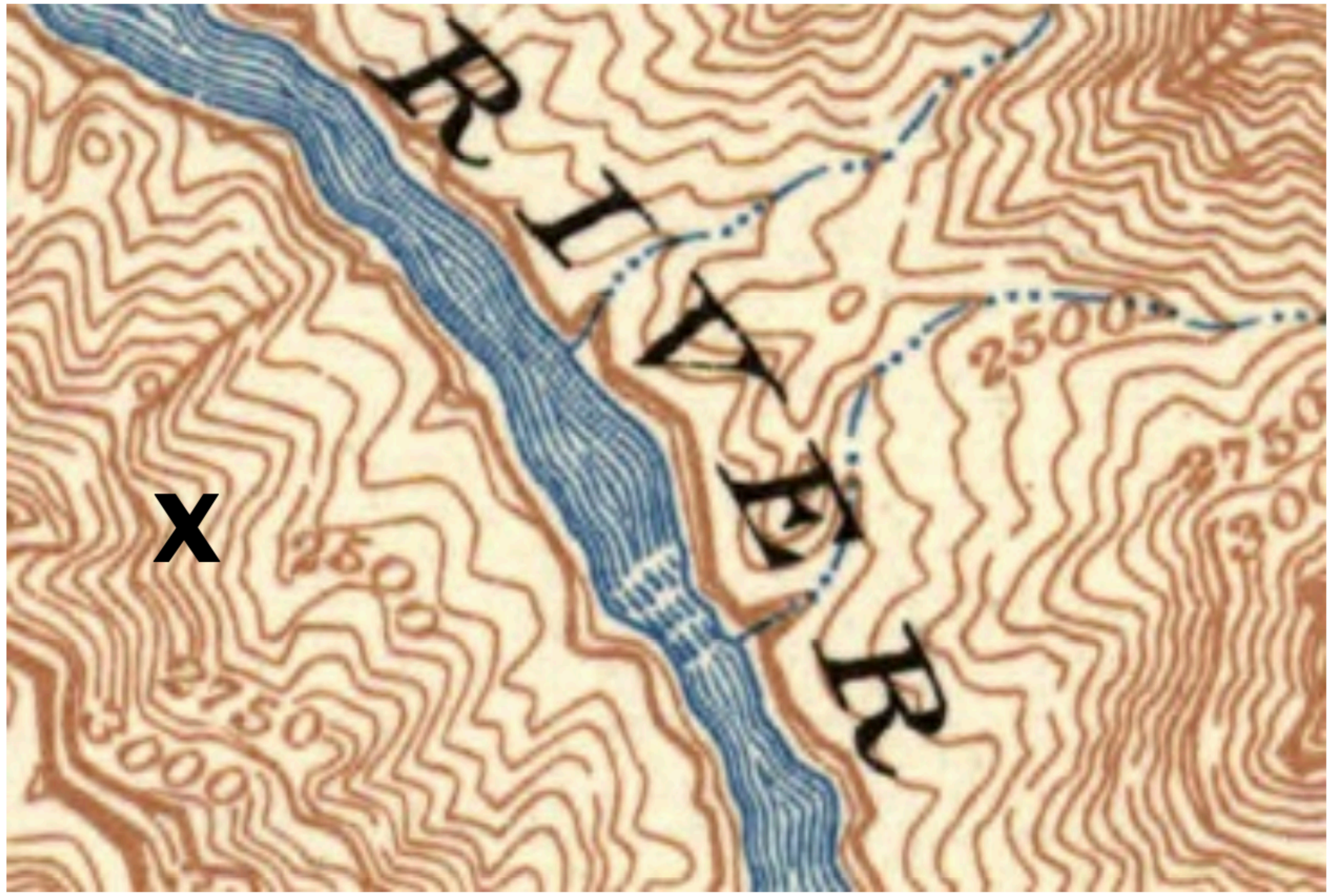
- Look around 360°



Intuition for gradient descent

How to get to the bottom of the river canyon?

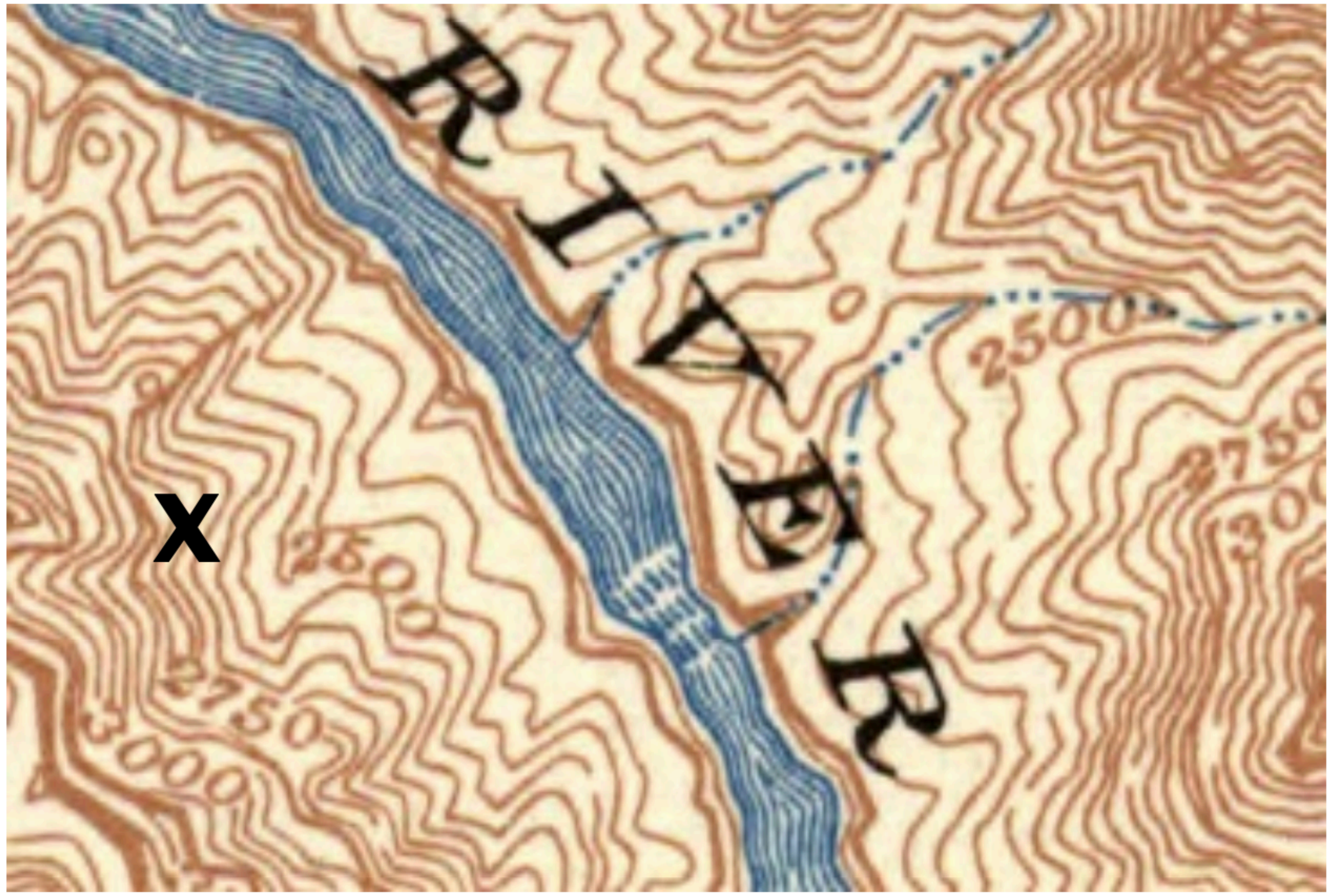
- Look around 360°
- Find the direction of steepest slope down



Intuition for gradient descent

How to get to the bottom of the river canyon?

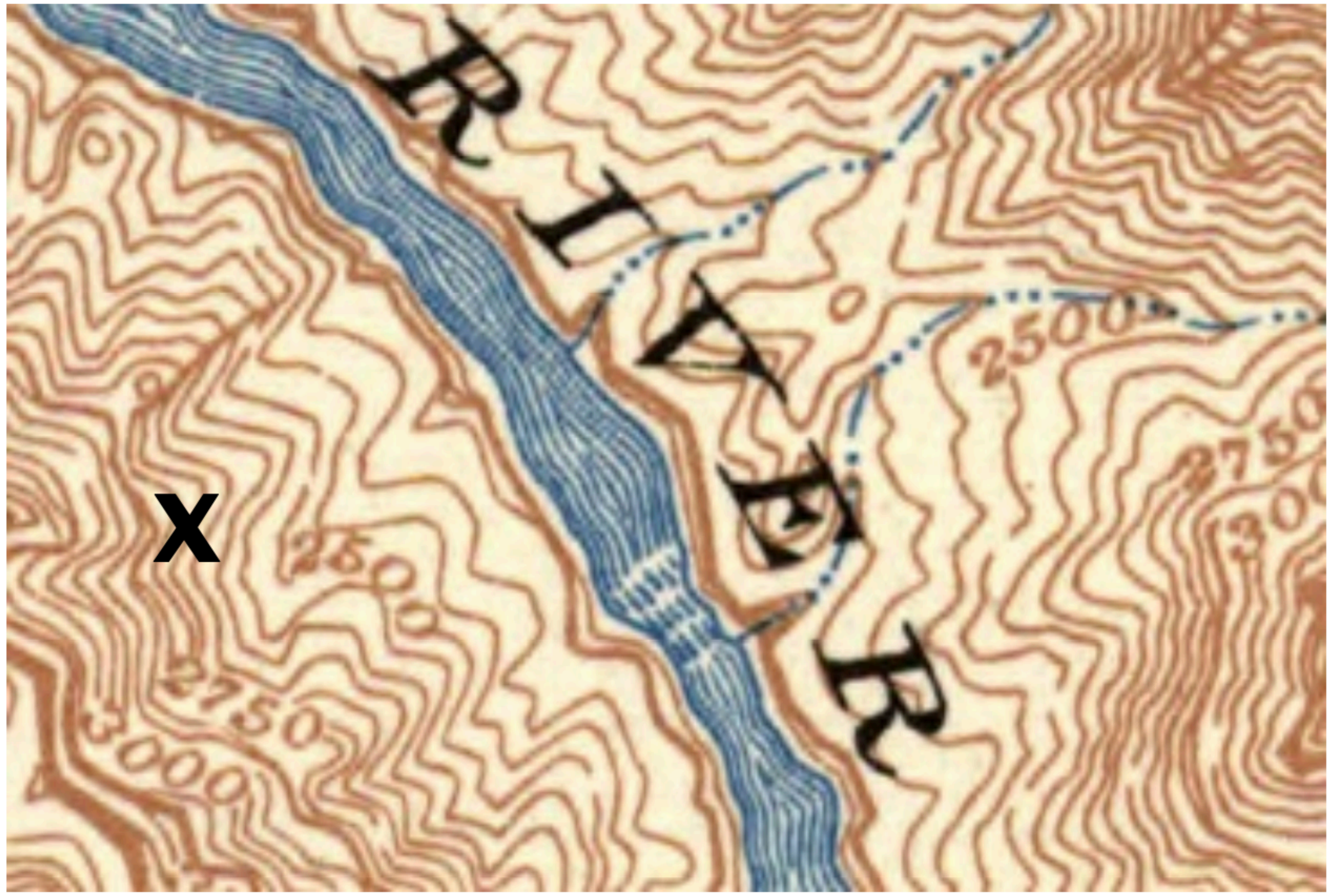
- Look around 360°
- Find the direction of steepest slope down
- Go that way



Intuition for gradient descent

How to get to the bottom of the river canyon?

- Look around 360°
- Find the direction of steepest slope down
- Go that way



What if multiple equally good alternatives?

