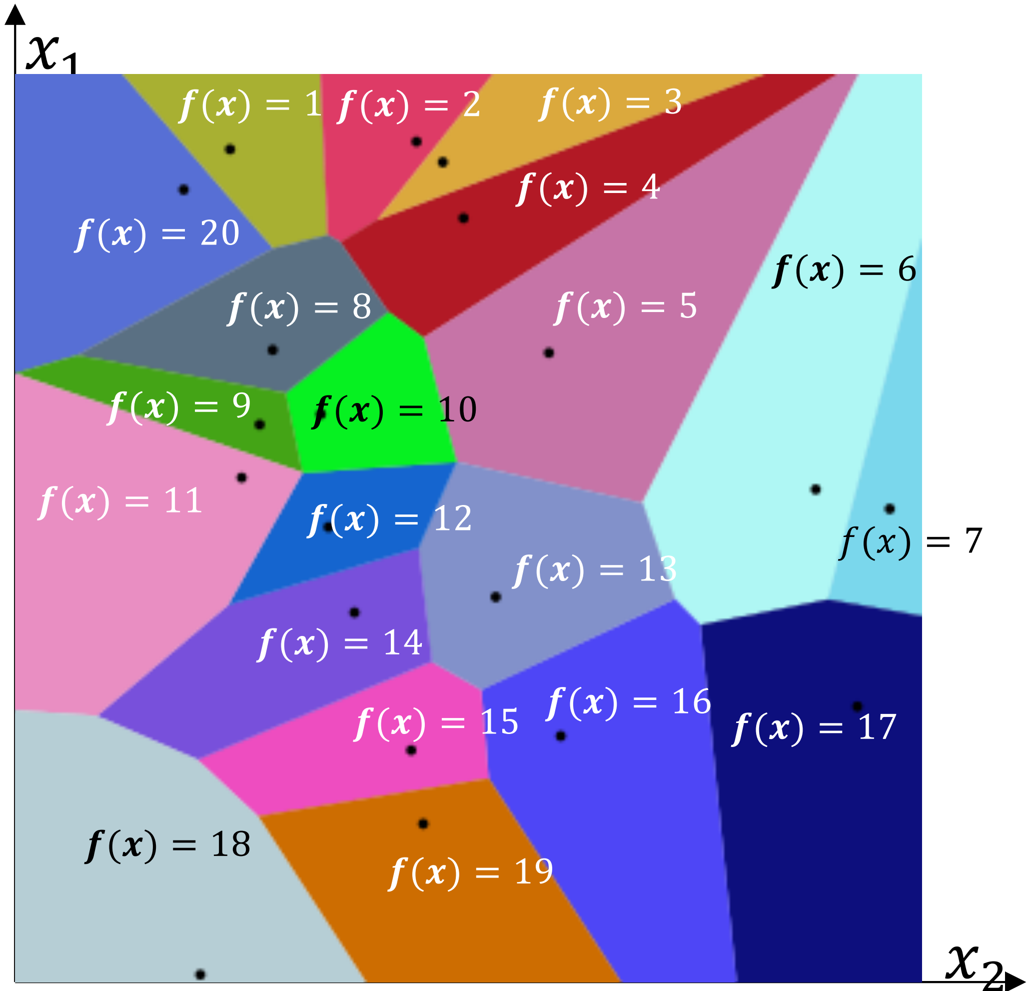
CS440/ECE448 Lecture 11: Softmax

Mark Hasegawa-Johnson

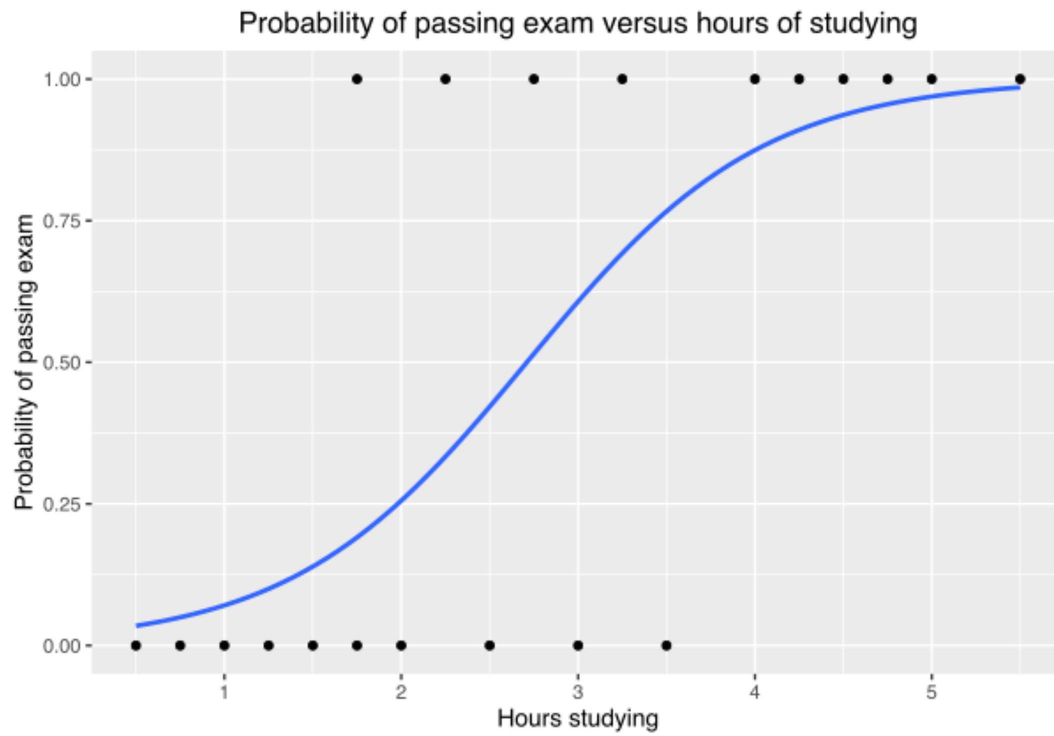


Outline

- Sigmoid review
- Multi-class linear classifiers
- One-hot vectors
- Softmax nonlinearity
- Derivative of the log softmax

Logistic regression: Output is a probability

- The label is either $y = 0$ or $y = 1$
- Model:
  $$P(Y = 1|\boldsymbol{x}) = f(\boldsymbol{x}) = \sigma(\boldsymbol{x}^T \boldsymbol{w})$$
- Logistic sigmoid function:
  $$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Loss function: Binary cross entropy

- Loss function

