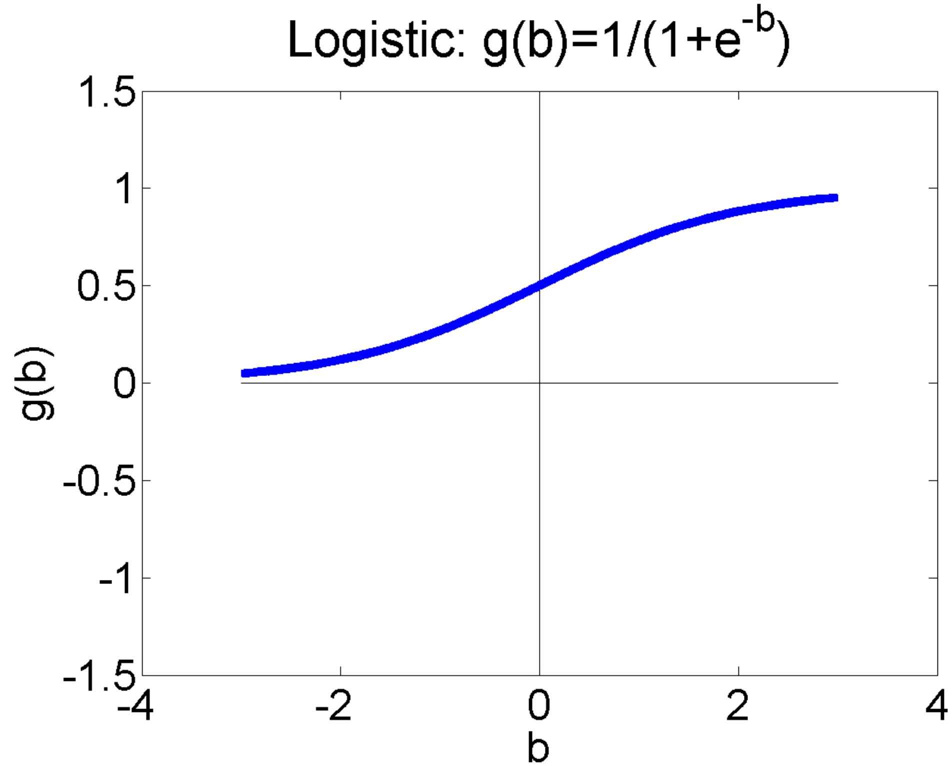
CS440/ECE448 Lecture 09: Logistic Regression

Mark Hasegawa-Johnson, 2/2021



Outline

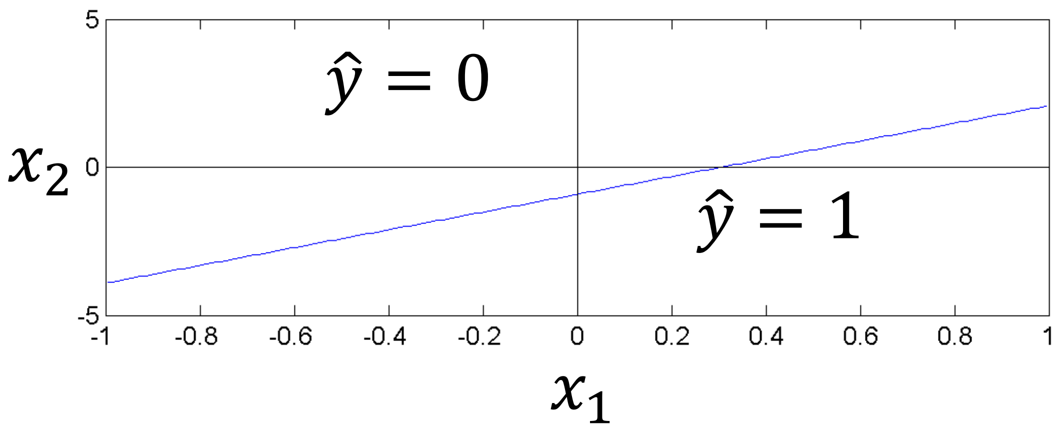
- Advantages and disadvantages of the perceptron
- Probabilistic-boundary classifiers
- How do you maximize a function?
- Learning a logistic regression
- Two-class logistic regression