Logistic Regression: Loss

Convex function

Image Credit: Medium

Logistic Regression: Loss

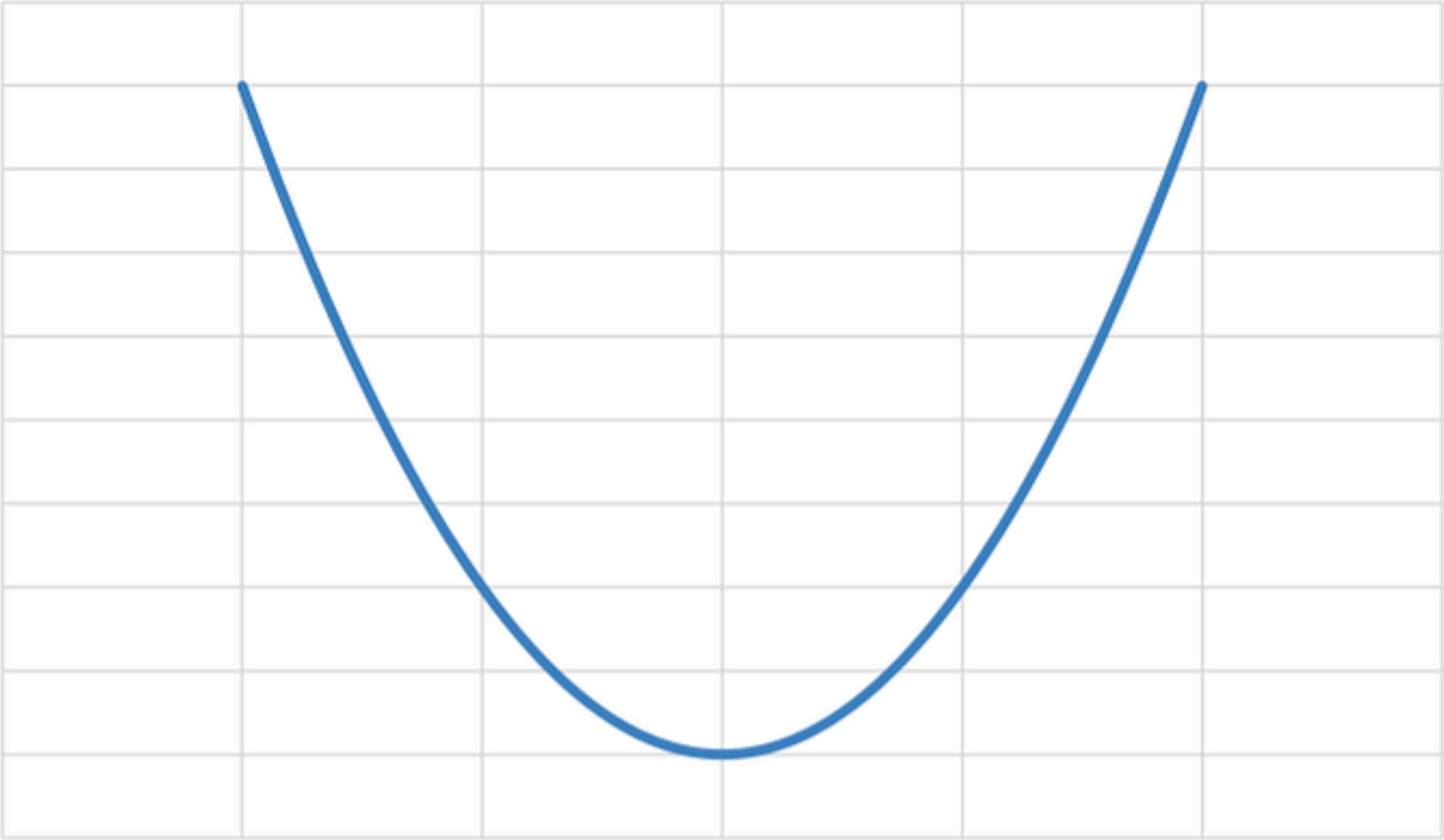


Image Credit: Medium

Convex function

- Has only one option for steepest gradient

Logistic Regression: Loss

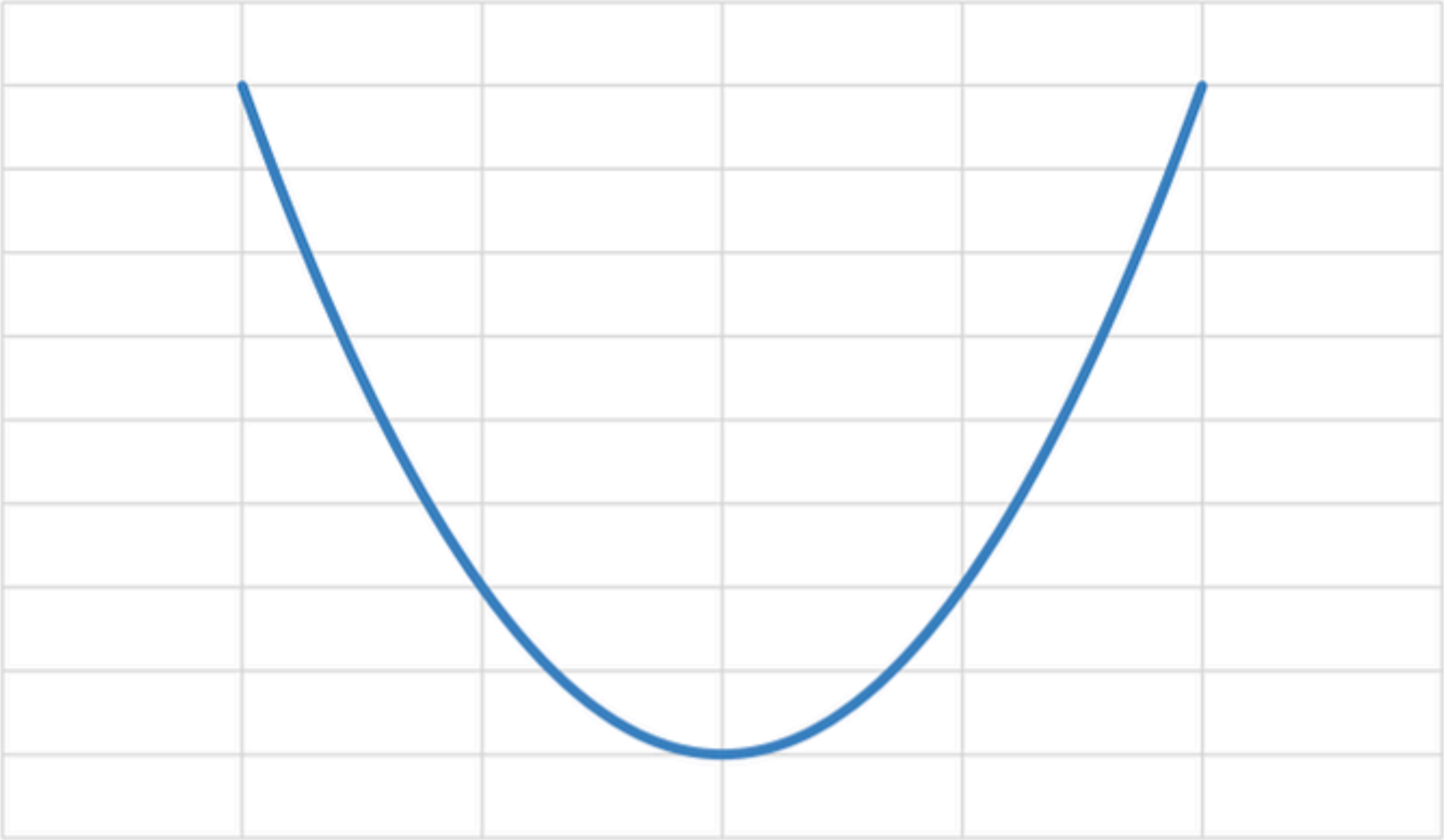


Image Credit: Medium

Convex function

- Has only one option for steepest gradient
  - Or one minimum

Logistic Regression: Loss

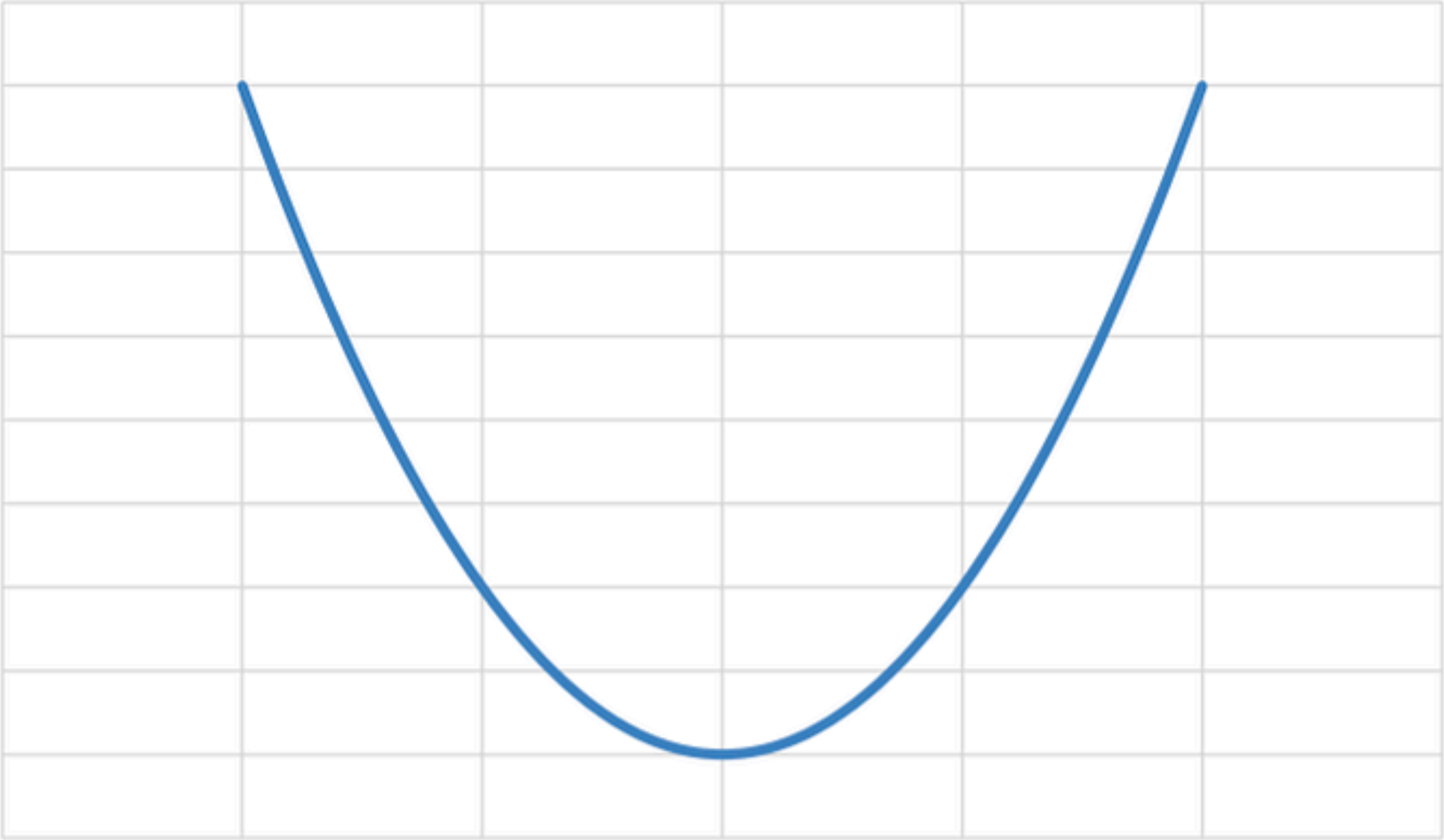
Convex function

Image Credit: Medium

- Has only one option for steepest gradient
  - Or one minimum
- Gradient descent starting from any point is guaranteed to find the minimum

USC Viterbi

Logistic Regression: Loss



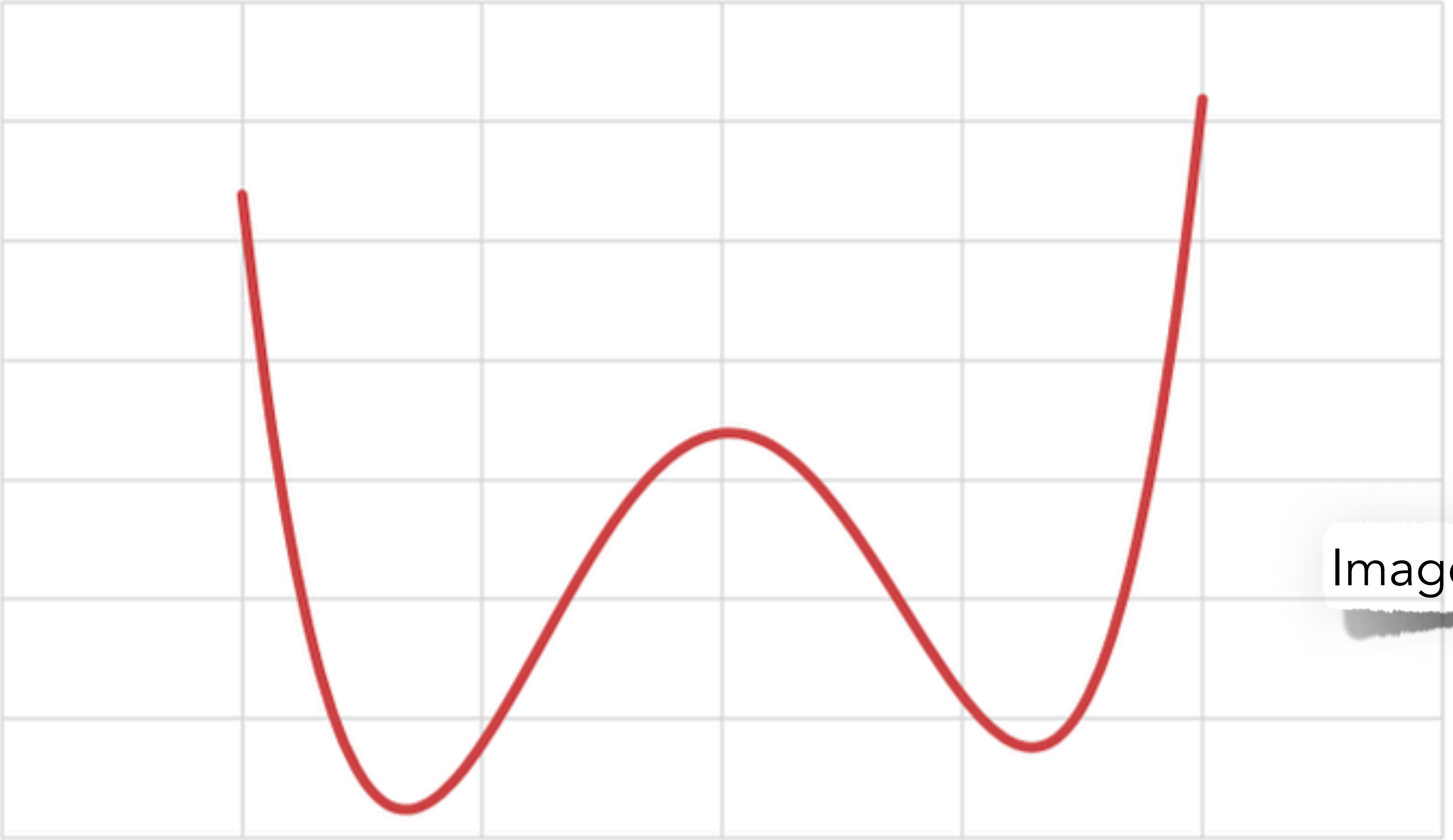


Image Credit: Medium

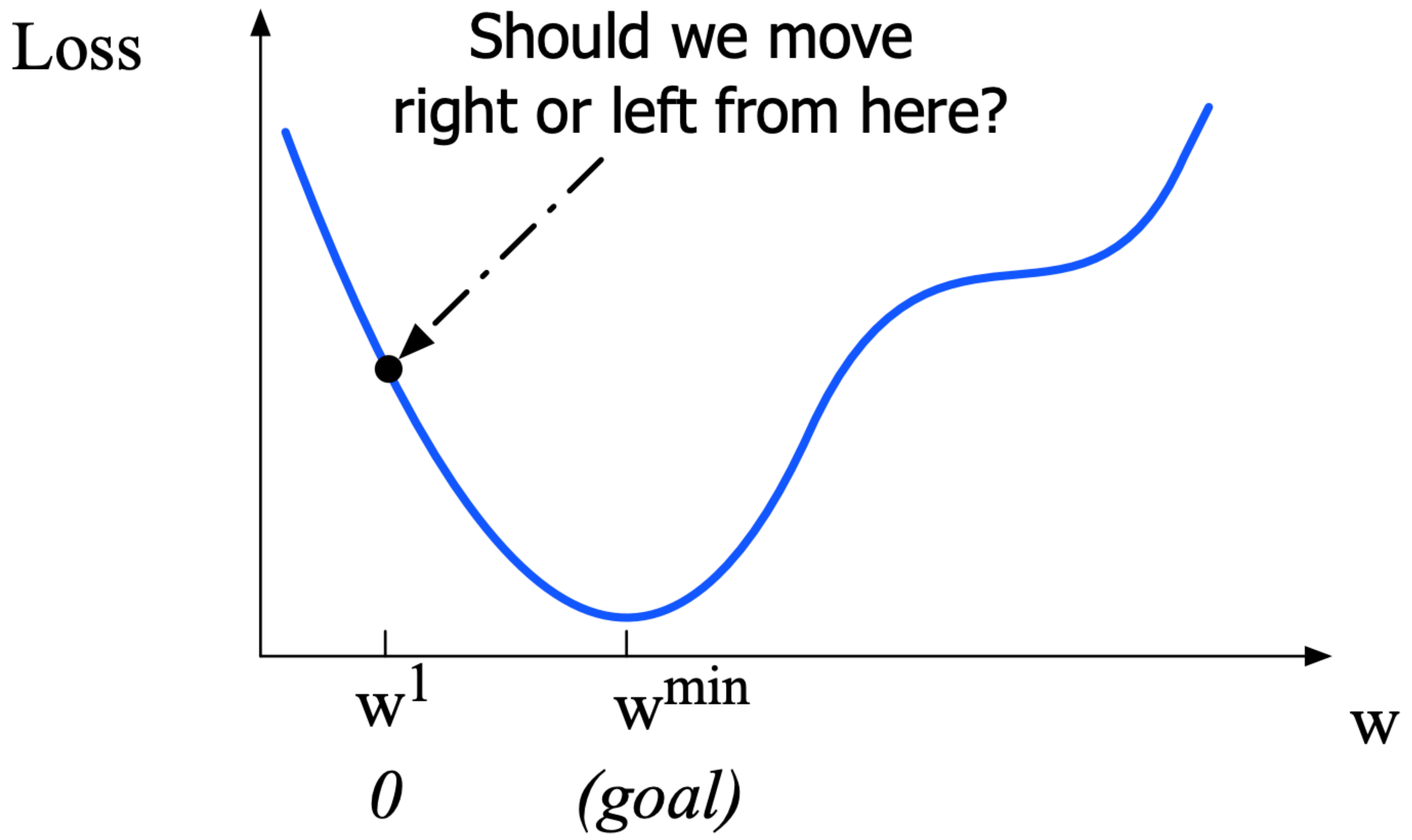
Convex function

Non-convex function

- Has only one option for steepest gradient
  - Or one minimum
- Gradient descent starting from any point is guaranteed to find the minimum