$$\mathcal{L} = -\sum_{i=1}^{n} \log P(Y = y_i | \boldsymbol{x}_i)$$

$$= \sum_{y_i=1} -\log \sigma\left(\boldsymbol{x}_i^T \boldsymbol{w}\right) = \sum_{y_i=0} -\log\left(1 - \sigma(\boldsymbol{x}_i^T \boldsymbol{w})\right)$$

- Derivative of the sigmoid

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

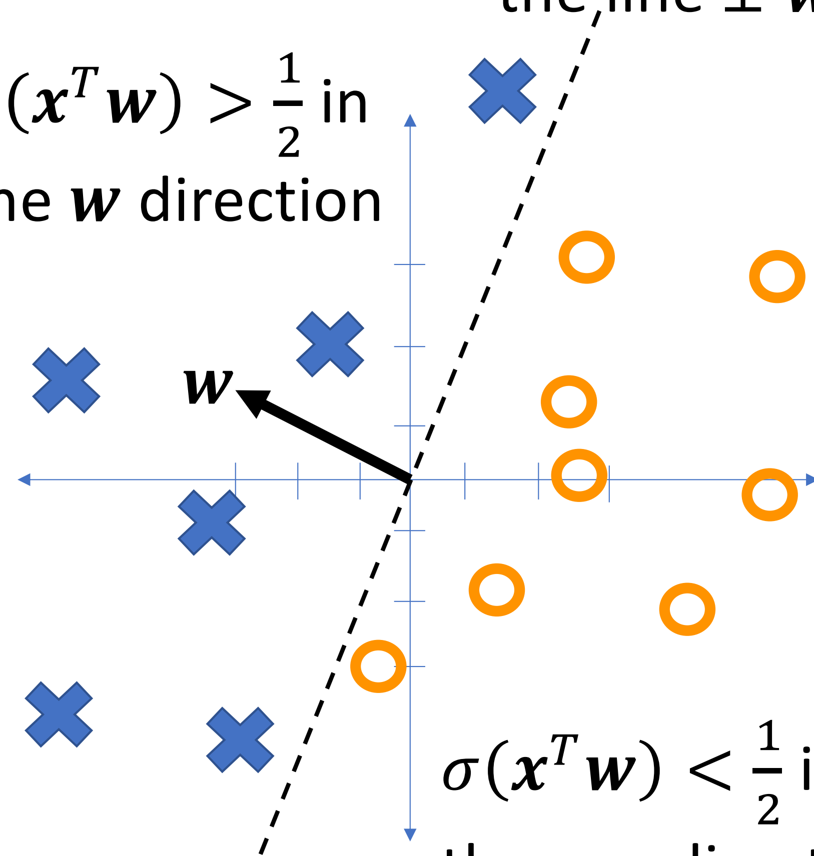
- Derivative of the log sigmoid

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = -\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \left(y_i - \sigma\left(\boldsymbol{x}_i^T \boldsymbol{w}\right)\right) \boldsymbol{x}_i$$