Linear Classifiers in General

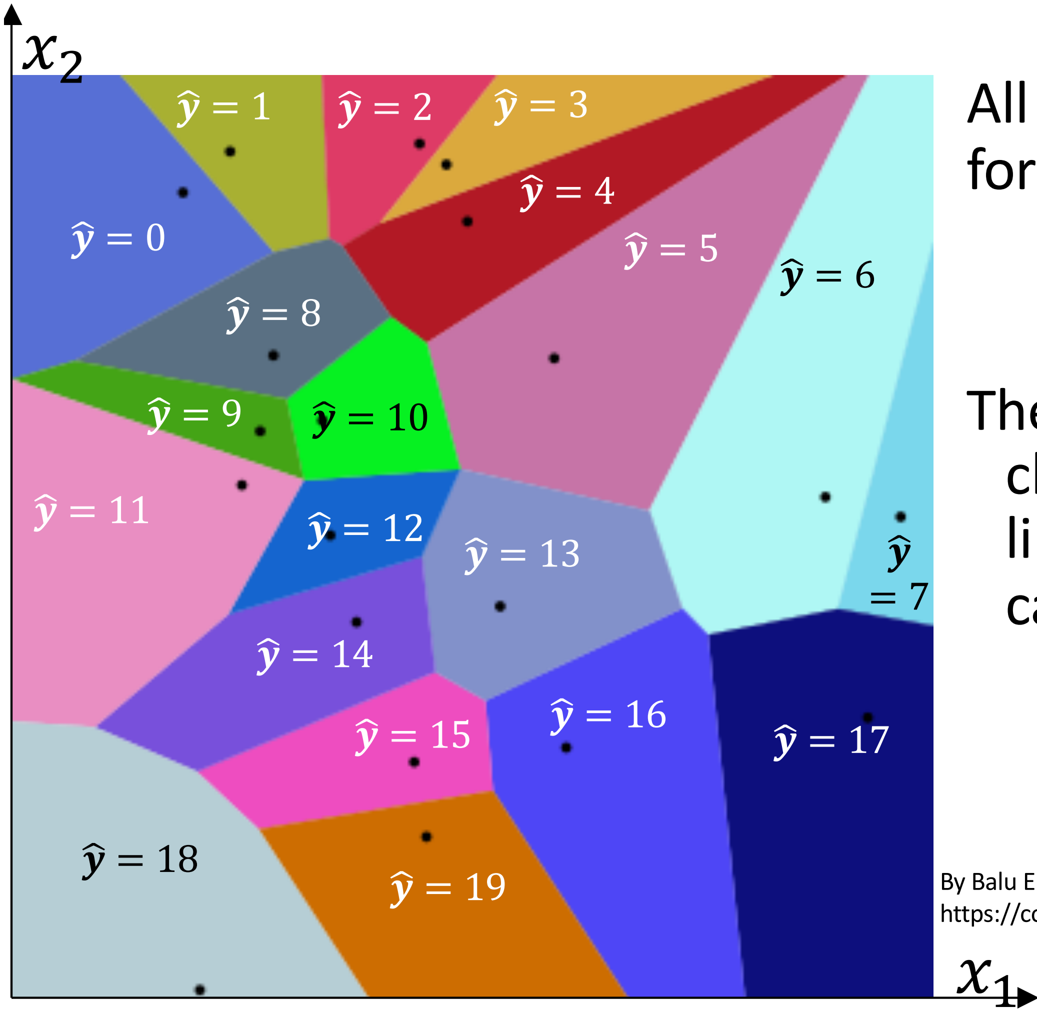
Consider the classifier

$$\hat{y} = u\left(b + \sum_{j=1}^{D} w_j x_j\right)$$

This is called a "linear classifier" because the boundary between the two classes is a line.



Multi-Class Linear Classifiers



All multi-class linear classifiers have the form

$$\hat{y} = \text{argmax}_{c=0}^{V-1}(w_c^T x)$$

The region of x-space associated with each class label is convex with piece-wise linear boundaries. Such regions are called "Voronoi regions."

Training a Multi-Class Perceptron

For each training instance $\boldsymbol{x}$ w/ground truth label $y \in \{0,1, \ldots, V-1\}$:

- Classify with current weights: $\hat{y} = \text{argmax}_{c=0}^{V-1}\left(w_c^T x\right)$
- Update weights:
  - if $\hat{y}$ is correct ($y = \hat{y}$) then do nothing
  - If $\hat{y}$ is incorrect ($y \neq \hat{y}$) then:
    - Update the correct-class vector as $w_y = w_y + \eta x$
    - Update the wrong-class vector as $w_{\hat{y}} = w_{\hat{y}} - \eta x$
    - Don't change the vectors of any other class

Multi-class perceptron: advantages and disadvantages

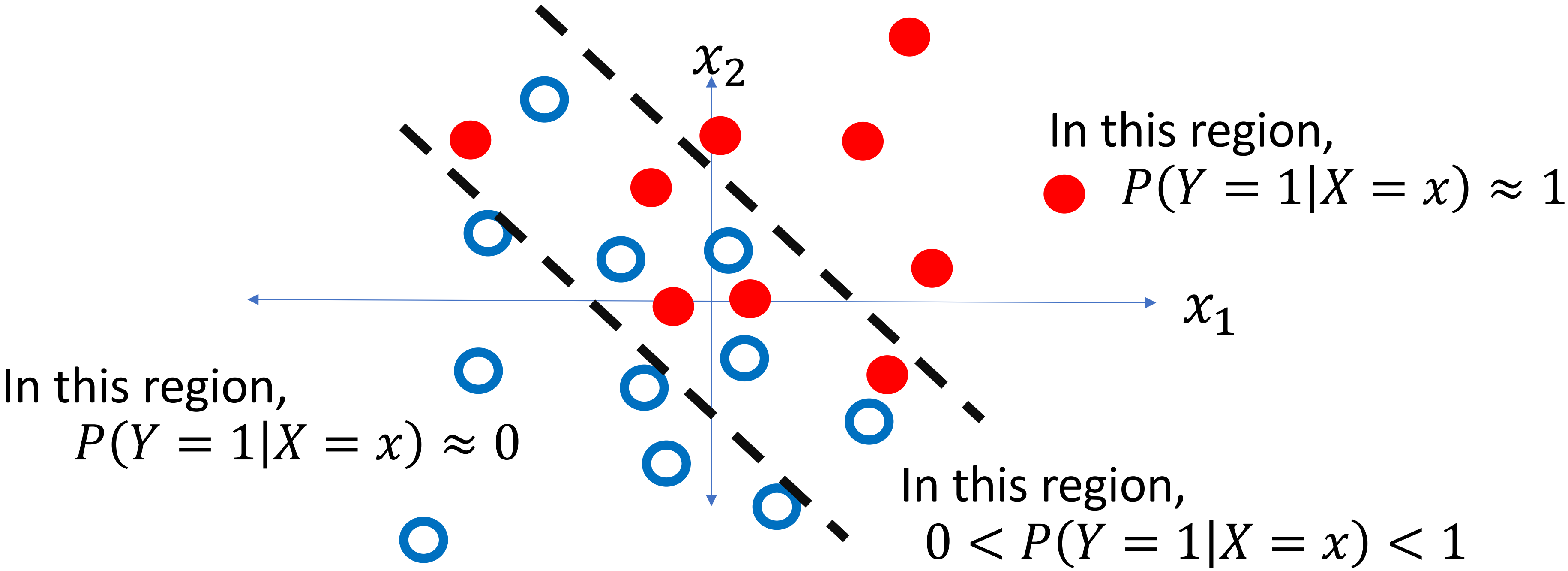
- ADVANTAGE: If the classes are linearly separable, then multi-class perceptron algorithm will find a set of linear functions that separate them
- DISADVANTAGE: If the classes are not linearly separable, then the w_c converge only if we force η to decay to zero ($\eta = \frac{1}{n}$ for the n^{th} training token). After they've converged, we don't know exactly how good or how bad the resulting w_c are.

Outline

- Advantages and disadvantages of the perceptron
- Probabilistic-boundary classifiers
- How do you maximize a function?
- Learning a logistic regression
- Two-class logistic regression

Probabilistic boundaries

Instead of trying to find the exact boundaries, logistic regression models the probability that token x belongs to class y.