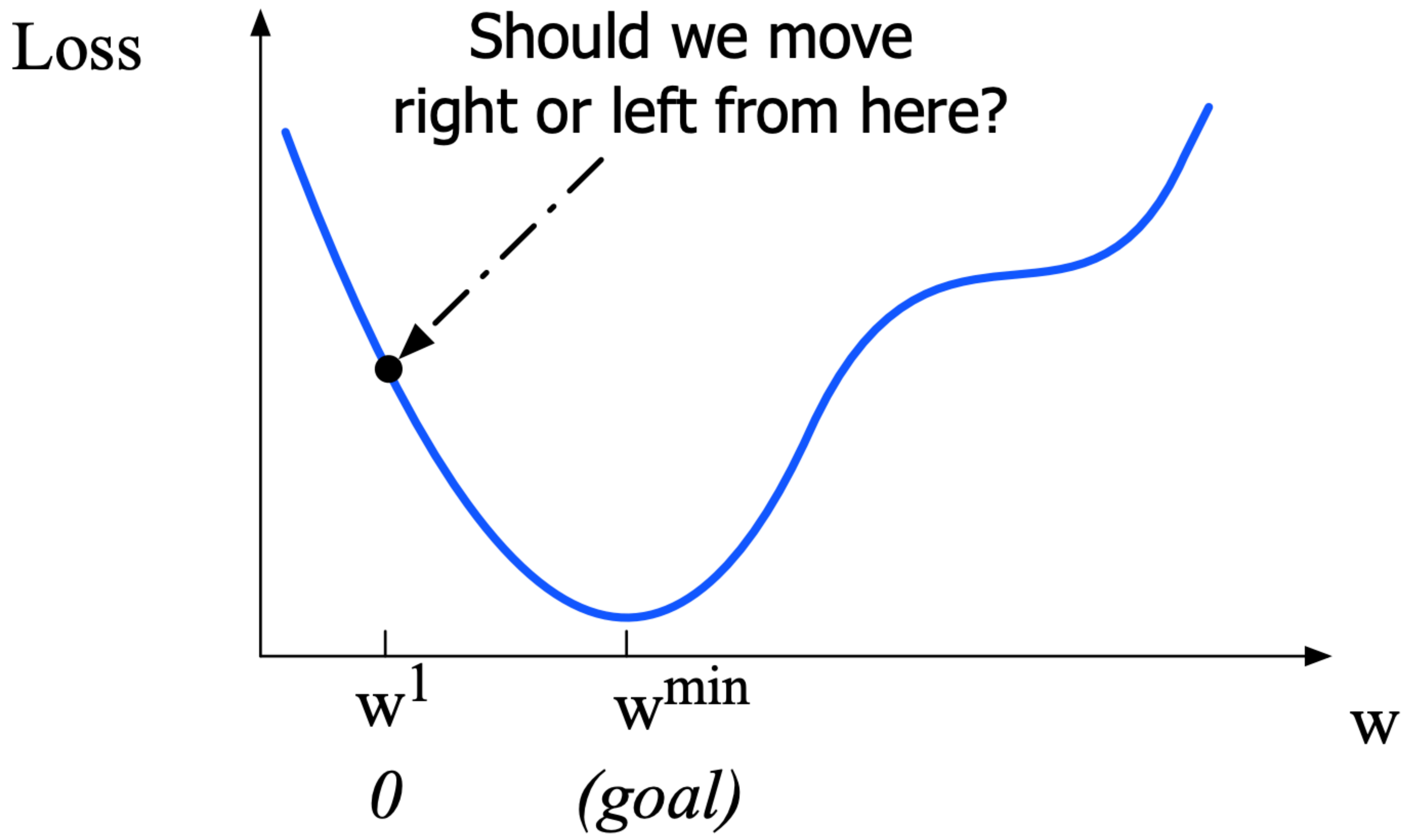
Neural Networks - multiple alternatives

Consider: a single scalar w



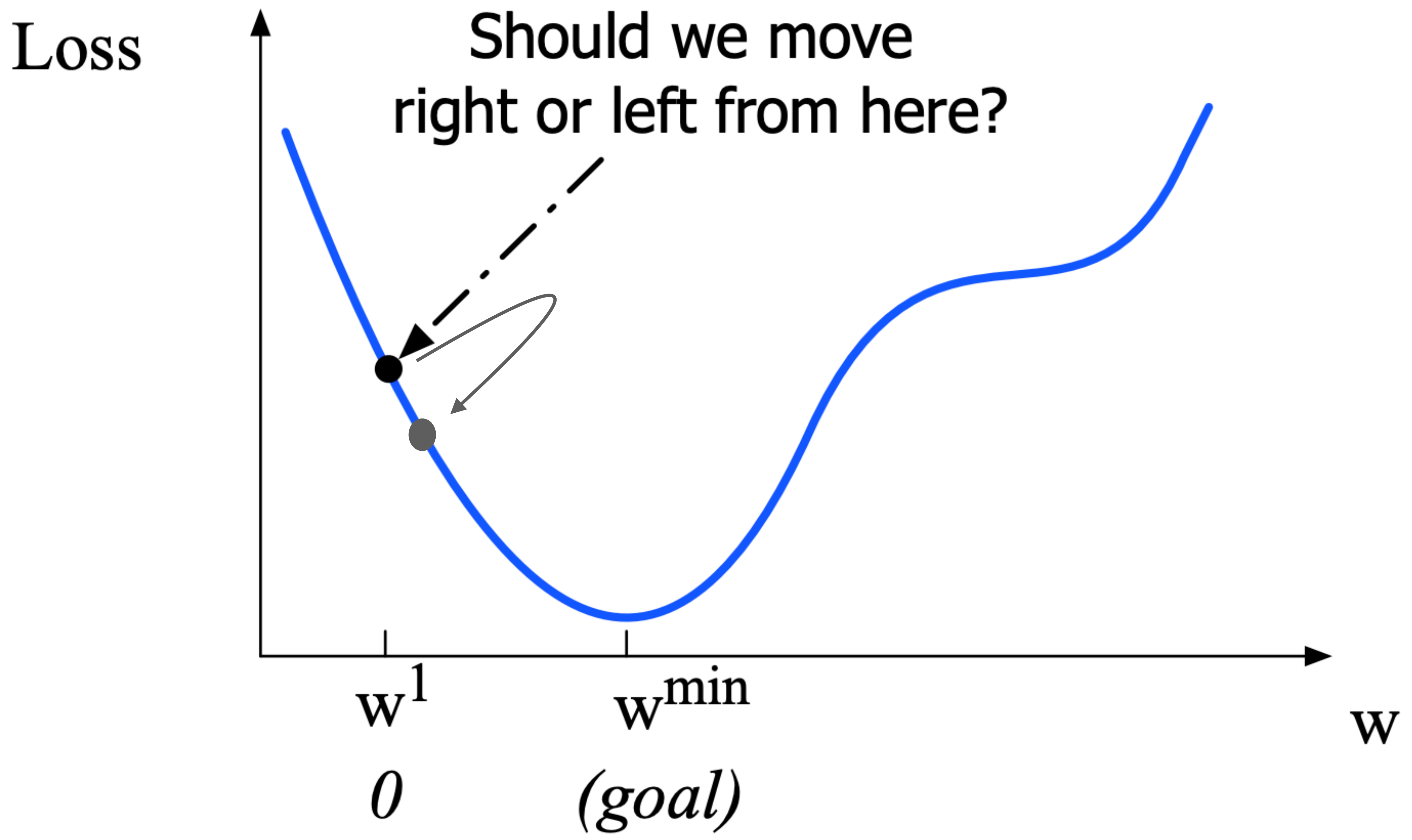
Consider: a single scalar w

Given current w, should we make it bigger or smaller?



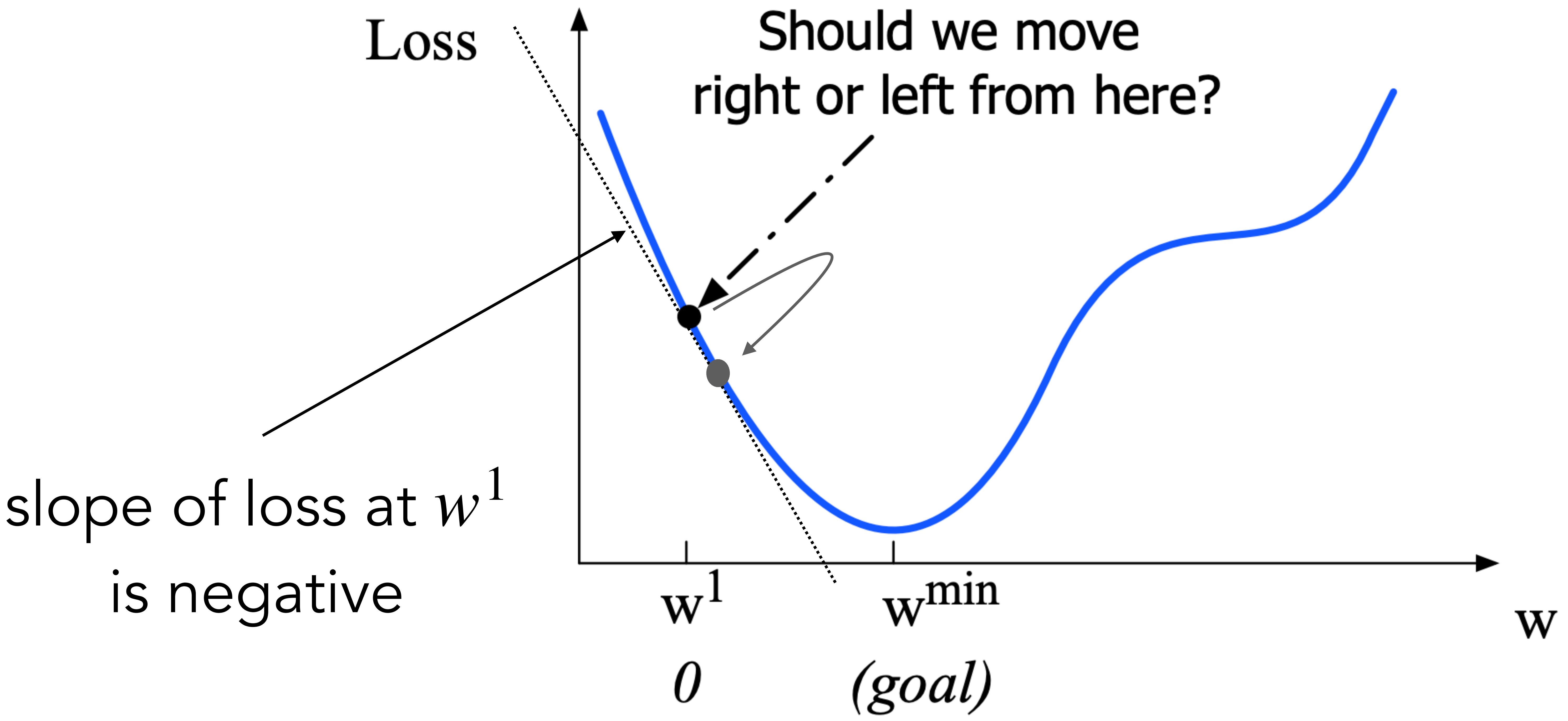
Consider: a single scalar w

Given current w, should we make it bigger or smaller?



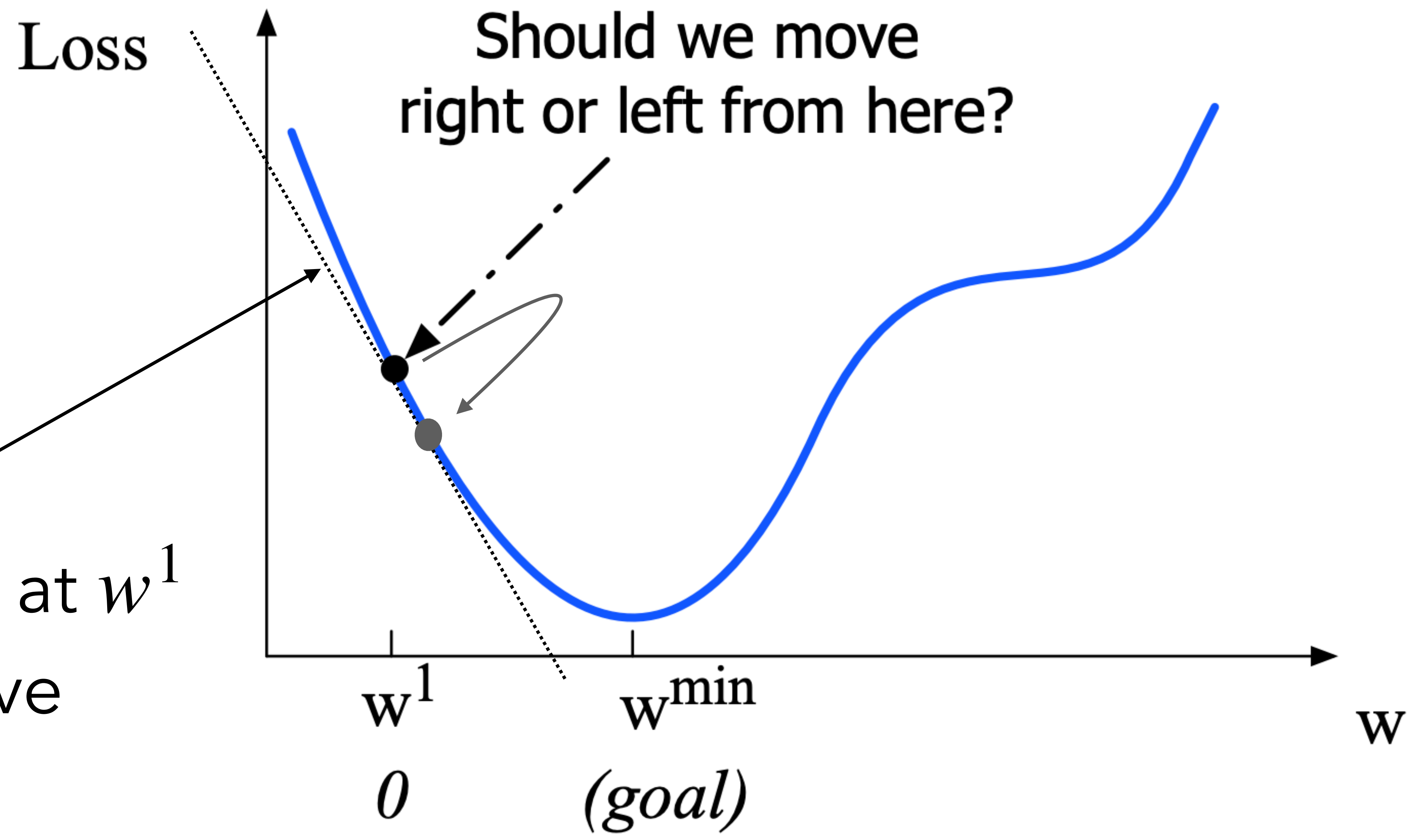
Consider: a single scalar w

Given current w, should we make it bigger or smaller?



Consider: a single scalar w

Given current w, should we make it bigger or smaller?