Geometric interpretation

Suppose $\boldsymbol{x}$ is a 2d vector. Then:

- $\boldsymbol{x}^T\boldsymbol{w} = 0$, and $\sigma(\boldsymbol{x}^T\boldsymbol{w}) = \frac{1}{2}$, on the line where $\boldsymbol{x} \perp \boldsymbol{w}$
- $\boldsymbol{x}^T\boldsymbol{w} > 0$, and $\sigma(\boldsymbol{x}^T\boldsymbol{w}) > \frac{1}{2}$, in the $\boldsymbol{w}$ direction
- $\boldsymbol{x}^T\boldsymbol{w} < 0$, and $\sigma(\boldsymbol{x}^T\boldsymbol{w}) < \frac{1}{2}$, in the $-\boldsymbol{w}$ direction



Outline

- Sigmoid review
- Multi-class linear classifiers
- One-hot vectors
- Softmax nonlinearity
- Derivative of the log softmax

Linear classifier: Notation

- The observation $\boldsymbol{x}^T = [x_1, \dots, x_d]$ is a real-valued vector (d is the number of feature dimensions)
- The class label $y \in \mathcal{Y}$ is drawn from some finite set of class labels.
- Usually the output vocabulary, $\mathcal{Y}$, is some set of strings. For convenience, though, we usually map the class labels to a sequence of integers, $\mathcal{Y} = \{1, \dots, v\}$, where v is the vocabulary size.

Linear classifier: Definition

A linear classifier is defined by

$$f(\boldsymbol{x}) = \operatorname{argmax} \boldsymbol{W}\boldsymbol{x}$$

where:

$$\boldsymbol{W}\boldsymbol{x} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,d} \\ \vdots & \ddots & \vdots \\ w_{v,1} & \cdots & w_{v,d} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}_1^T \boldsymbol{x} \\ \vdots \\ \boldsymbol{w}_v^T \boldsymbol{x} \end{bmatrix}$$

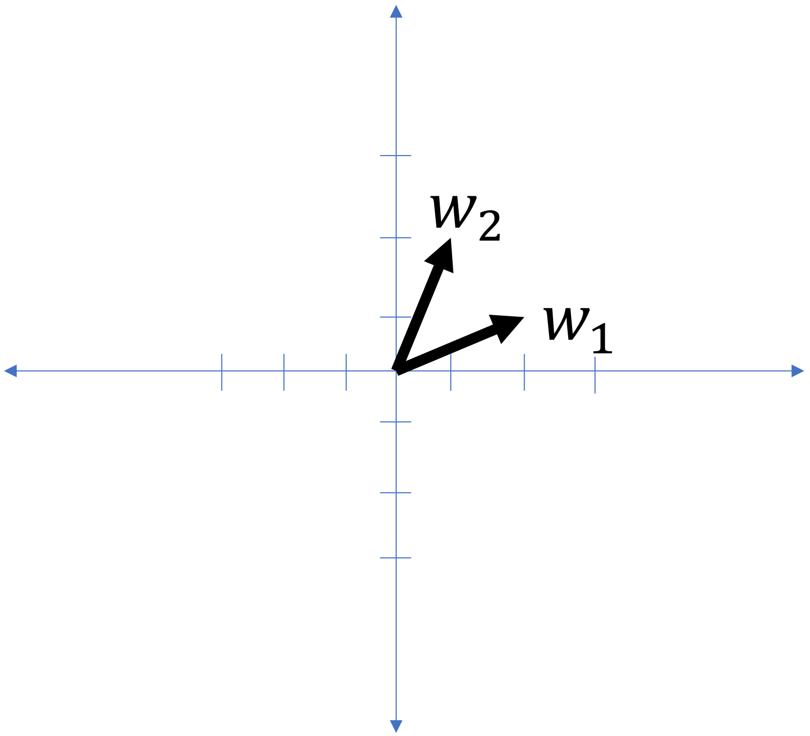
$\boldsymbol{w}_k$ is the **weight vector** corresponding to **class k**, and the argmax function finds the element of the vector $\boldsymbol{W}\boldsymbol{x}$ with the largest value.

There are a total of vd trainable parameters: the elements of the matrix $\boldsymbol{W}$.