Logistic regression and the softmax function

- Perceptron: $\hat{y} = \mathrm{argmax}_{c=0}^{V-1}\left(w_c^T x\right)$
- **Logistic regression:** $\mathrm{P}(\mathrm{Y} = \mathrm{c}|X = x) = \mathrm{softmax}_{c=0}^{V-1}\left(w_c^T x\right)$

where the "softmax" function is defined as

$$\mathrm{softmax}_{c=0}^{V-1}(w_c^T x) = \frac{e^{w_c^T x}}{\sum_{k=0}^{V-1} e^{w_k^T x}}$$

Logistic regression and the softmax function

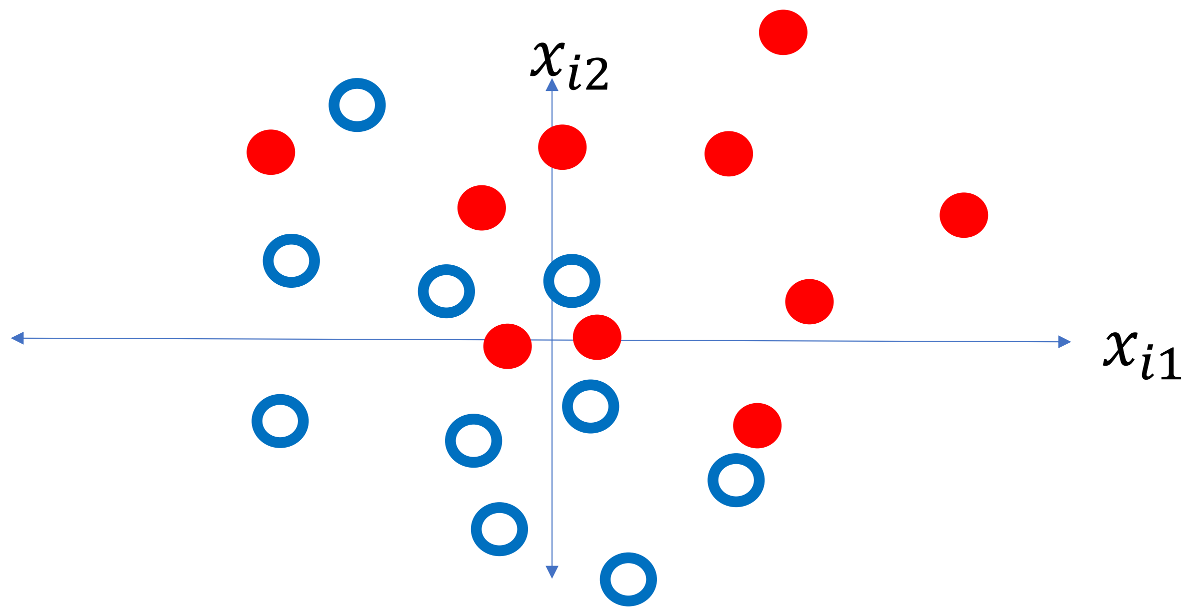
$$\mathrm{P}(\mathrm{Y} = \mathrm{c}|X = x) = \mathrm{softmax}_{c=0}^{V-1}(w_c^T x) = \frac{e^{w_c^T x}}{\sum_{k=0}^{V-1} e^{w_k^T x}}$$

- The exponential function ($e^{w_c^T x}$, sometimes written as $\exp(w_c^T x)$) guarantees that $\mathrm{P}(\mathrm{Y} = \mathrm{c}|X = x)$ is a positive number.
- The sum, in the denominator, guarantees that

$$1 = \sum_{c=0}^{V-1} \mathrm{P}(\mathrm{Y} = \mathrm{c}|X = x)$$

Learning logistic regression

- Suppose we have some data.
- We want to learn vectors $w_c = [w_{c1}, \dots, w_{cD}]^T$ so that $\mathrm{P}(\mathrm{Y} = \mathrm{c}|X = x) = \mathrm{softmax}_{c=0}^{1}(w_c^T x)$.

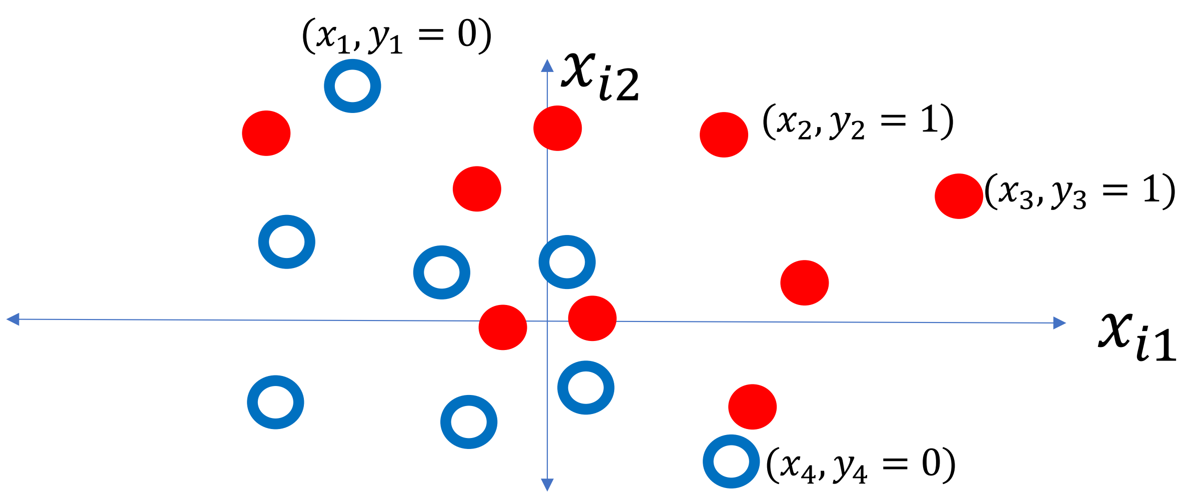


Learning logistic regression: Training data

Data:

$$\mathfrak{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

where each $x_i = [x_{i1}, \dots, x_{iD}]^T$ is a vector, and each y_i is an integer class label, $0 \le y_i \le V - 1$.



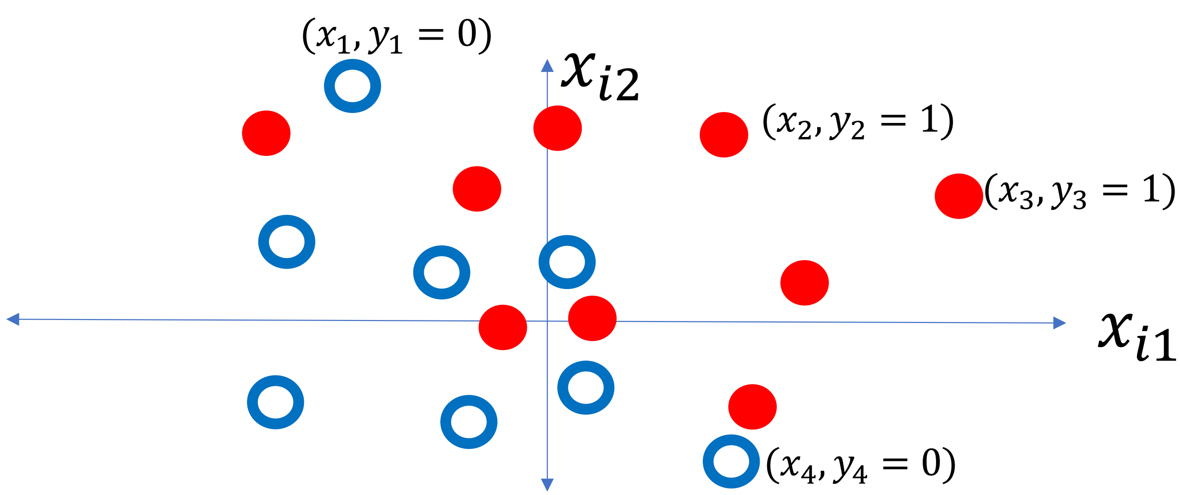
Learning logistic regression: Model parameters