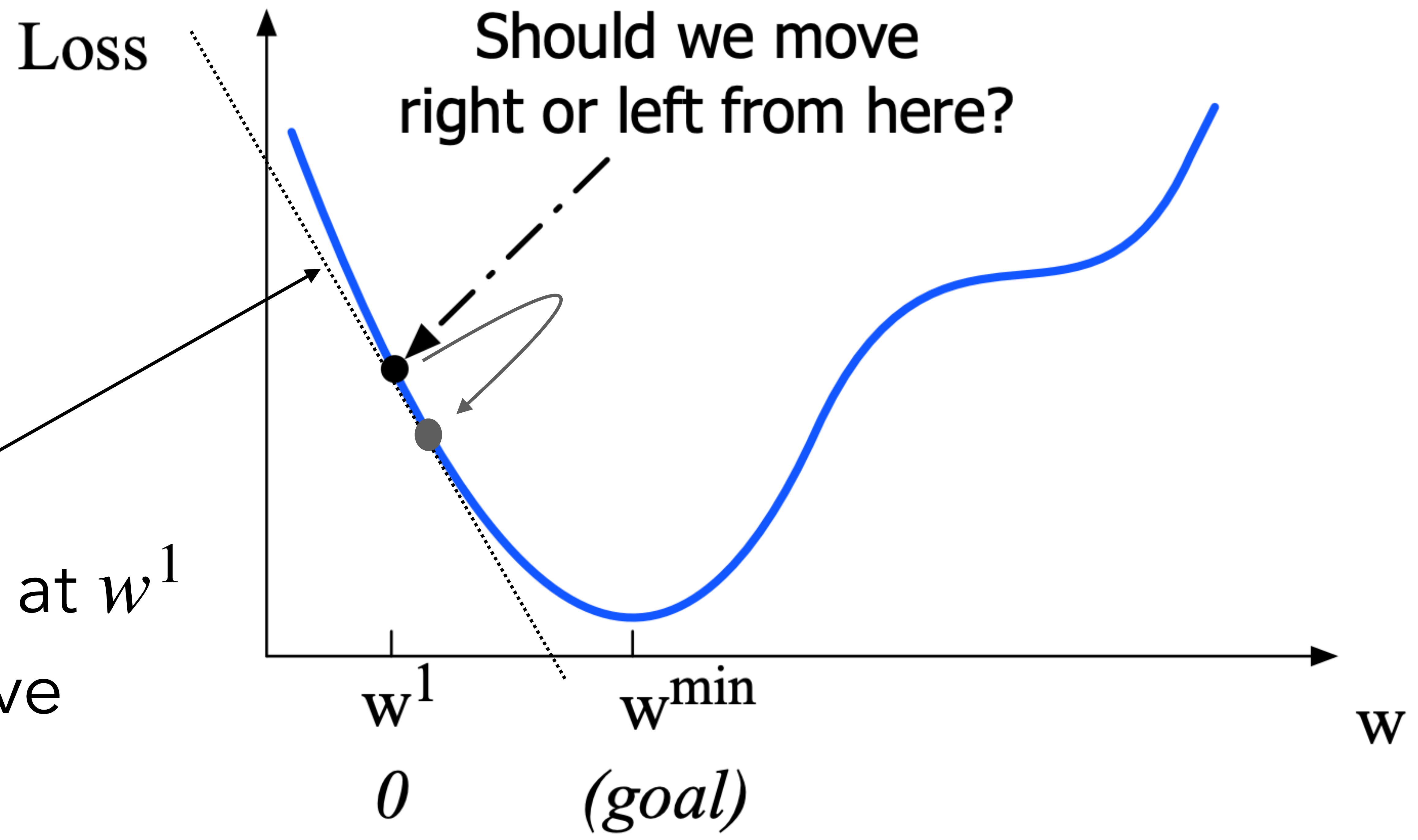
slope of loss at w^1
is negative

need to move positive

Consider: a single scalar w

Given current w, should we make it bigger or smaller?

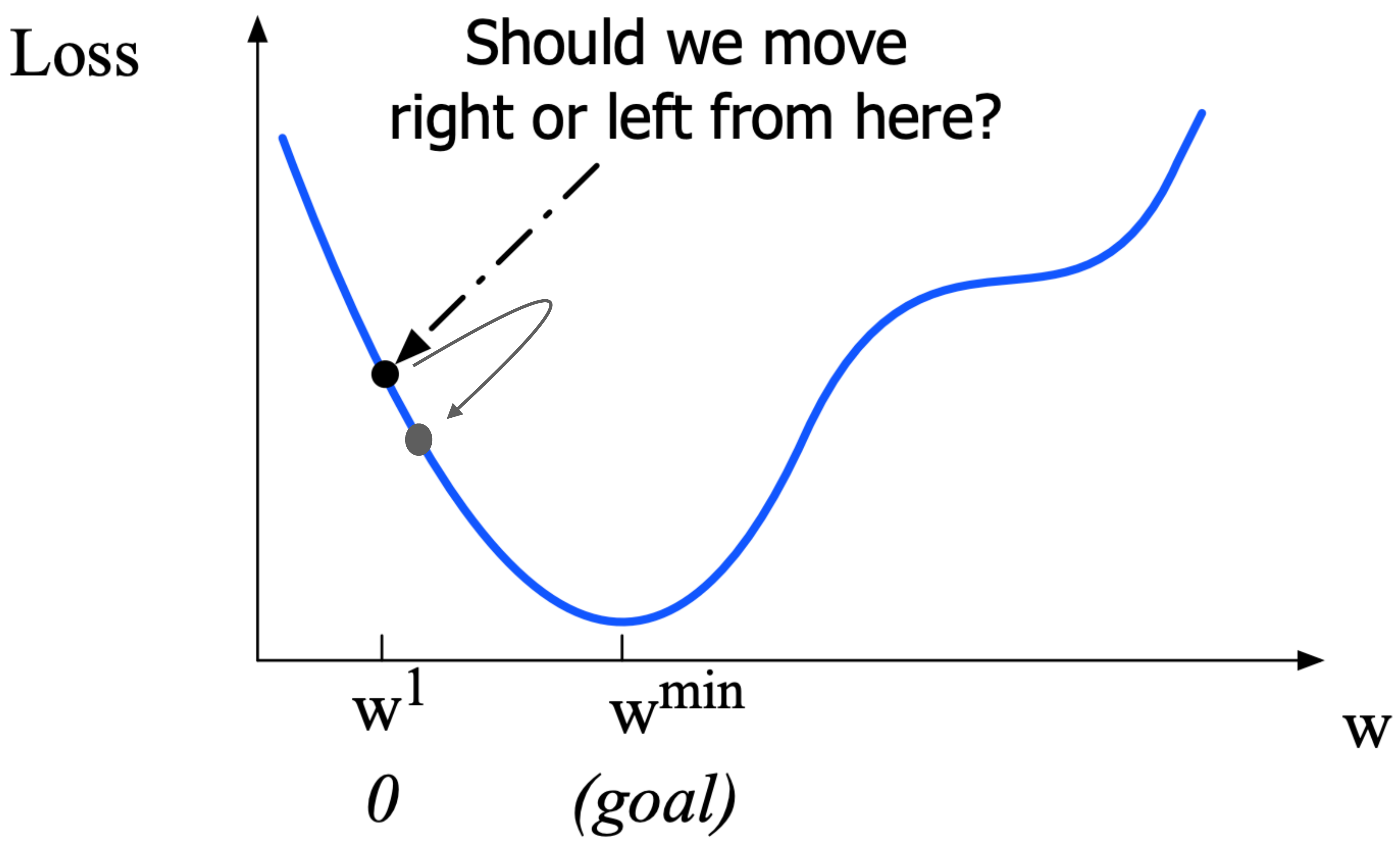
Move w in the reverse direction from the slope of the function



slope of loss at w^1 is negative

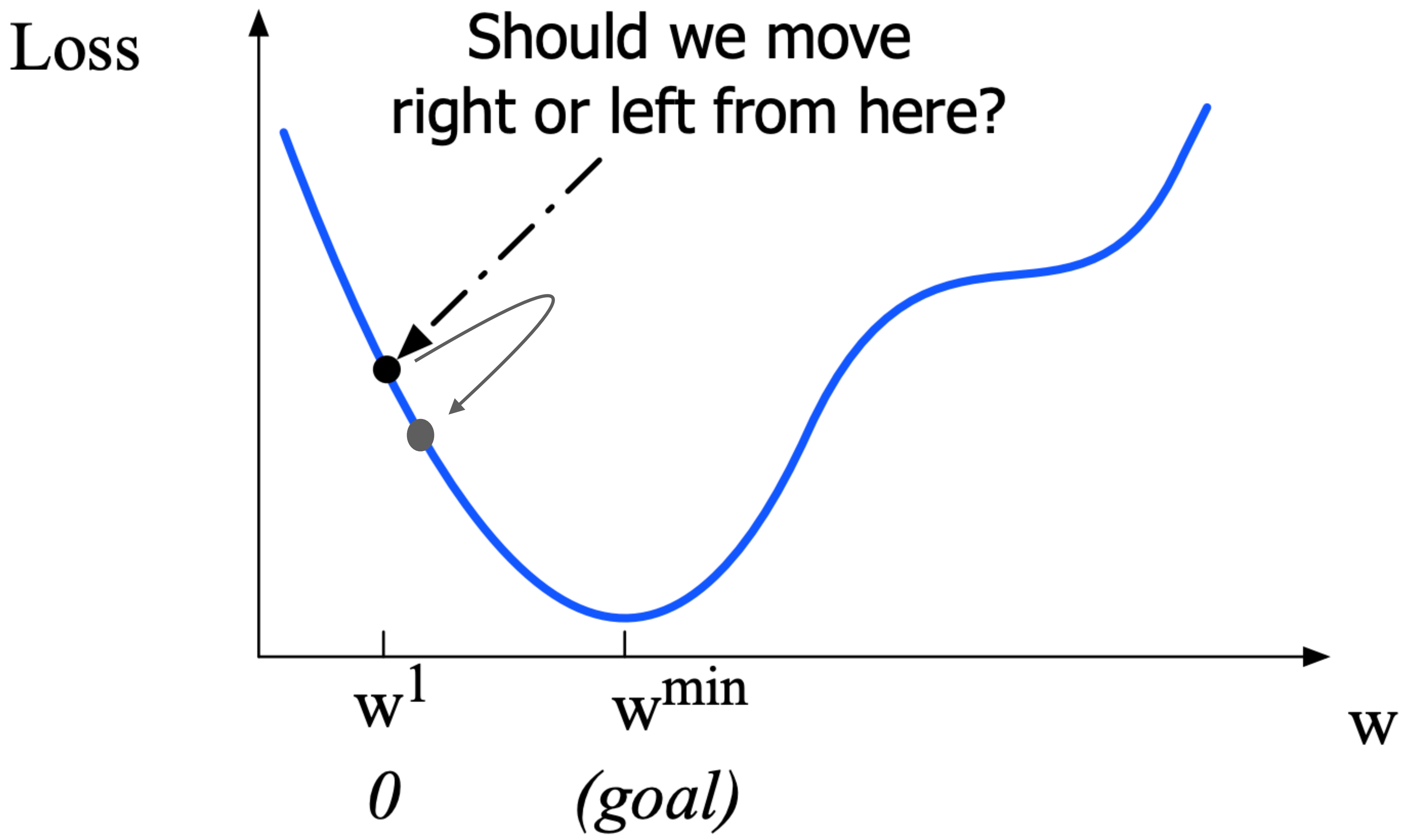
need to move positive

Gradients



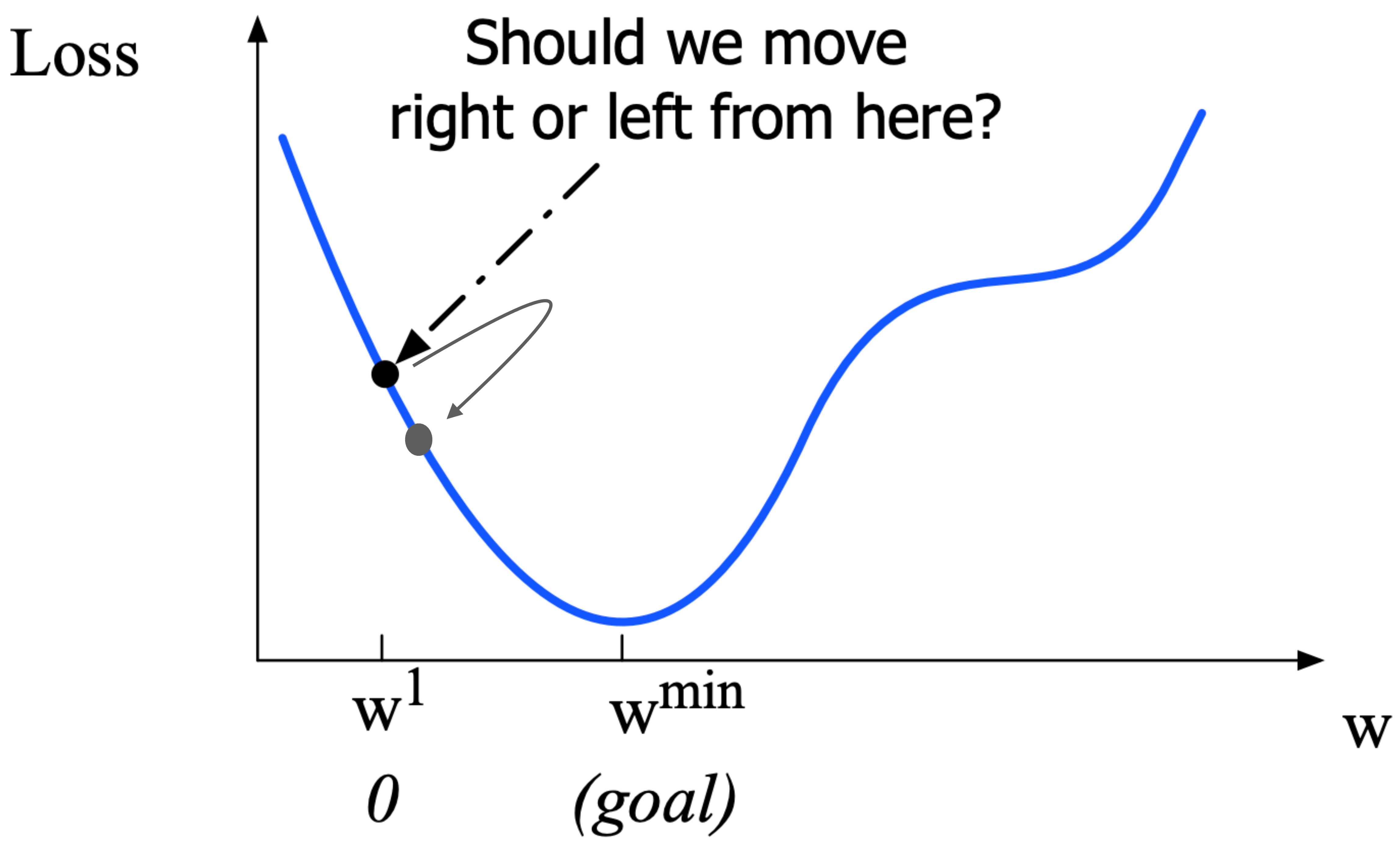
Gradients

The **gradient** of a function of many variables is a vector pointing in the direction of the greatest increase in a function.



Gradients

The **gradient** of a function of many variables is a vector pointing in the direction of the greatest increase in a function.