Example

Consider a two-class classification problem, i.e., $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2]^T$, where:

$$\boldsymbol{w}_1^T = [w_{1,1}, w_{1,2}] = [2,1]$$
$$\boldsymbol{w}_2^T = [w_{2,1}, w_{2,2}] = [1,2]$$



Example

Notice that in the two-class case, the equation

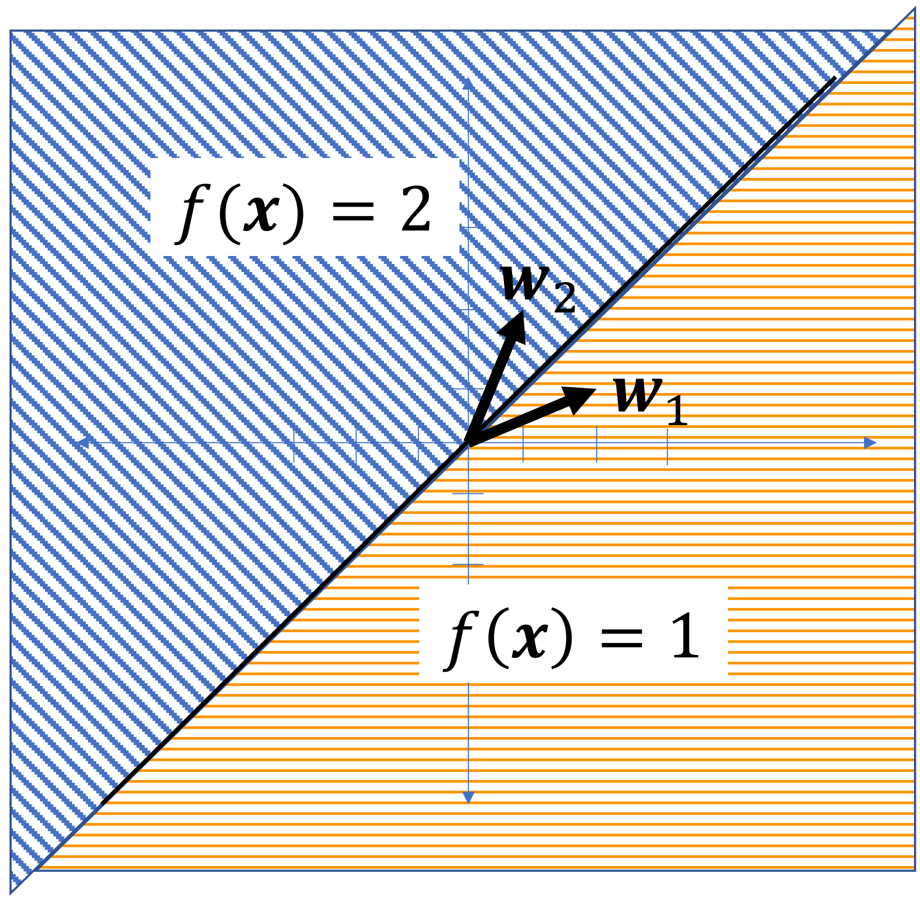
$$f(\boldsymbol{x}) = \text{argmax}\, \boldsymbol{W}\boldsymbol{x}$$

Simplifies to

$$f(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{w}_1^T\boldsymbol{x} > \boldsymbol{w}_2^T\boldsymbol{x} \\ 2 & \boldsymbol{w}_1^T\boldsymbol{x} < \boldsymbol{w}_2^T\boldsymbol{x} \end{cases}$$