We want to learn the model parameters

$$\theta = \{w_0, \ldots, w_{V-1}\}$$

so that

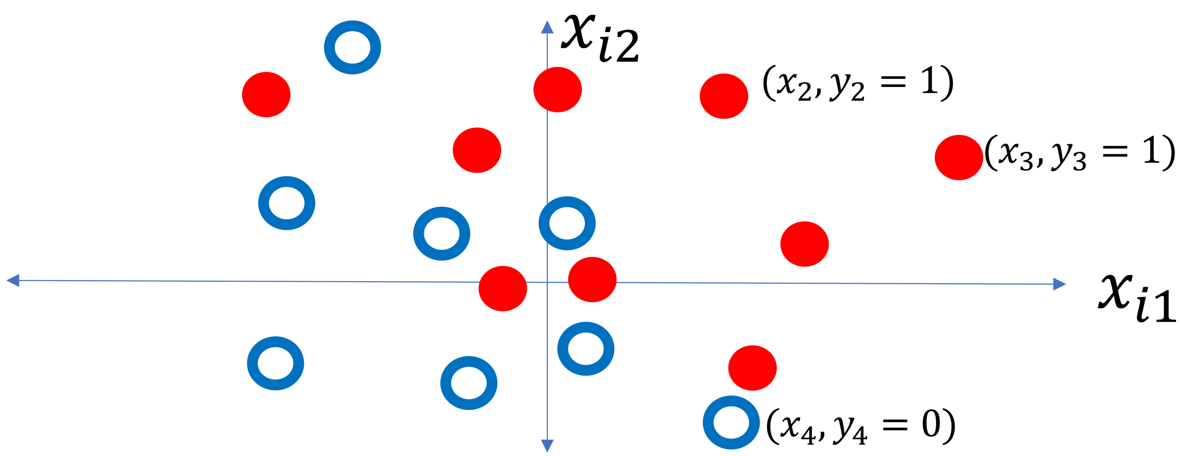
$$\mathrm{P}(\mathrm{Y} = y_i | X = x_i) = \mathrm{softmax}\left(w_{y_i}^T x_i\right)$$



Learning logistic regression: Training criterion

We want to learn the model parameters, $\theta = \{w_0, \ldots, w_{V-1}\}$, in order to maximize the probability of the observed data:

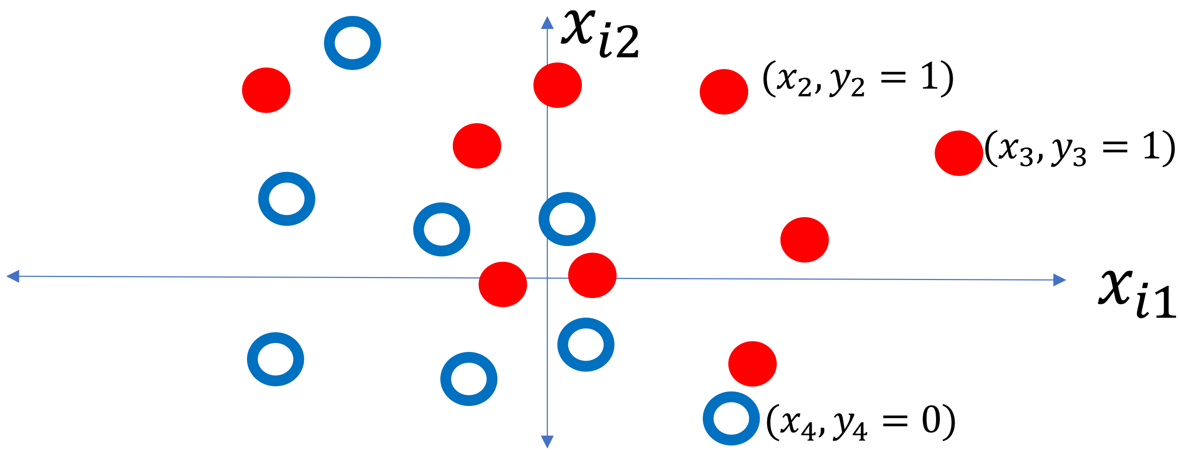
$$P(\mathfrak{D}|\theta) = \prod_{i=1}^{n} P(Y = y_i | X = x_i)$$



Learning logistic regression

We want to learn the model parameters, $\theta = \{w_0, \dots, w_{V-1}\}$, in order to maximize the probability of the observed data:

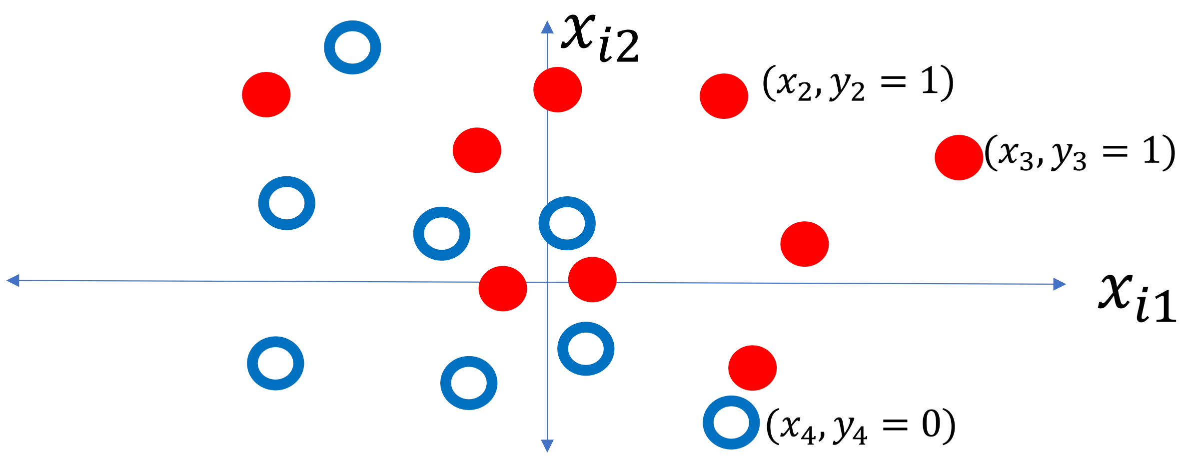
$$P(\mathfrak{D}|\theta) = \prod_{i=1}^{n} \text{softmax}\left(w_{y_i}^T x_i\right)$$