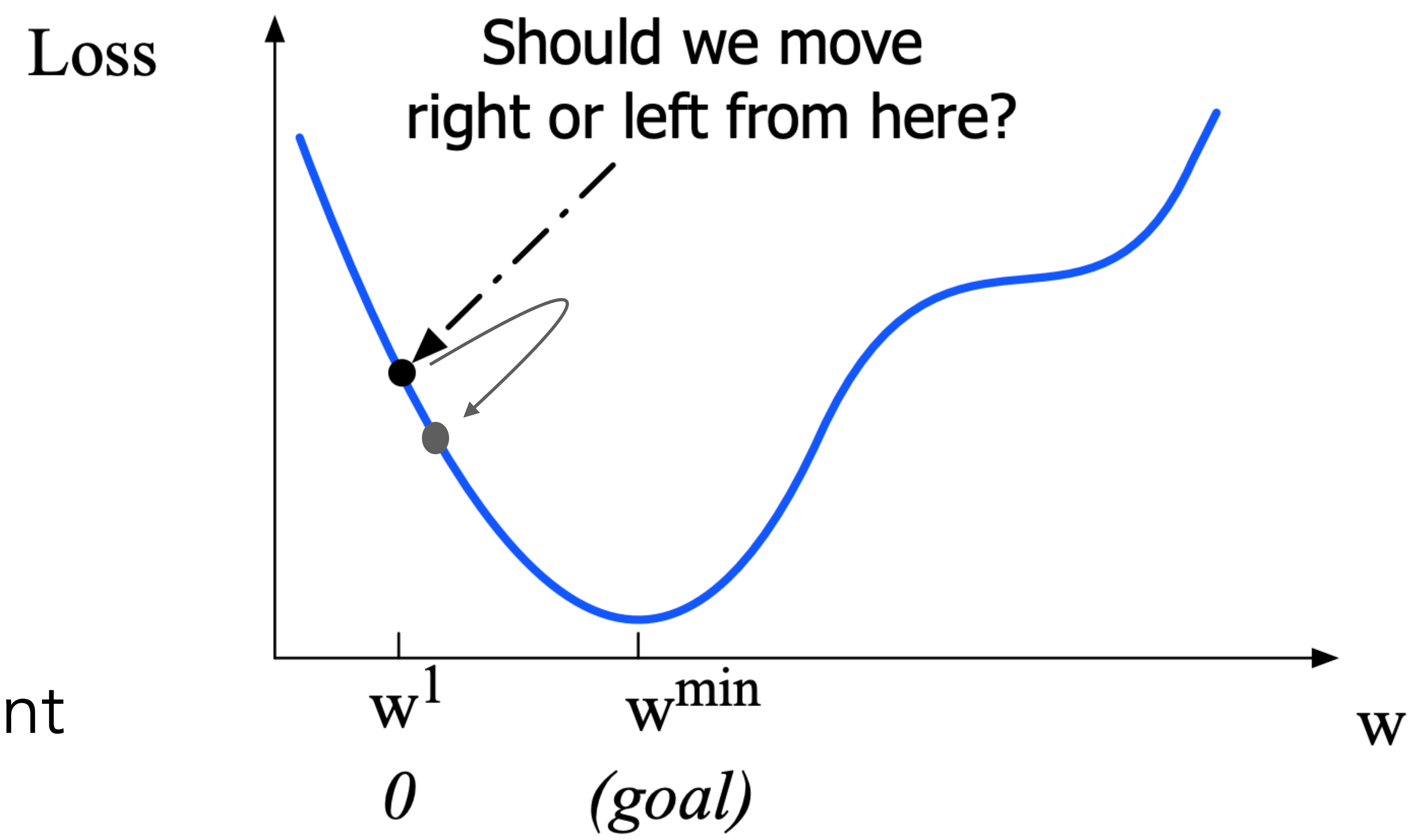


Gradient Descent

Gradients

The **gradient** of a function of many variables is a vector pointing in the direction of the greatest increase in a function.

Find the gradient of the loss function at the current point and move in the **opposite** direction.

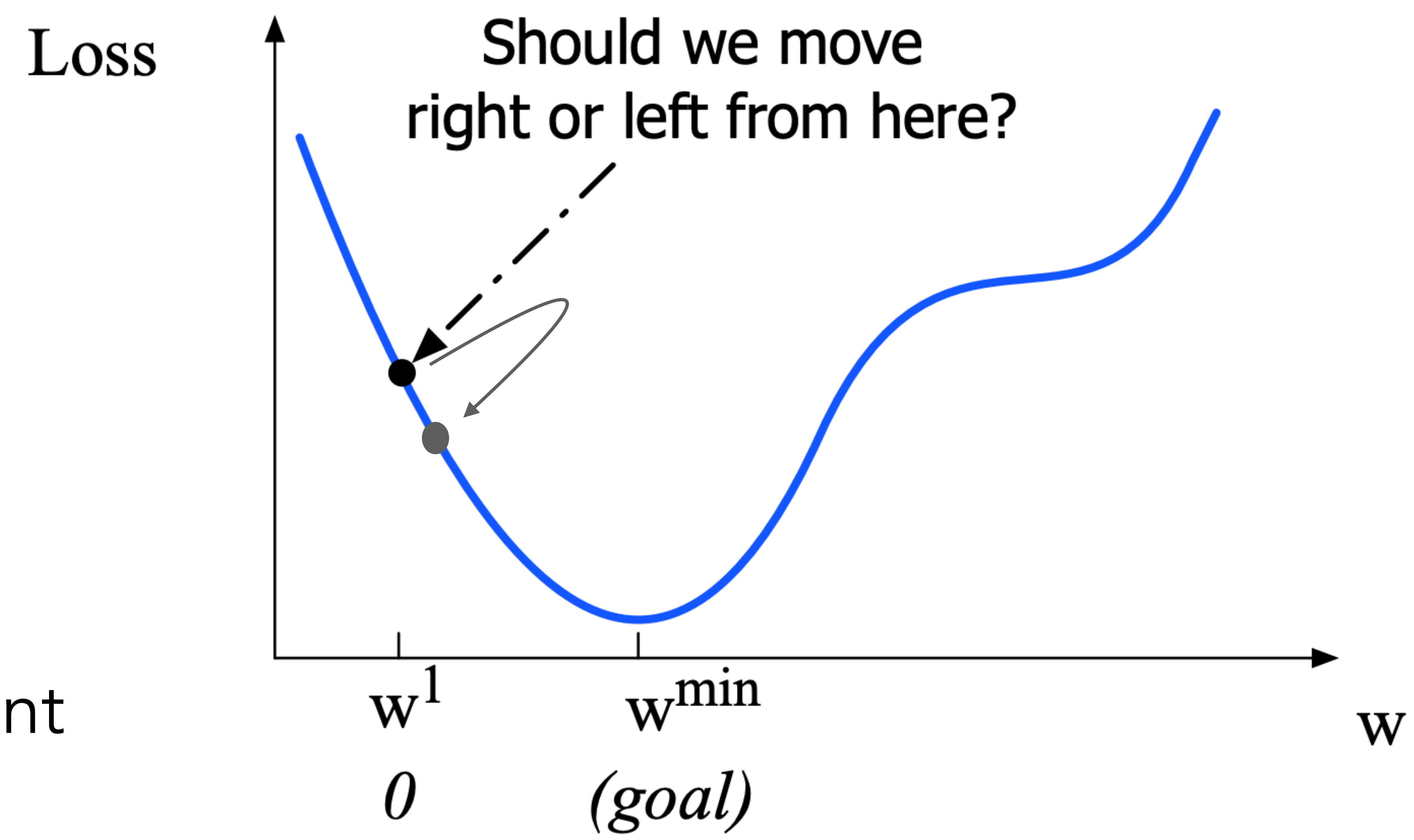


Gradient Descent

Gradients

The **gradient** of a function of many variables is a vector pointing in the direction of the greatest increase in a function.

Find the gradient of the loss function at the current point and move in the **opposite** direction.



Gradient Descent

But by how much?

Gradient Updates

Gradient Updates

- Move w by the value of the gradient $\frac{\partial}{\partial w} L(f(x; w), y^*)$, weighted by a learning rate η

Gradient Updates

- Move w by the value of the gradient $\frac{\partial}{\partial w} L(f(x; w), y^*)$, weighted by a learning rate η

$$w_{t+1} = w_t - \eta \frac{\partial}{\partial w} L(f(x; w), y^*)$$

Gradient Updates

- Move w by the value of the gradient $\frac{\partial}{\partial w}L(f(x; w), y^*)$, weighted by a learning rate η
- Higher learning rate means move w faster

$$w_{t+1} = w_t - \eta \frac{\partial}{\partial w}L(f(x; w), y^*)$$

Gradient Updates

- Move w by the value of the gradient $\frac{\partial}{\partial w} L(f(x; w), y^*)$, weighted by a learning rate η
- Higher learning rate means move w faster

η **Too high: the learner will take big steps and overshoot**

$$w_{t+1} = w_t - \eta \frac{\partial}{\partial w} L(f(x; w), y^*)$$

Gradient Updates

- Move w by the value of the gradient $\frac{\partial}{\partial w}L(f(x;w),y^*)$, weighted by a learning rate η
- Higher learning rate means move w faster

η **Too high: the learner will take big steps and overshoot**

$$w_{t+1} = w_t - \eta\frac{\partial}{\partial w}L(f(x;w),y^*)$$

η **Too low: the learner will take too long**

Gradient Updates

- Move w by the value of the gradient $\frac{\partial}{\partial w}L(f(x;w),y^*)$, weighted by a learning rate η
- Higher learning rate means move w faster

η **Too high: the learner will take big steps and overshoot**

$$w_{t+1} = w_t - \eta \frac{\partial}{\partial w} L(f(x;w),y^*)$$

η **Too low: the learner will take too long**

If parameter θ is a vector of d dimensions:

The gradient is just such a vector; it expresses the directional components of the sharpest slope along each of the d dimensions.

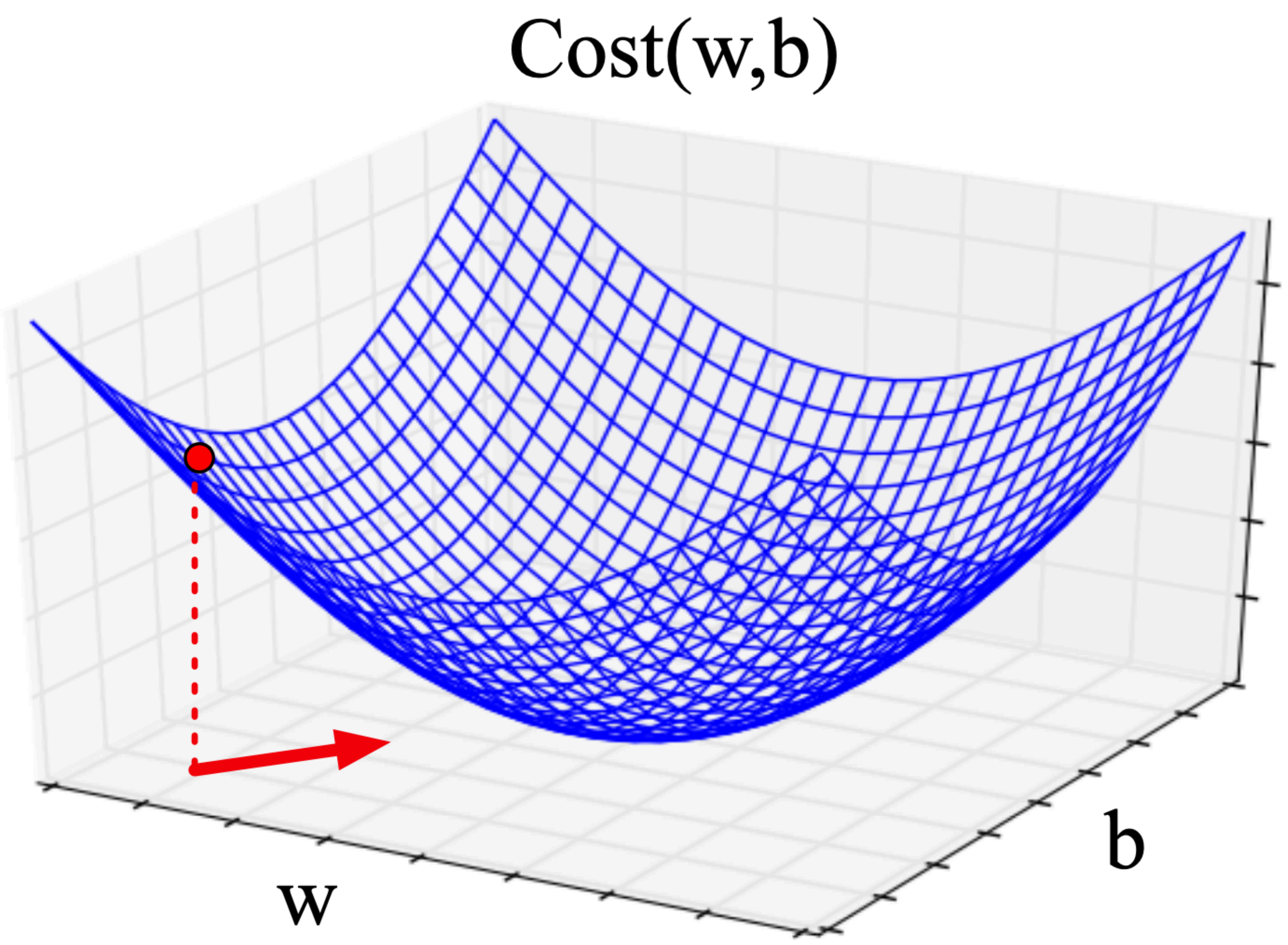
Under 2 dimensions

Under 2 dimensions

Consider 2 dimensions, w and b:

Under 2 dimensions

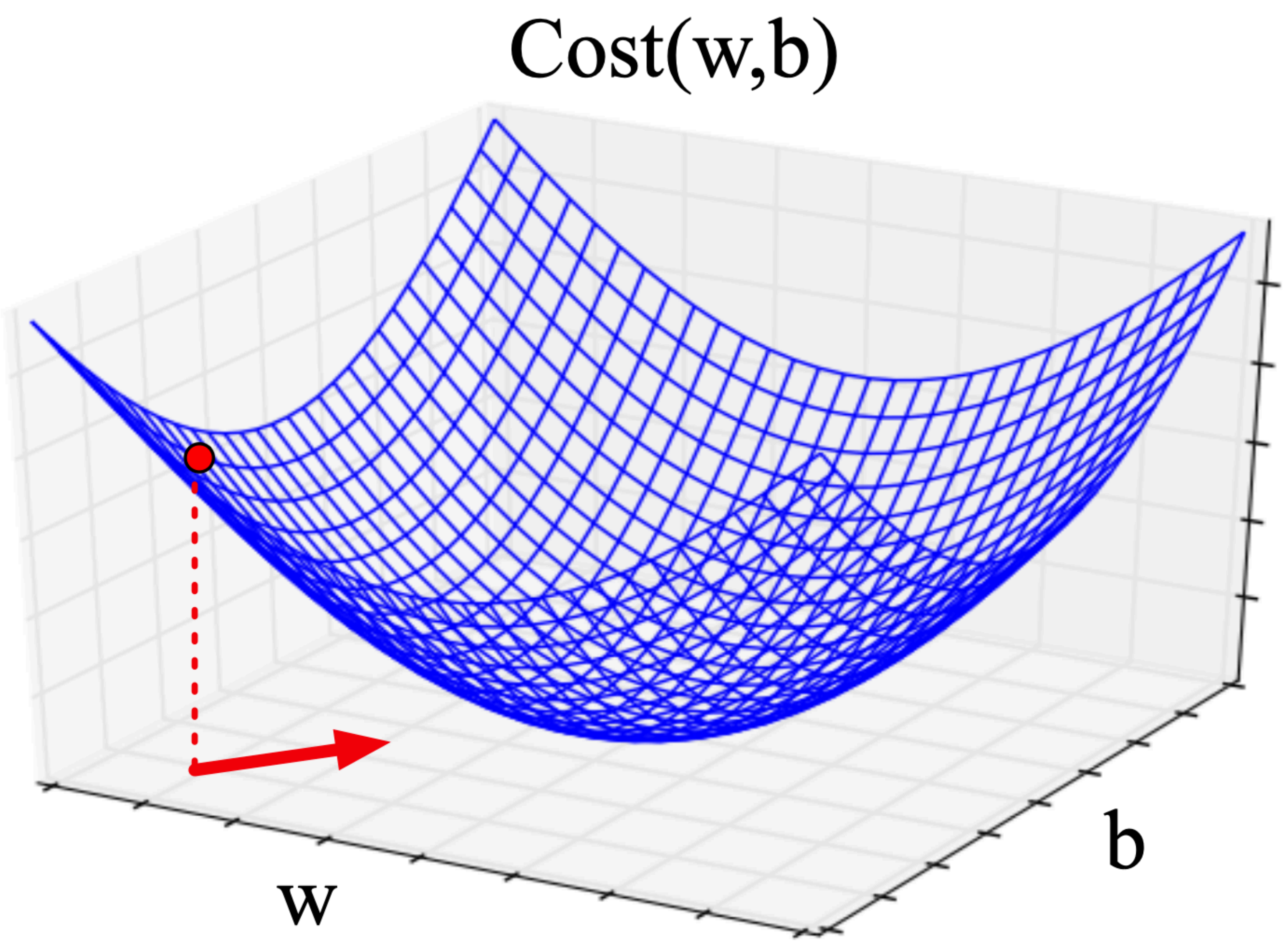
Consider 2 dimensions, w and b:



Under 2 dimensions

Consider 2 dimensions, w and b:

Visualizing the gradient vector at the red point

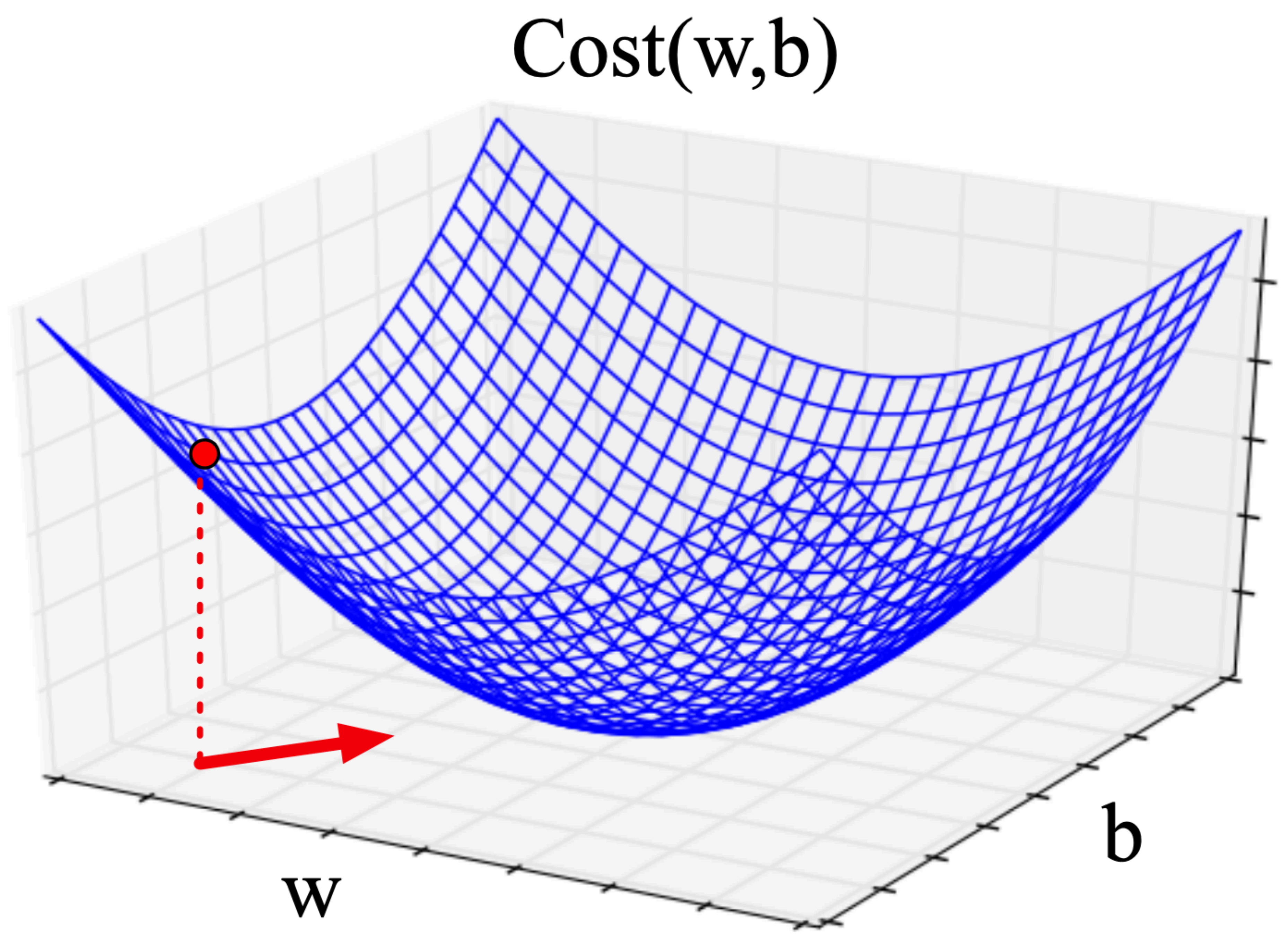


Under 2 dimensions

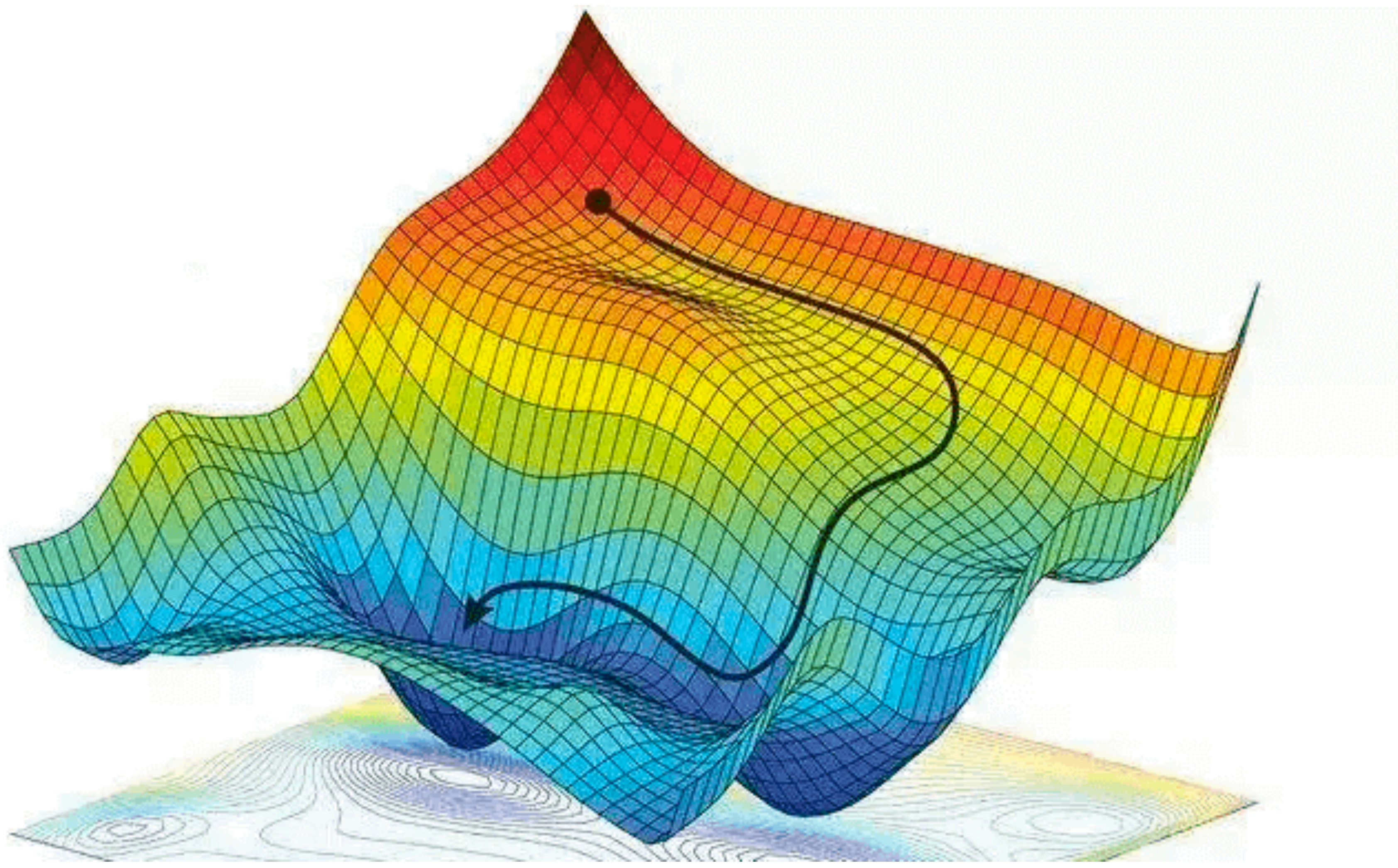
Consider 2 dimensions, w and b:

Visualizing the gradient vector at the red point

It has two dimensions shown in the $x - y$ plane

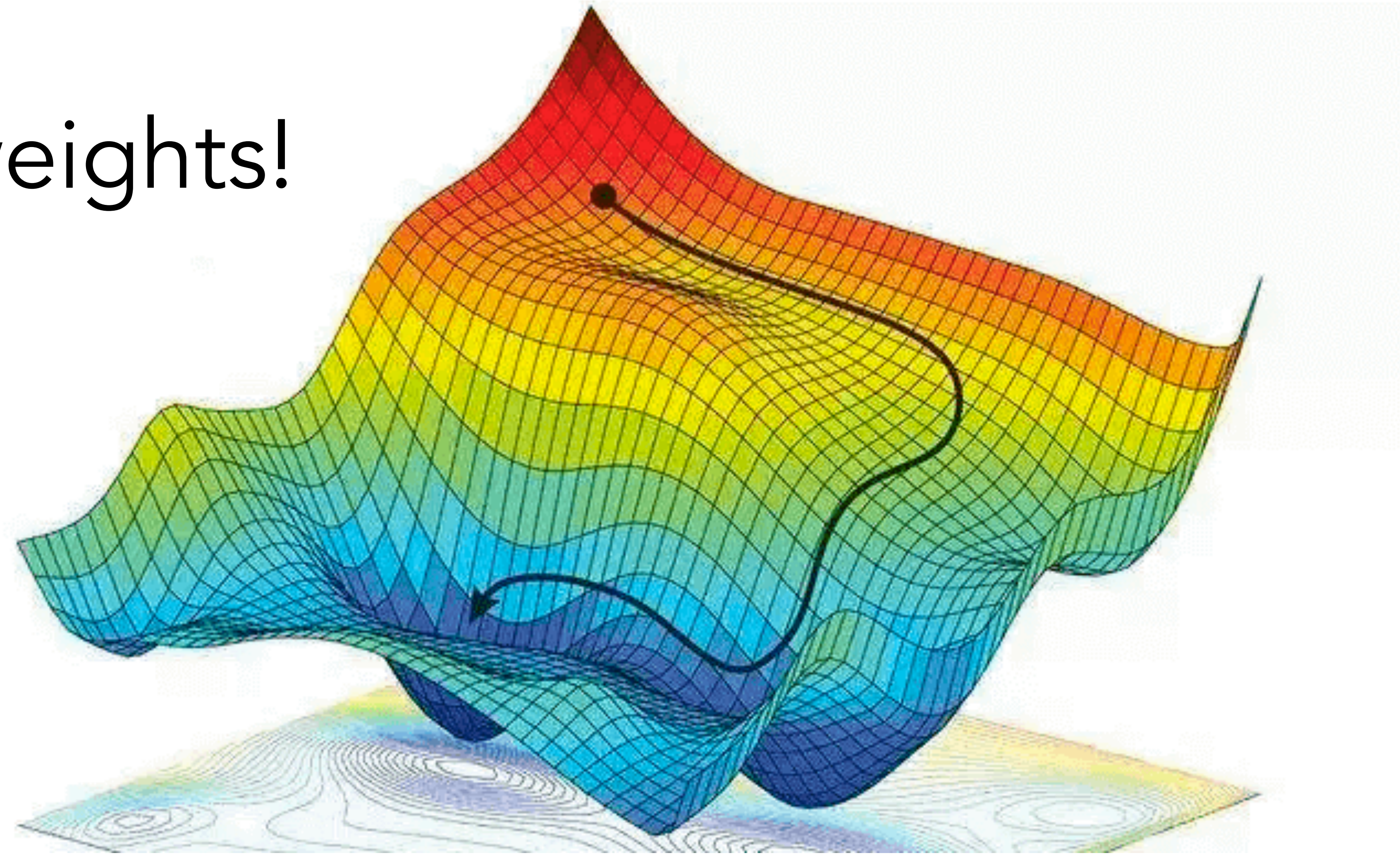


Real-life gradients, however…



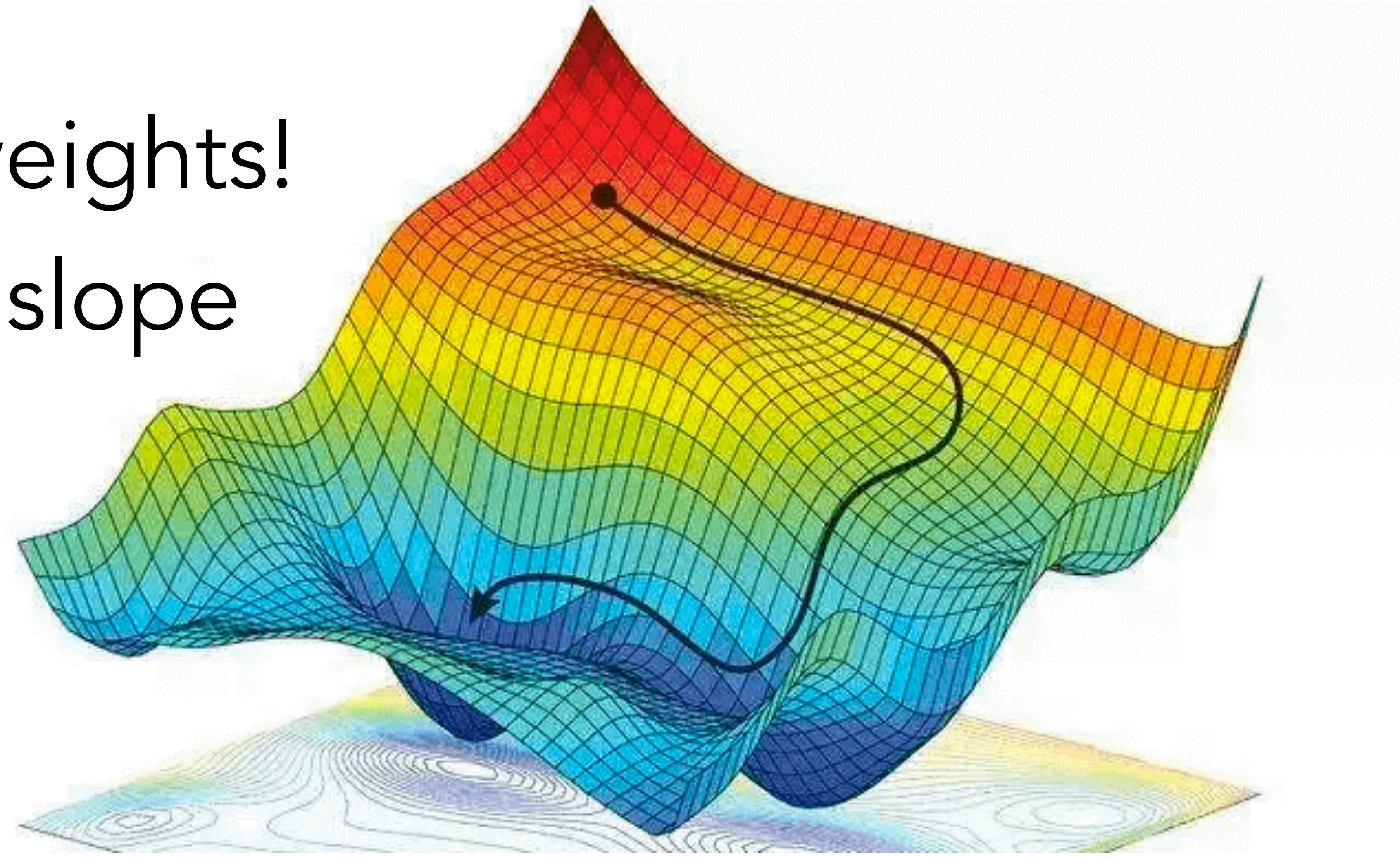
Real-life gradients, however…

- …are much longer; models usually contain lots and lots of weights!