The class boundary is the line whose equation is

$$(\boldsymbol{w}_2 - \boldsymbol{w}_1)^T\boldsymbol{x} = 0$$



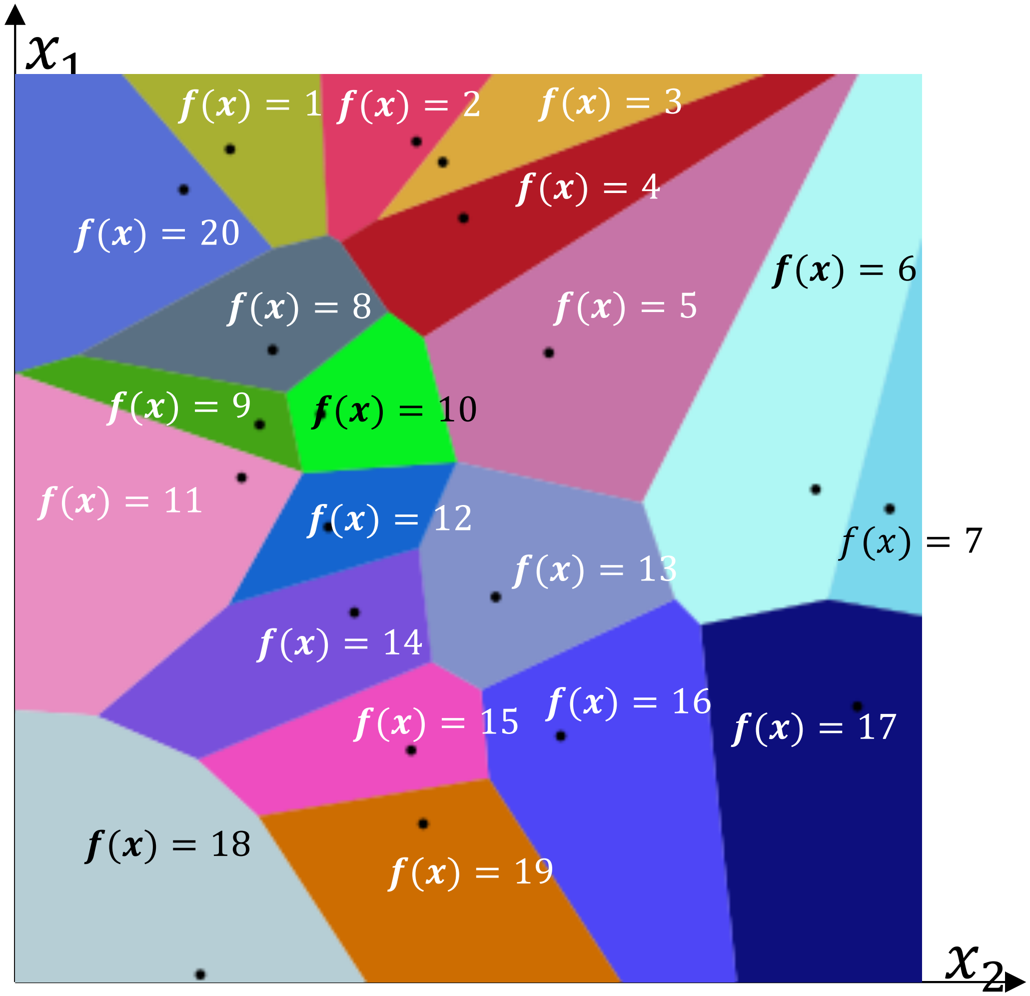
Multi-class linear classifier

In a general multi-class linear classifier,

$$f(\boldsymbol{x}) = \operatorname{argmax} \boldsymbol{W}\boldsymbol{x}$$

The boundary between class k and class l is the line (or plane, or hyperplane) given by the equation