Learning logistic regression

We want to learn the model parameters, $\theta = \{w_0, \dots, w_{V-1}\}$, in order to maximize the probability of the observed data:

$$P(\mathfrak{D}|\theta) = \prod_{i=1}^{n} \frac{\exp\left(w_{y_i}^T x_i\right)}{\sum_{k=0}^{V-1} \exp(w_k^T x_i)}$$



Outline

- Advantages and disadvantages of the perceptron
- Probabilistic-boundary classifiers
- How do you maximize a function?
- Learning a logistic regression
- Two-class logistic regression

How do you maximize a function?

Our goal is to find $\theta = \{w_0, \ldots, w_{V-1}\}$ in order to maximize

$$P(\mathfrak{D}|\theta) = \prod_{i=1}^{n} \frac{\exp\left(w_{y_i}^T x_i\right)}{\sum_{k=0}^{V-1} \exp(w_k^T x_i)}$$

Here are some things you know:

1. Logarithm turns products into sums.
2. Gradient ascent: if you want to find θ in order to maximize $f(\theta)$, you take a step in the direction $+\nabla_\theta f$.

How do you ~~maximize~~ minimize a function?

Our goal is to find $\theta = \{w_0, \dots, w_{V-1}\}$ in order to maximize

$$\mathfrak{L} = -\log P(\mathfrak{D}|\theta) = -\log \prod_{i=1}^{n} \frac{\exp\left(w_{y_i}^T x_i\right)}{\sum_{k=0}^{V-1} \exp(w_k^T x_i)}$$

Here are some things you know:

1. Logarithm turns products into sums.
2. Gradient ~~ascent~~ descent: if you want to find θ in order to ~~maximize~~ minimize $f(\theta)$, you take a step in the direction $-\nabla_\theta f$.

How do you ~~maximize~~ minimize a function?

Our goal is to find $\theta = \{w_0, \dots, w_{V-1}\}$ in order to maximize

$$\mathfrak{L} = -\log P(\mathfrak{D}|\theta) = -\sum_{i=1}^{n}\left(w_{y_i}^T x_i - \log\sum_{k=0}^{V-1}\exp\left(w_k^T x_i\right)\right)$$

Here are some things you know:

1. <u>Logarithm turns products into sums.</u>
2. Gradient ~~ascent~~ descent: if you want to find θ in order to ~~maximize~~ minimize $f(\theta)$, you take a step in the direction $-\nabla_\theta f$.

How do you minimize a function?

Our goal is to find $\theta = \{w_0, \ldots, w_{V-1}\}$ by taking a step in the direction:

$$-\nabla_\theta \mathfrak{L} = \nabla_\theta \log P(\mathfrak{D}|\theta) = \sum_{i=1}^{n} \nabla_\theta \left(w_{y_i}^T x_i - \log \sum_{k=0}^{V-1} \exp\left(w_k^T x_i\right) \right)$$

Here are some things you know:

1. Logarithm turns products into sums.
2. Gradient descent: if you want to find θ in order to minimize $f(\theta)$, you take a step in the direction $-\nabla_\theta f$.

The gradient of the log softmax

Our goal is to find $\theta = \{w_0, \dots, w_{V-1}\}$ by taking a step in the direction $-\nabla_\theta \mathfrak{L}$.
The gradient is just the partial derivative w.r.t. each vector:

$$\nabla_{w_c}\left(w_{y_i}^T x_i - \log \sum_{k=0}^{V-1} \exp\left(w_k^T x_i\right)\right) = \begin{cases} \left(1 - \dfrac{\exp(w_c^T x_i)}{\sum_{k=0}^{V-1} \exp(w_k^T x_i)}\right) x_i & c = y_i \\ \left(0 - \dfrac{\exp(w_c^T x_i)}{\sum_{k=0}^{V-1} \exp(w_k^T x_i)}\right) x_i & c \neq y_i \end{cases}$$

Outline

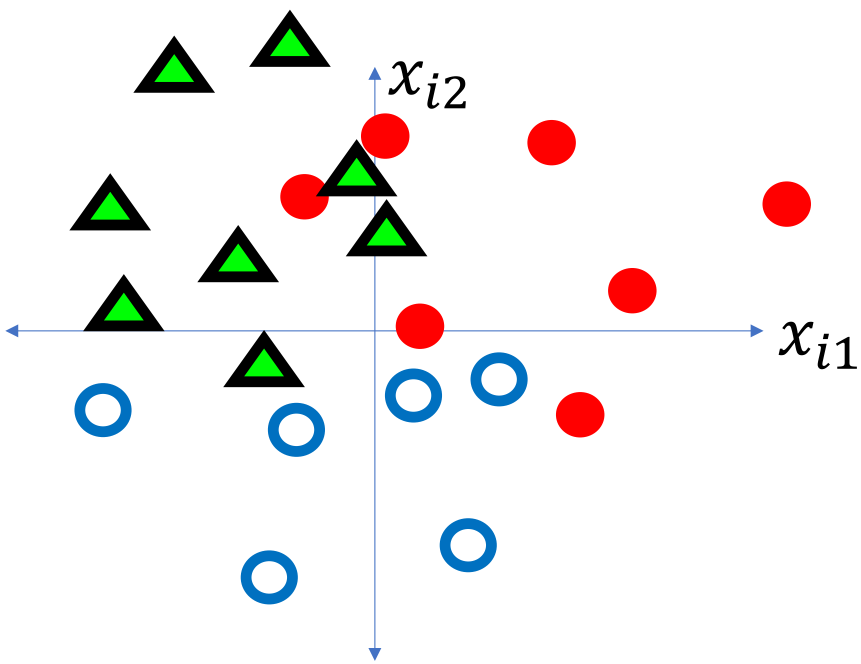
- Advantages and disadvantages of the perceptron
- Probabilistic-boundary classifiers
- How do you maximize a function?
- Learning a logistic regression
- Two-class logistic regression