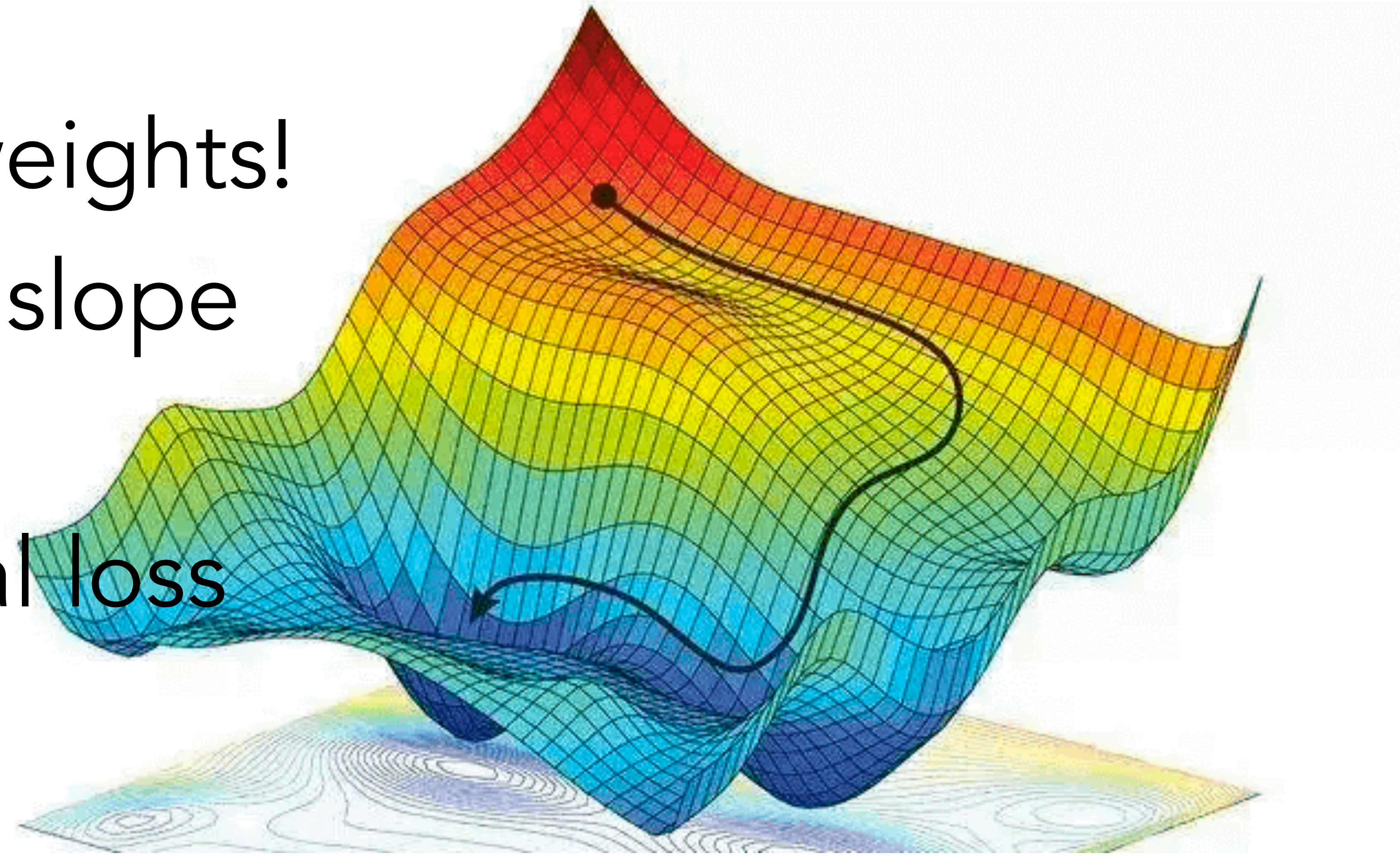
Real-life gradients, however…

- …are much longer; models usually contain lots and lots of weights!
- For each dimension θ_i the gradient component i tells us the slope with respect to that variable

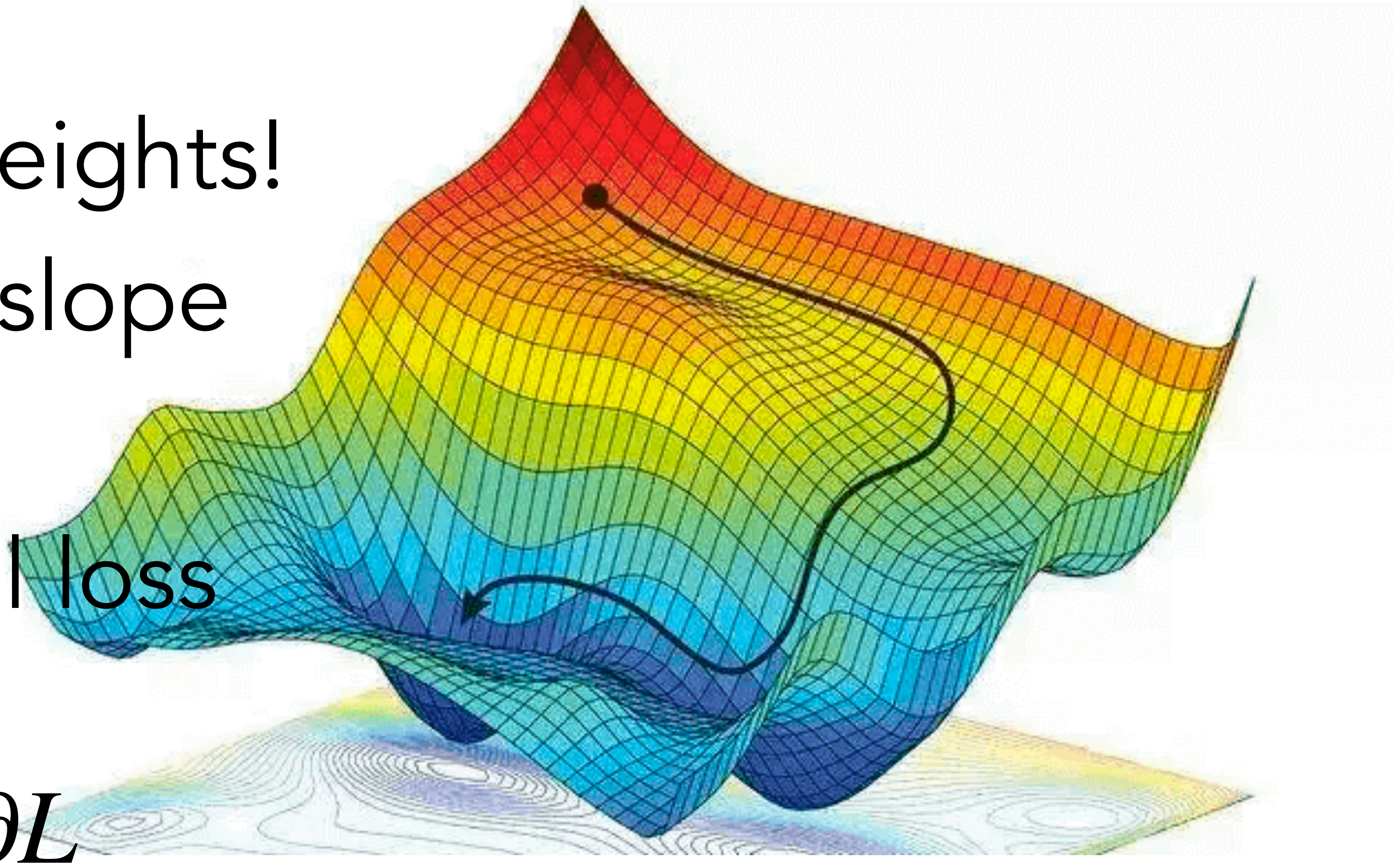


Real-life gradients, however…

- …are much longer; models usually contain lots and lots of weights!
- For each dimension θ_i the gradient component i tells us the slope with respect to that variable
  - "How much would a small change in θ_i influence the total loss function L?"

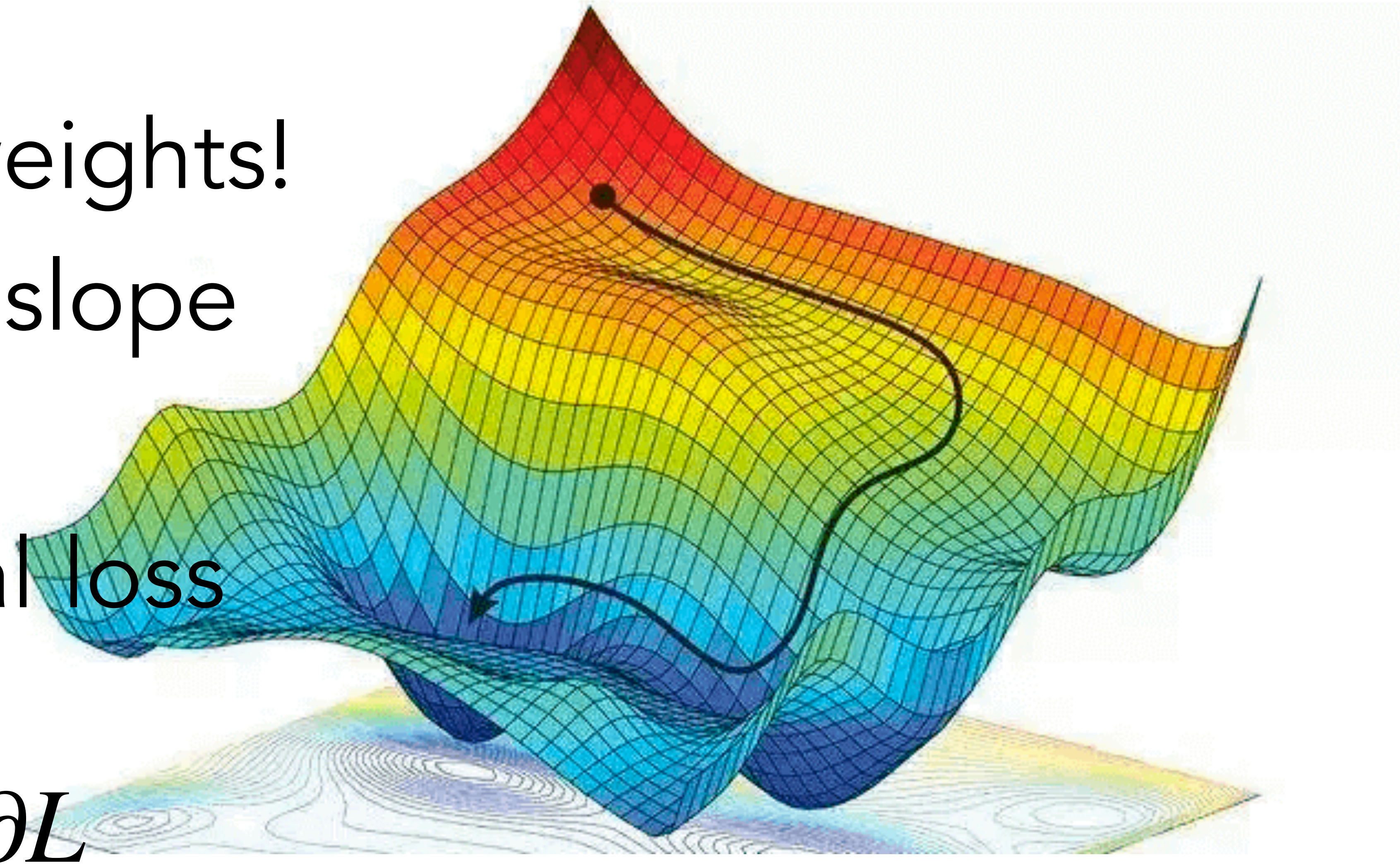


Real-life gradients, however…



- …are much longer; models usually contain lots and lots of weights!
- For each dimension θ_i the gradient component i tells us the slope with respect to that variable
  - "How much would a small change in θ_i influence the total loss function L?"
- We express the slope as a partial derivative $\frac{\partial}{\partial \theta_i}$ of the loss, $\frac{\partial L}{\partial \theta_i}$

Real-life gradients, however…



- …are much longer; models usually contain lots and lots of weights!
- For each dimension θ_i the gradient component i tells us the slope with respect to that variable
  - "How much would a small change in θ_i influence the total loss function L?"
- We express the slope as a partial derivative $\frac{\partial}{\partial \theta_i}$ of the loss, $\frac{\partial L}{\partial \theta_i}$
  - The gradient is then defined as a vector of these partials

Real-life gradients

We will represent $\hat{y}$ as $f(x;\theta)$ to make the dependence on θ more obvious

$$\nabla_\theta L(f(x;\theta),y)) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\theta),y) \\ \frac{\partial}{\partial w_2} L(f(x;\theta),y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\theta),y) \end{bmatrix}$$

Real-life gradients

We will represent $\hat{y}$ as $f(x;\theta)$ to make the dependence on θ more obvious

$$\nabla_{\theta} L(f(x;\theta), y)) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\theta), y) \\ \frac{\partial}{\partial w_2} L(f(x;\theta), y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\theta), y) \end{bmatrix}$$

The final equation for updating θ at time step $t+1$ based on the gradient is thus:

$$\theta_{t+1} = \theta_t - \eta \frac{\partial}{\partial \theta} L(f(x;\theta), y)$$

Gradients for Logistic Regression

Case: Sentiment Analysis

Recall: the cross-entropy loss for logistic regression

$$L_{CE}(y, \hat{y}) = -\left[y \log \sigma(\mathbf{w} \cdot \mathbf{x} + b) + (1 - y)\log(\sigma(-\mathbf{w} \cdot \mathbf{x} + b))\right]$$

Gradients for Logistic Regression

Case: Sentiment Analysis

Recall: the cross-entropy loss for logistic regression

$$L_{CE}(y, \hat{y}) = -\left[y \log \sigma(\mathbf{w} \cdot \mathbf{x} + b) + (1 - y)\log(\sigma(-\mathbf{w} \cdot \mathbf{x} + b))\right]$$