$$(\boldsymbol{w}_k - \boldsymbol{w}_l)^T \boldsymbol{x} = 0$$

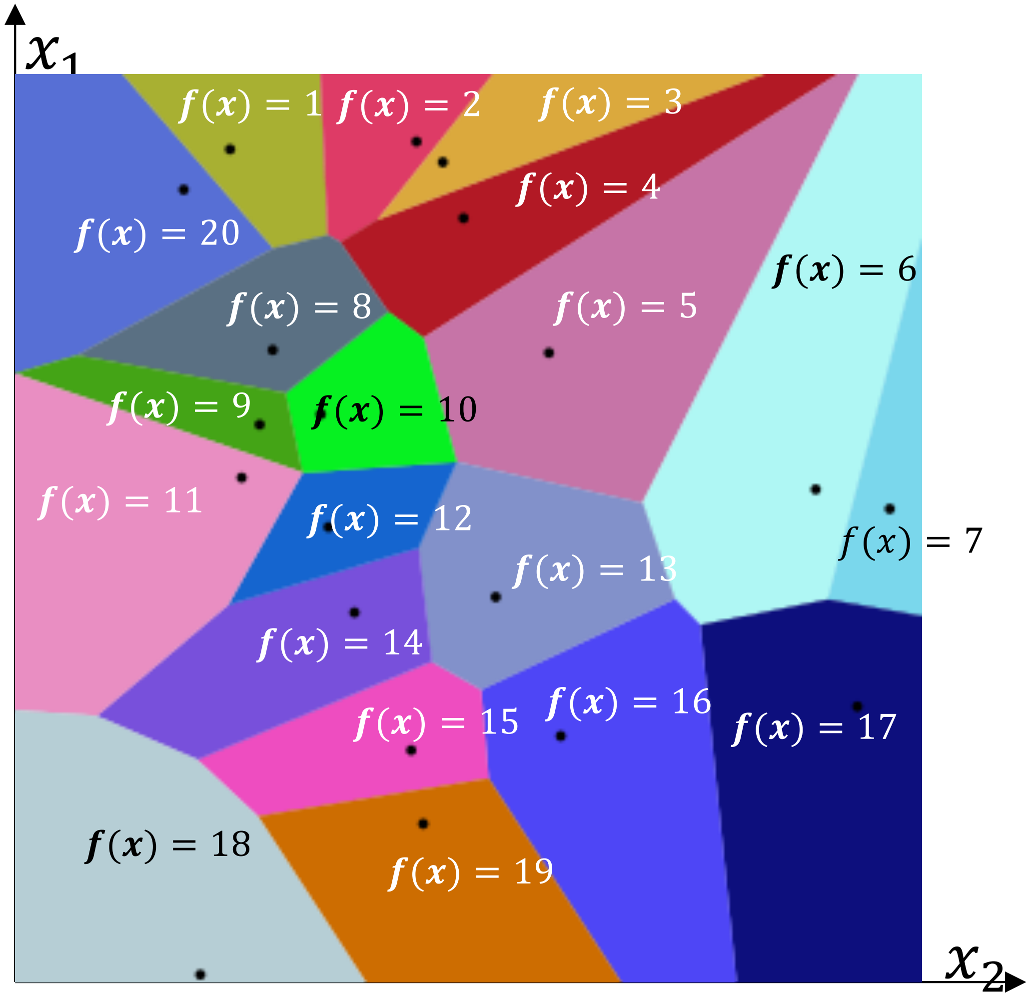


Voronoi regions

The classification regions in a linear classifier are called Voronoi regions.

A **Voronoi region** is a region that is

- Convex (if $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are points in the region, then every point on the line segment connecting them is also in the region)
- Bounded by piece-wise linear boundaries



Outline

- Sigmoid review
- Multi-class linear classifiers
- One-hot vectors
- Softmax nonlinearity
- Derivative of the log softmax

One-hot vectors

A **one-hot vector** is a binary vector in which all elements are 0 except for a single element that's equal to 1.