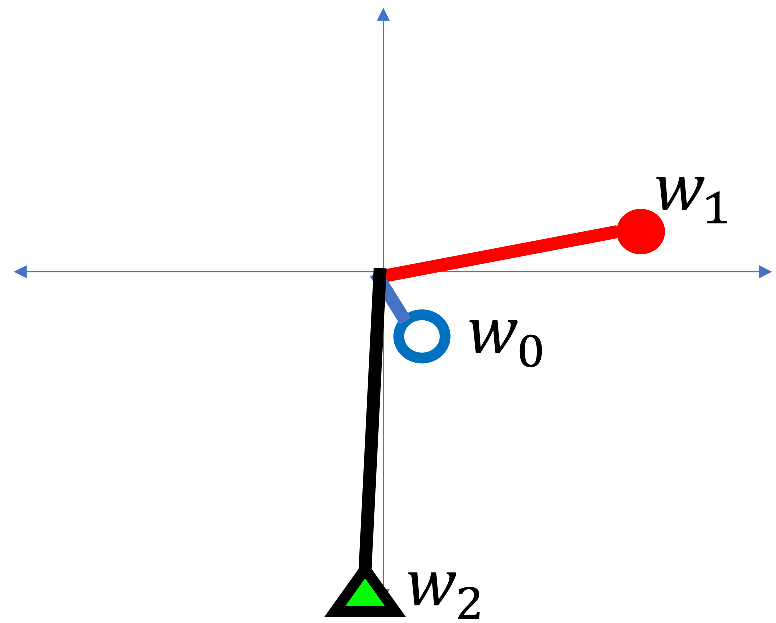
Logistic regression training

- In each iteration, present a batch of training data, $\mathfrak{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$.
  - If the batch contains all the data, this is called "gradient descent"
  - If the batch contains a randomly chosen subset of the data, this is called "stochastic gradient descent"
- Calculate $\mathrm{P}(\mathrm{Y} = c|X = x_i) = \mathrm{softmax}(w_c^T x_i)$ for each training token x_i, for each class c.
- Update all the weight vectors as $w_c = w_c - \eta \nabla_{w_c} \mathfrak{L}$

Logistic regression training example

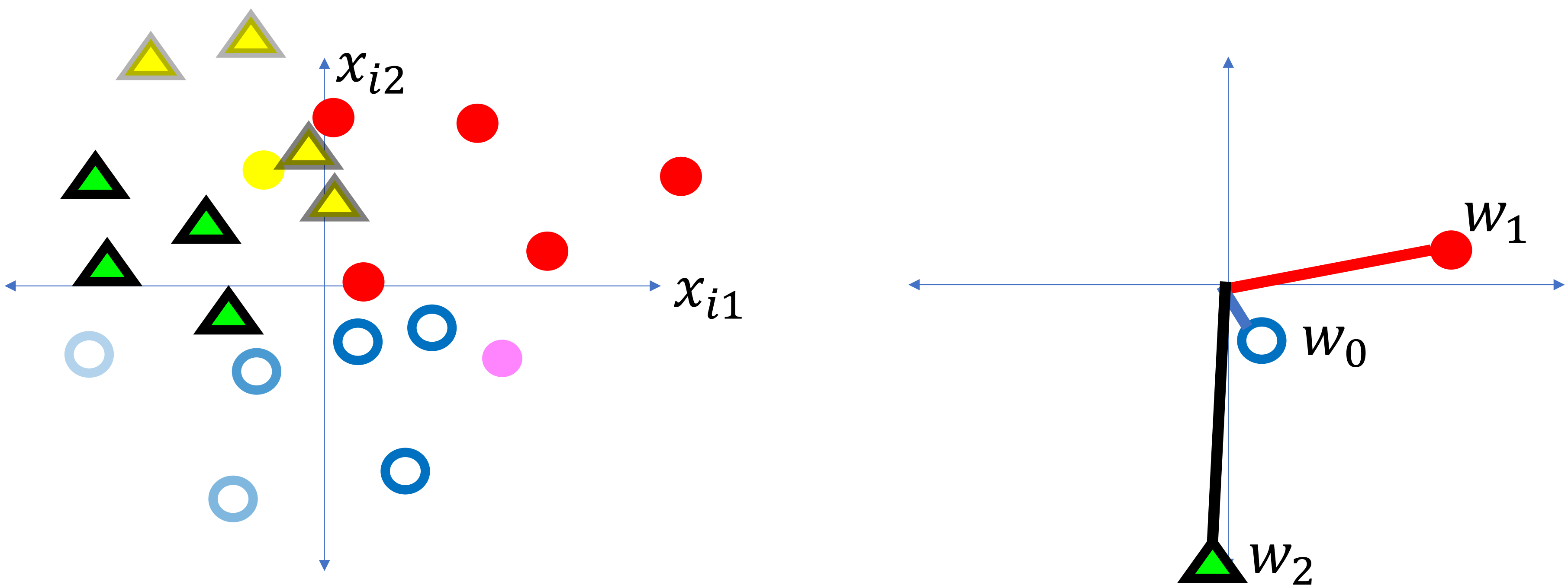
Start with the given dataset $\mathfrak{D}$ (left side), and with randomly initiated weight vectors (right side).





Logistic regression training example

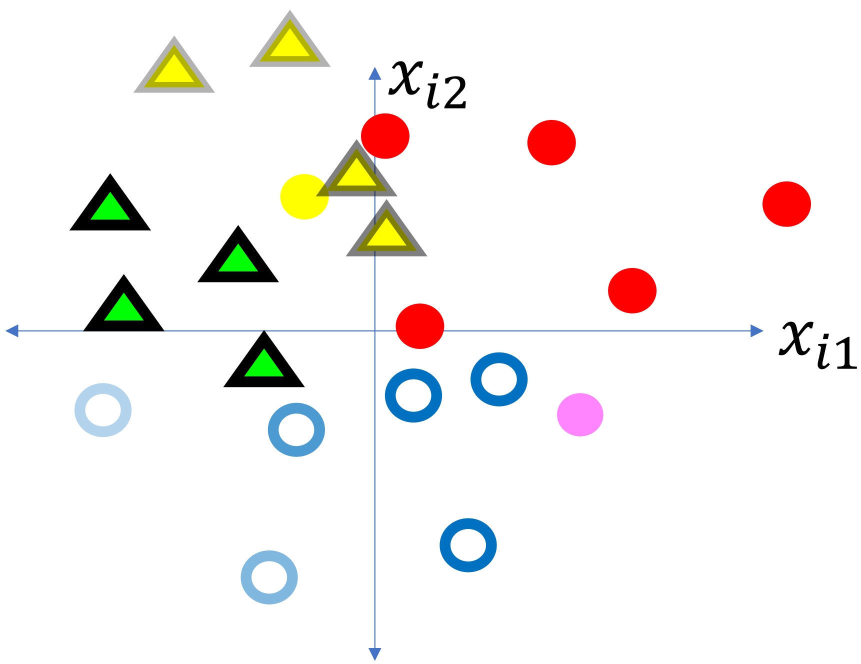
Calculate the probabilities $\mathrm{P}(\mathrm{Y} = c|X = x_i)$ for every class c, for every training token x_i (shown as transparency and color change, left side)