Derivatives have a closed form solution:

$$\frac{\partial L_{CE}(y, \hat{y})}{\partial w_j} = [\sigma(\mathbf{w} \cdot \mathbf{x} + b) - y]x_j$$

Pseudocode

Pseudocode

function STOCHASTIC GRADIENT DESCENT $(L(), f(), x, y)$ returns θ

Pseudocode

function STOCHASTIC GRADIENT DESCENT $(L(), f(), x, y)$ returns θ

where: L is the loss function

f is a function parameterized by θ

$\mathbf{x}$ is the set of training inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N)}$

y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots y^{(N)}$

Pseudocode

function STOCHASTIC GRADIENT DESCENT $(L(), f(), x, y)$ returns θ

  # where: L is the loss function

  # f is a function parameterized by θ

  # $\mathbf{x}$ is the set of training inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N)}$

  # y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots y^{(N)}$

$\theta \leftarrow 0$ (or randomly initialized)

Pseudocode

function STOCHASTIC GRADIENT DESCENT $(L(), f(), x, y)$ returns θ

  # where: L is the loss function

  # f is a function parameterized by θ

  # $\mathbf{x}$ is the set of training inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N)}$

  # y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots y^{(N)}$

$\theta \leftarrow 0$ (or randomly initialized)

repeat till done

Pseudocode

function STOCHASTIC GRADIENT DESCENT ($L(), f(), x, y$) returns θ

  # where: L is the loss function

  # f is a function parameterized by θ

  # $\mathbf{x}$ is the set of training inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N)}$

  # y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots y^{(N)}$

$\theta \leftarrow 0$ (or randomly initialized)

repeat till done

 for each training tuple$(x^{(i)}, y^{(i)})$: (in random order)

Pseudocode

function STOCHASTIC GRADIENT DESCENT $(L(), f(), x, y)$ returns θ

  # where: L is the loss function

  # f is a function parameterized by θ

  # $\mathbf{x}$ is the set of training inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N)}$

  # y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots y^{(N)}$

$\theta \leftarrow 0$ (or randomly initialized)

repeat till done

 for each training tuple$(x^{(i)}, y^{(i)})$: (in random order)

1. Compute $\hat{y}^{(i)} = f(\mathbf{x}^{(i)}; \theta)$ # What is our estimated output $\hat{y}^{(i)}$?

Pseudocode

function STOCHASTIC GRADIENT DESCENT $(L(), f(), x, y)$ returns θ

 # where: L is the loss function

 # f is a function parameterized by θ

 # $\mathbf{x}$ is the set of training inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N)}$

 # y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots y^{(N)}$

$\theta \leftarrow 0$ (or randomly initialized)

repeat till done

 for each training tuple$(x^{(i)}, y^{(i)})$: (in random order)

1. Compute $\hat{y}^{(i)} = f(\mathbf{x}^{(i)}; \theta)$ # What is our estimated output $\hat{y}^{(i)}$?
2. Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?

Pseudocode

function STOCHASTIC GRADIENT DESCENT $(L(), f(), x, y)$ returns θ

  # where: L is the loss function

  # f is a function parameterized by θ

  # $\mathbf{x}$ is the set of training inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N)}$

  # y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots y^{(N)}$

$\theta \leftarrow 0$ (or randomly initialized)

repeat till done

  for each training tuple$(x^{(i)}, y^{(i)})$: (in random order)

1. Compute $\hat{y}^{(i)} = f(\mathbf{x}^{(i)}; \theta)$ # What is our estimated output $\hat{y}^{(i)}$?
2. Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?
3. $g \leftarrow \nabla L(f(\mathbf{x}^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss?

Pseudocode

function STOCHASTIC GRADIENT DESCENT ($L(), f(), x, y$) returns θ

  # where: L is the loss function

  # f is a function parameterized by θ

  # $\mathbf{x}$ is the set of training inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N)}$

  # y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots y^{(N)}$

$\theta \leftarrow 0$ (or randomly initialized)

repeat till done

  for each training tuple$(x^{(i)}, y^{(i)})$: (in random order)

1. Compute $\hat{y}^{(i)} = f(\mathbf{x}^{(i)}; \theta)$ # What is our estimated output $\hat{y}^{(i)}$?
2. Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?
3. $g \leftarrow \nabla L(f(\mathbf{x}^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss?
4. $\theta \leftarrow \theta - \eta g$ # Go the other way instead

Pseudocode

function STOCHASTIC GRADIENT DESCENT $(L(), f(), x, y)$ returns θ

  # where: L is the loss function

  # f is a function parameterized by θ

  # $\mathbf{x}$ is the set of training inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N)}$

  # y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots y^{(N)}$

$\theta \leftarrow 0$ (or randomly initialized)

repeat till done

  for each training tuple$(x^{(i)}, y^{(i)})$: (in random order)

1. Compute $\hat{y}^{(i)} = f(\mathbf{x}^{(i)}; \theta)$ # What is our estimated output $\hat{y}^{(i)}$?
2. Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?
3. $g \leftarrow \nabla L(f(\mathbf{x}^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss?
4. $\theta \leftarrow \theta - \eta g$ # Go the other way instead

return θ

Pseudocode

function STOCHASTIC GRADIENT DESCENT ($L(), f(), x, y$) returns θ

  # where: L is the loss function

  # $\quad f$ is a function parameterized by θ

  # $\quad \mathbf{x}$ is the set of training inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N)}$

  # $\quad y$ is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots y^{(N)}$

$\theta \leftarrow 0$ (or randomly initialized)

repeat till done

  for each training tuple($x^{(i)}, y^{(i)}$): (in random order)

1. Compute $\hat{y}^{(i)} = f(\mathbf{x}^{(i)}; \theta)$ # What is our estimated output $\hat{y}^{(i)}$?
2. Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?
3. $g \leftarrow \nabla L(f(\mathbf{x}^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss?
4. $\theta \leftarrow \theta - \eta g$ # Go the other way instead

return θ

Stochastic Gradient Descent

Mini-Batching

function STOCHASTIC GRADIENT DESCENT $(L(), f(), x, y, m)$ returns θ

where: L is the loss function

f is a function parameterized by θ

$\mathbf{x}$ is the set of training inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N)}$

y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots y^{(N)}$ and m is the mini-batch size

$\theta \leftarrow 0$ (or randomly initialized)

repeat till done

for each randomly sampled minibatch of size m:

1. for each training tuple$(\mathbf{x}^{(i)}, y^{(i)})$ in the minibatch: (in random order)
   i. Compute $\hat{y}^{(i)} = f(\mathbf{x}^{(i)}; \theta)$ # What is our estimated output $\hat{y}^{(i)}$?
   ii. Compute the loss $L_{mini} \leftarrow L_{mini} + L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?
2. $g \leftarrow \frac{1}{m} \nabla L_{mini}(f(\mathbf{x}^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss?
3. $\theta \leftarrow \theta - \eta g$ # Go the other way instead

return θ

Mini-Batching

function STOCHASTIC GRADIENT DESCENT $(L(), f(), x, y, m)$ returns θ

where: L is the loss function

f is a function parameterized by θ

$\mathbf{x}$ is the set of training inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N)}$

y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots y^{(N)}$ and m is the mini-batch size

$\theta \leftarrow 0$ (or randomly initialized)

repeat till done

for each randomly sampled minibatch of size m:

1. for each training tuple$(\mathbf{x}^{(i)}, y^{(i)})$ in the minibatch: (in random order)
   i. Compute $\hat{y}^{(i)} = f(\mathbf{x}^{(i)}; \theta)$ # What is our estimated output $\hat{y}^{(i)}$?
   ii. Compute the loss $L_{mini} \leftarrow L_{mini} + L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?
2. $g \leftarrow \frac{1}{m} \nabla L_{mini}(f(\mathbf{x}^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss?
3. $\theta \leftarrow \theta - \eta g$ # Go the other way instead

return θ

Why is this better than stochastic gradient descent?

Lecture Outline

- Recap
  - Smoothing
  - Basics of Supervised Machine Learning
    - Data: Preprocessing and Feature Extraction
- Quiz
- Announcements
- Basics of Supervised Machine Learning
  I. Data: Preprocessing and Feature Extraction
  II. Model:
      I. Logistic Regression
  III. Loss
  IV. Optimization Algorithm
  V. Inference

Regularization

- A model that perfectly match the training data has a problem

- A model that perfectly match the training data has a problem

Why?

Overfitting

- A model that perfectly match the training data has a problem
- It will also overfit to the data, modeling noise

Why?

Overfitting

- A model that perfectly match the training data has a problem
- It will also overfit to the data, modeling noise
  - A random word that perfectly predicts y (it happens to only occur in one class) will get a very high weight

Why?

Overfitting

Why?

- A model that perfectly match the training data has a problem
- It will also overfit to the data, modeling noise
  - A random word that perfectly predicts y (it happens to only occur in one class) will get a very high weight
  - Failing to generalize to a test set without this word

Overfitting

- A model that perfectly match the training data has a problem
- It will also overfit to the data, modeling noise
  - A random word that perfectly predicts y (it happens to only occur in one class) will get a very high weight
  - Failing to generalize to a test set without this word

Why?

A good model should be able to generalize

Overfitting

- A model that perfectly match the training data has a problem
- It will also overfit to the data, modeling noise
  - A random word that perfectly predicts y (it happens to only occur in one class) will get a very high weight
  - Failing to generalize to a test set without this word

Why?

A good model should be able to generalize

What happens when a feature only occurs with one class?

Overfitting

Why?

- A model that perfectly match the training data has a problem
- It will also overfit to the data, modeling noise
  - A random word that perfectly predicts y (it happens to only occur in one class) will get a very high weight
  - Failing to generalize to a test set without this word

A good model should be able to generalize

What happens when a feature only occurs with one class?

e.g. word “wow” for positive reviews

USC Viterbi

Overfitting: Features

This movie drew me in, and it'll do the same to you.

I can't tell you how much I hated this movie. It sucked.

Overfitting: Features

This movie drew me in, and it'll do the same to you.

I can't tell you how much I hated this movie. It sucked.

Useful or harmless features

x_1 = **"this"**

x_2 = **"movie**

x_3 = **"hated"**

x_4 = **"drew me in"**

Overfitting: Features

This movie drew me in, and it'll do the same to you.

Useful or harmless features

x_1 = **"this"**
x_2 = **"movie**
x_3 = **"hated"**
x_4 = **"drew me in"**

I can't tell you how much I hated this movie. It sucked.

4-gram features that just "memorize" training set and might cause problems

x_5 = **"the same to you"**
x_6 = **"tell you how much"**

Overfitting

Overfitting

- 4-gram model on tiny data will just memorize the data

Overfitting

- 4-gram model on tiny data will just memorize the data
    - 100% accuracy on the training set

Overfitting

- 4-gram model on tiny data will just memorize the data
    - 100% accuracy on the training set
- But it will be surprised by the novel 4-grams in the test data

Overfitting

- 4-gram model on tiny data will just memorize the data
  - 100% accuracy on the training set
- But it will be surprised by the novel 4-grams in the test data
  - Low accuracy on test set

Overfitting

- 4-gram model on tiny data will just memorize the data
  - 100% accuracy on the training set
- But it will be surprised by the novel 4-grams in the test data
  - Low accuracy on test set
- Models that are too powerful can overfit the data

Overfitting

- 4-gram model on tiny data will just memorize the data
  - 100% accuracy on the training set
- But it will be surprised by the novel 4-grams in the test data
  - Low accuracy on test set
- Models that are too powerful can overfit the data
  - Fitting the details of the training data so exactly that the model doesn't generalize well to the test set

Overfitting

- 4-gram model on tiny data will just memorize the data
    - 100% accuracy on the training set
- But it will be surprised by the novel 4-grams in the test data
    - Low accuracy on test set
- Models that are too powerful can overfit the data
    - Fitting the details of the training data so exactly that the model doesn't generalize well to the test set

How to avoid overfitting?

Overfitting

- 4-gram model on tiny data will just memorize the data
  - 100% accuracy on the training set
- But it will be surprised by the novel 4-grams in the test data
  - Low accuracy on test set
- Models that are too powerful can overfit the data
  - Fitting the details of the training data so exactly that the model doesn't generalize well to the test set

How to avoid overfitting?

Regularization in logistic regression

Overfitting

- 4-gram model on tiny data will just memorize the data
  - 100% accuracy on the training set
- But it will be surprised by the novel 4-grams in the test data
  - Low accuracy on test set
- Models that are too powerful can overfit the data
  - Fitting the details of the training data so exactly that the model doesn't generalize well to the test set

How to avoid overfitting?

Regularization in logistic regression

Dropout in neural networks

Regularization

Regularization

- A solution for overfitting: Add a regularization term $R(\theta)$ to the loss function

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y^{(i)} \,|\, \mathbf{x}^{(i)}) - \alpha R(\theta)$$

Regularization

- A solution for overfitting: Add a regularization term $R(\theta)$ to the loss function
  - (for now written as maximizing logprob rather than minimizing loss)

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y^{(i)} \mid \mathbf{x}^{(i)}) - \alpha R(\theta)$$

Regularization

- A solution for overfitting: Add a regularization term $R(\theta)$ to the loss function
  - (for now written as maximizing logprob rather than minimizing loss)

- Idea: choose an $R(\theta)$ that penalizes large weights
  - fitting the data well with lots of big weights not as good as
  - fitting the data a little less well, with small weights

$$\hat{\theta} = \underset{\theta}{\arg\max} \sum_{i=1}^{n} \log P(y^{(i)} | \mathbf{x}^{(i)}) - \alpha R(\theta)$$

L2 / Ridge Regularization

L2 / Ridge Regularization

- The sum of the squares of the weights

$$R(\theta) = \|\theta\|_2^2 = \sum_{j=1}^{d} \theta_j^2$$

L2 / Ridge Regularization

- The sum of the squares of the weights
- The name is because this is the (square of the) L2 norm $||\theta||_2^2$, = Euclidean distance of θ to the origin.

$$R(\theta) = ||\theta||_2^2 = \sum_{j=1}^{d} \theta_j^2$$

L2 / Ridge Regularization

- The sum of the squares of the weights
- The name is because this is the (square of the) L2 norm $||\theta||_2^2$, = Euclidean distance of θ to the origin.

$$R(\theta) = ||\theta||_2^2 = \sum_{j=1}^{d} \theta_j^2$$

L2 regularized objective function:

$$\hat{\theta} = \underset{\theta}{\arg\max} \sum_{i=1}^{n} \log P(y^{(i)} \,|\, x^{(i)}) - \alpha \sum_{j=1}^{d} \theta_j^2$$

L1 / Lasso Regularization

L1 / Lasso Regularization

- The sum of the (absolute value of the) weights

L1 / Lasso Regularization

- The sum of the (absolute value of the) weights

$$R(\theta) = \|\theta\|_1 = \sum_{j=1}^{d} |\theta_j|$$

L1 / Lasso Regularization

- The sum of the (absolute value of the) weights
- Named after the L1 norm $||\theta||_1$ = sum of the absolute values of the weights = Manhattan distance

$$R(\theta) = ||\theta||_1 = \sum_{j=1}^{d} |\theta_j|$$

L1 / Lasso Regularization

- The sum of the (absolute value of the) weights
- Named after the L1 norm $||\theta||_1$ = sum of the absolute values of the weights = Manhattan distance

$$R(\theta) = ||\theta||_1 = \sum_{j=1}^{d} |\theta_j|$$

L1 regularized objective function:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y^{(i)} | x^{(i)}) - \alpha \sum_{j=1}^{d} |\theta_j|$$