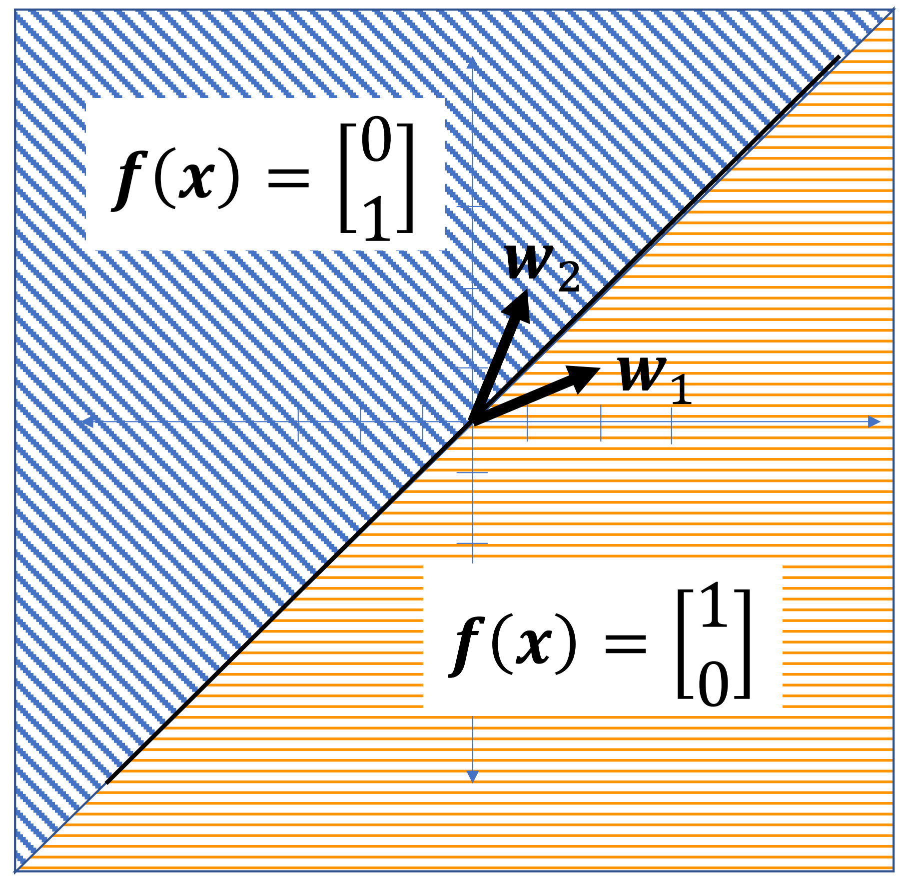
Example: Binary classifier

Consider the classifier

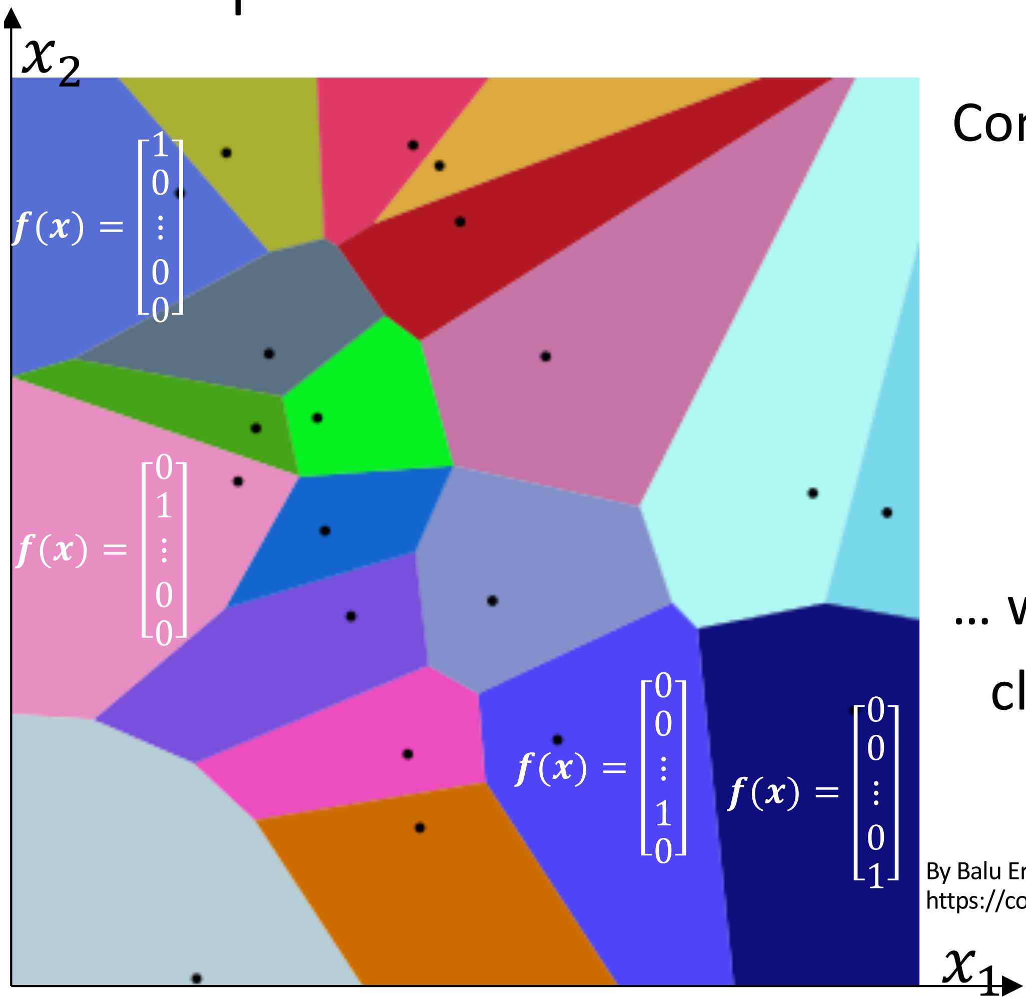
$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \mathbb{1}_{\text{argmax}\, \boldsymbol{W}\boldsymbol{x}=1} \\ \mathbb{1}_{\text{argmax}\, \boldsymbol{W}\boldsymbol{x}=2} \end{bmatrix}$$

…where $\mathbb{1}_P$ is called the "indicator function," and it means:

$$\mathbb{1}_P = \begin{cases} 1 & P \text{ is true} \\ 0 & P \text{ is false} \end{cases}$$