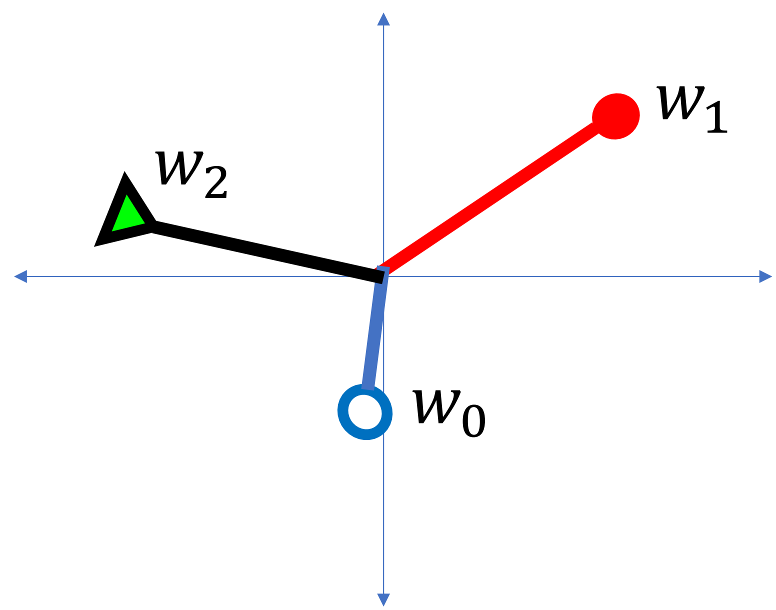


Logistic regression training example

Modify the weight vectors to reduce the loss function, as

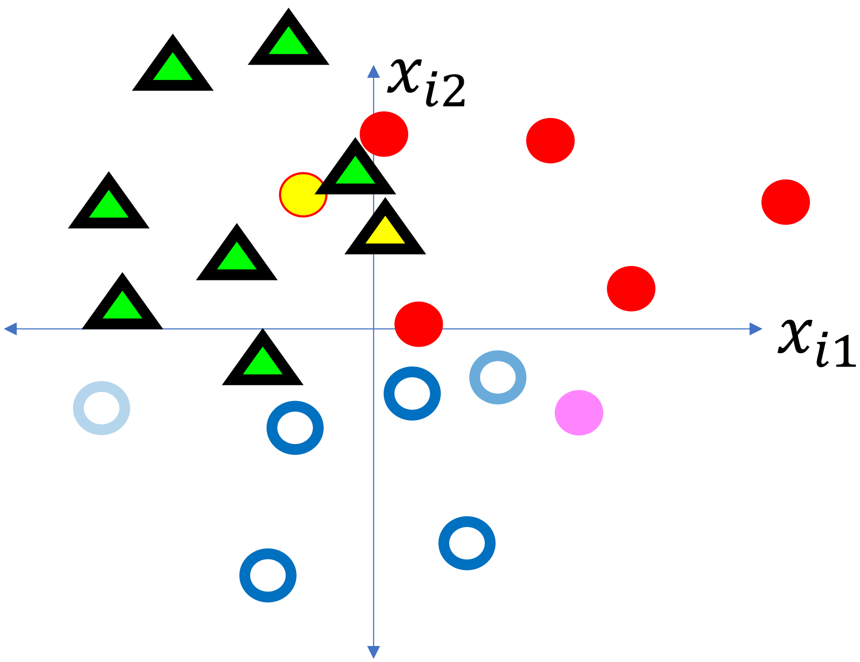
$$w_c = w_c - \eta \nabla_{w_c} \mathfrak{L}$$

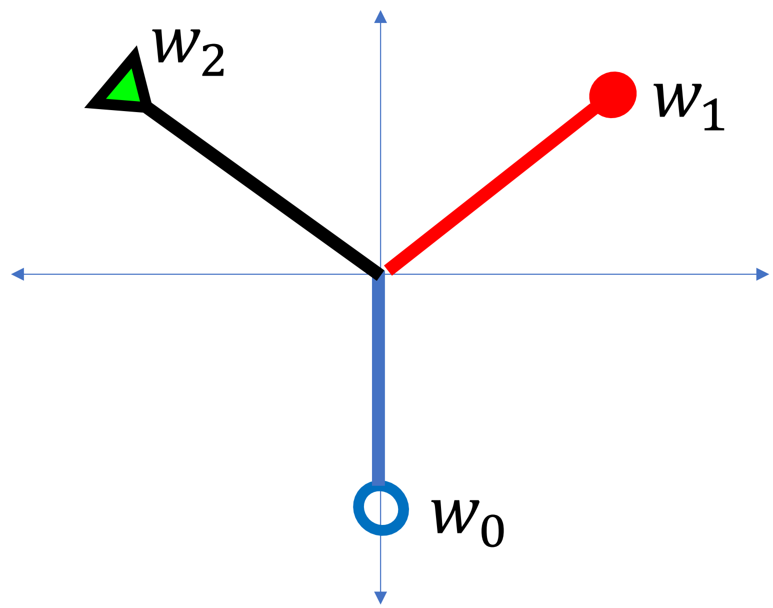




Logistic regression training example

Repeat until the loss stops decreasing.





Some details: Learning Rate

- The learning rate, for logistic regression, is much smaller than for perceptron. Typically $\eta \approx 0.001$.
- It's very hard to know in advance what learning rate will work for a particular problem. Usually you need to try some experiments to see what works.

Some details: Cross entropy

- The loss function is called “cross entropy,” because it is similar in some ways to the entropy of a thermodynamic system in physics.
- Usually we normalize by the number of training tokens, so that the scale is easier to understand:

$$\mathfrak{L} = -\frac{1}{n}\log P(\mathfrak{D}|\theta) = -\frac{1}{n}\sum_{i=1}^{n}\log \mathrm{P}(\mathrm{Y} = y_i|X = x_i)$$

Outline

- Advantages and disadvantages of the perceptron
- Probabilistic-boundary classifiers
- How do you maximize a function?
- Learning a logistic regression
- Two-class logistic regression

Some details: Binary cross entropy

- For two-class problems, it's wasteful to compute both $\mathrm{P}(\mathrm{Y} = 0|X = x_i)$ and $\mathrm{P}(\mathrm{Y} = 1|X = x_i)$, so sometimes we don't.
- Instead, we use binary cross entropy, which is:

$$\mathfrak{L} = -\frac{1}{n}\left(\sum_{i:y_i=1} \log \mathrm{P}(\mathrm{Y} = 1|X = x_i) + \sum_{i:y_i=0} \log\big(1 - \mathrm{P}(\mathrm{Y} = 1|X = x_i)\big)\right)$$

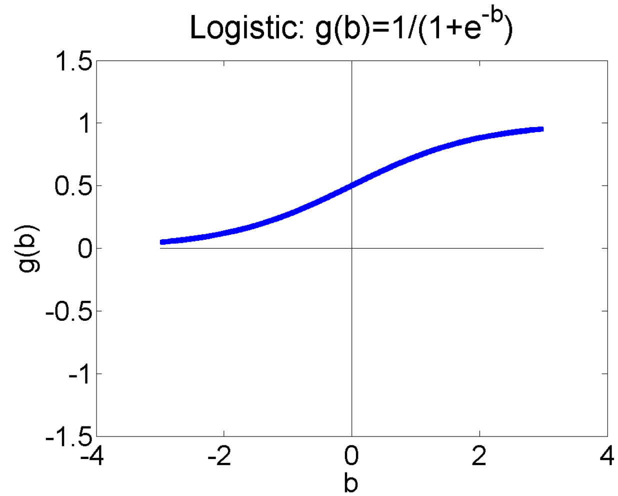
Some details: Logistic function

The probability $\mathrm{P}(\mathrm{Y}=1|X=x)$ in the two-class case is particularly simple. It's

$$\mathrm{P}(\mathrm{Y}=1|X=x)=\mathrm{softmax}(w_1^T x)=\frac{e^{w_1^T x}}{e^{w_1^T x}+e^{w_0^T x}}=\frac{1}{1+e^{-w^T x}}$$

where $w=w_1-w_0$.

Some details: Logistic function



This function,

$$P(Y = 1|X = x) = \frac{1}{1 + e^{-w^T x}}$$

is called the "logistic sigmoid function."

- It's called "sigmoid" because it is S-shaped.
- It was first discovered by Verhulst in the 1830s, as a model of population growth. The idea was that the population grows exponentially until it runs up against resource limitations, and then starts to stagnate.

Logistic Regression

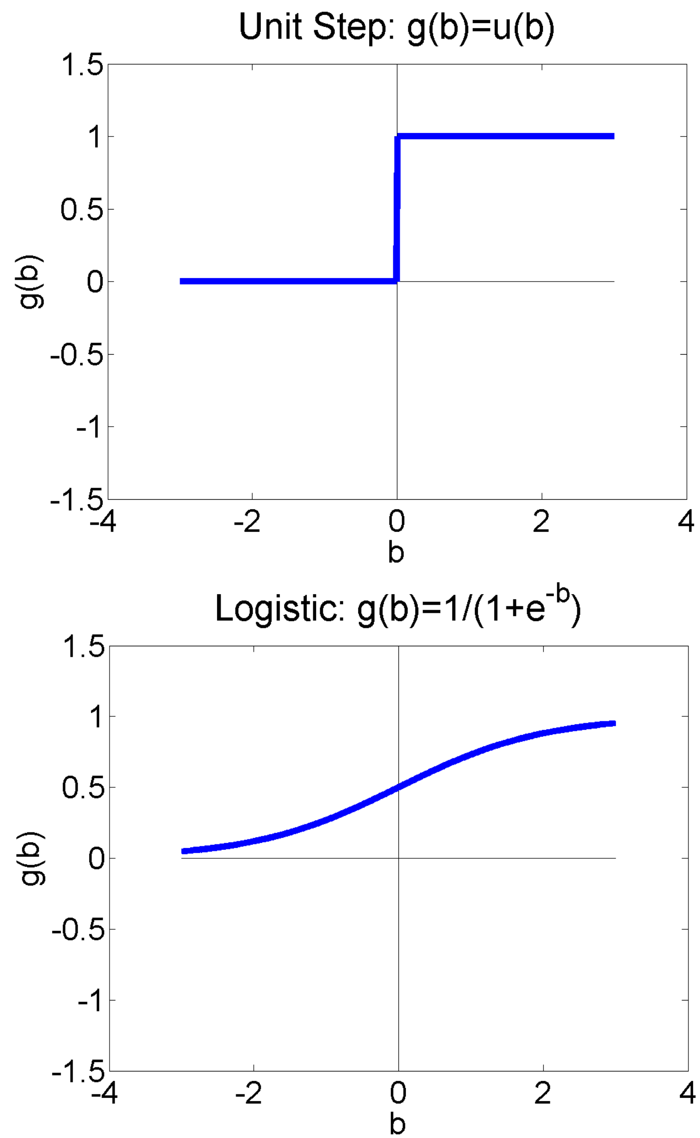
We can frame the basic idea of logistic regression in this way: replace the non-differentiable decision function

$$\hat{y} = \mathrm{u}(w^T x)$$

with a differentiable decision function:

$$\hat{y} = \sigma(w^T x) = \frac{1}{1 + e^{-w^T x}}$$

…so that the classifier can be trained using gradient descent.



Conclusion: Comparing logistic regression vs. the perceptron

Logistic regression:

For all training tokens, whether right or wrong,

$$w = w - \eta \nabla_w \mathfrak{L} = w + \eta \frac{1}{n} \nabla_w \log \mathrm{P}(\mathrm{Y} = y_i | X = x_i)$$

Perceptron:

- If $y_i = \widehat{y_i}$ then do nothing.
- If $y_i \neq \widehat{y_i}$ then set $w = w + \eta y_i x_i$

Conclusion: Comparing multi-class logistic regression vs. multi-class perceptron

Logistic regression:

For all training tokens, for all classes, even if c $\neq y_i$,

$$w_c = w_c - \eta \nabla_{w_c} \mathfrak{L} = w_c + \eta \frac{1}{n} \nabla_{w_c} \log \mathrm{P}(\mathrm{Y} = y_i | X = x_i)$$

Multi-class Perceptron:

- If $y_i = \widehat{y_i}$ then do nothing.
- If $y_i \neq \widehat{y_i}$ then
  - update the correct class, y_i, as $w_{y_i} = w_{y_i} + \eta x_i$
  - update the incorrect class, $\widehat{y_i}$, as $w_{\widehat{y_i}} = w_{\widehat{y_i}} - \eta x_i$