Example: Multi-Class



Consider the classifier

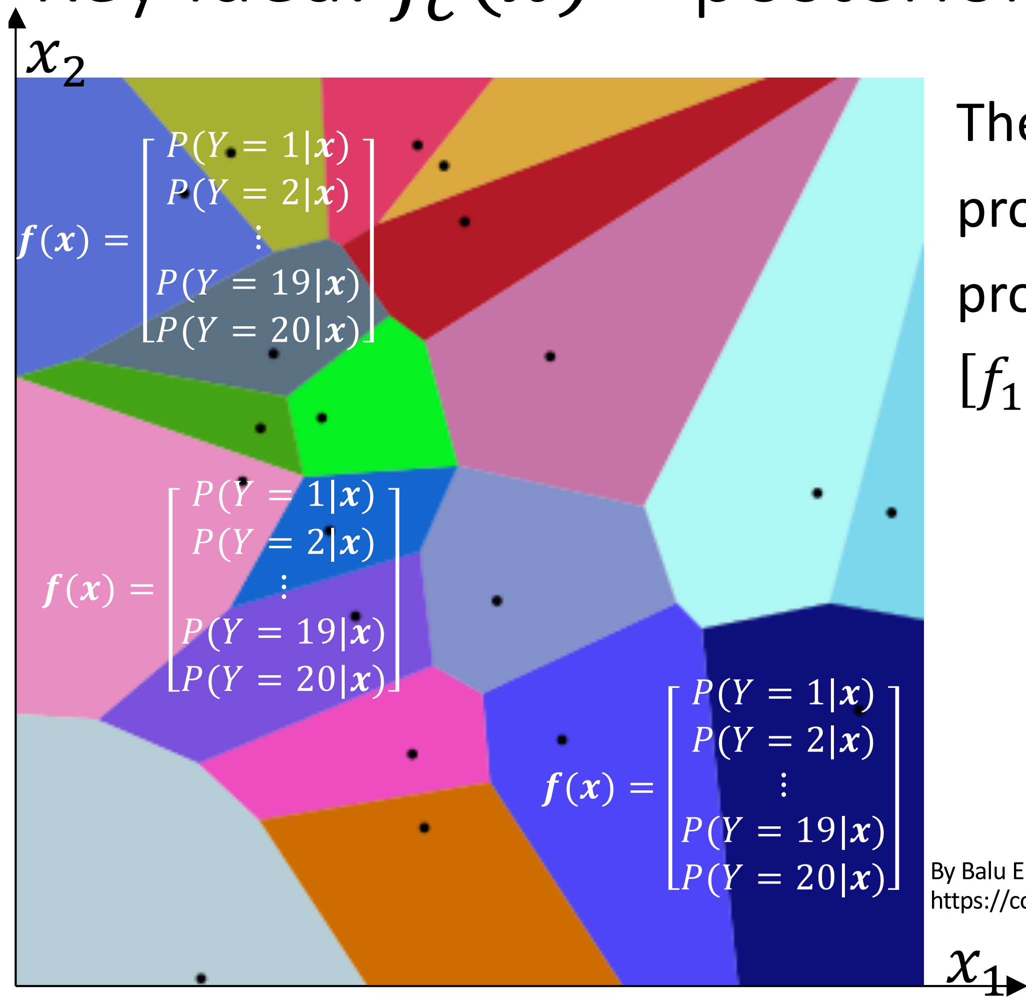
$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_v(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \mathbb{I}_{\operatorname{argmax} \boldsymbol{W}\boldsymbol{x}=1} \\ \vdots \\ \mathbb{I}_{\operatorname{argmax} \boldsymbol{W}\boldsymbol{x}=v} \end{bmatrix}$$

… with 20 classes. Then some of the classifications might look like this.

Outline

- Sigmoid review
- Multi-class linear classifiers
- One-hot vectors
- Softmax nonlinearity
- Derivative of the log softmax

Key idea: $f_c(\boldsymbol{x})$ =posterior probability of class $\boldsymbol{c}$



The key idea of a softmax: Instead of producing a one-hot output, the neural net produces a vector $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), \ldots, f_v(\boldsymbol{x})]^T$ such that

$$f_c(\boldsymbol{x}) = \Pr(Y = c|\boldsymbol{x})$$

Key idea: $f_c(\boldsymbol{x})$ =posterior probability of class $\boldsymbol{c}$

A softmax computes $f_c(\boldsymbol{x}) = \Pr(Y = c|\boldsymbol{x})$. The conditions for this to be true are:

1. It needs to satisfy the axioms of probability:
$$0 \le f_c(\boldsymbol{x}) \le 1, \qquad \sum_{c=1}^{v} f_c(\boldsymbol{x}) = 1$$
2. The weight matrix, $\boldsymbol{W}$, is trained using a loss function that encourages $\boldsymbol{f}(\boldsymbol{x})$ to approximate posterior probability of the labels on some training dataset:
$$f_c(\boldsymbol{x}) = \Pr(Y = c|\boldsymbol{x})$$

Softmax satisfies the axioms of probability

- Axiom #1, probabilities are non-negative ($f_k(\boldsymbol{x}) \geq 0$). There are many ways to do this, but one way that works is to choose:

$$f_c(\boldsymbol{x}) \propto \exp(\boldsymbol{w}_c^T \boldsymbol{x})$$

- Axiom #2, probabilities should sum to one ($\sum_{k=1}^{v} f_k(\boldsymbol{x}) = 1$). This can be done by normalizing:

$$f_c(\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_c^T \boldsymbol{x})}{\sum_{k=0}^{V-1} \exp(\boldsymbol{w}_k^T \boldsymbol{x})}$$

The softmax function

This is called the softmax function:

$$\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), \dots, f_v(\boldsymbol{x})]^T$$

$$f_c(\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_c^T \boldsymbol{x})}{\sum_{k=1}^{v} \exp(\boldsymbol{w}_k^T \boldsymbol{x})}$$

…where $\boldsymbol{w}_k^T$ is the k$^{\text{th}}$ row of the matrix $\boldsymbol{W}$.

Quiz

Go to PrairieLearn, try the quiz!

Outline

- Sigmoid review
- Multi-class linear classifiers
- One-hot vectors
- Softmax nonlinearity
- Derivative of the log softmax

Gradient descent

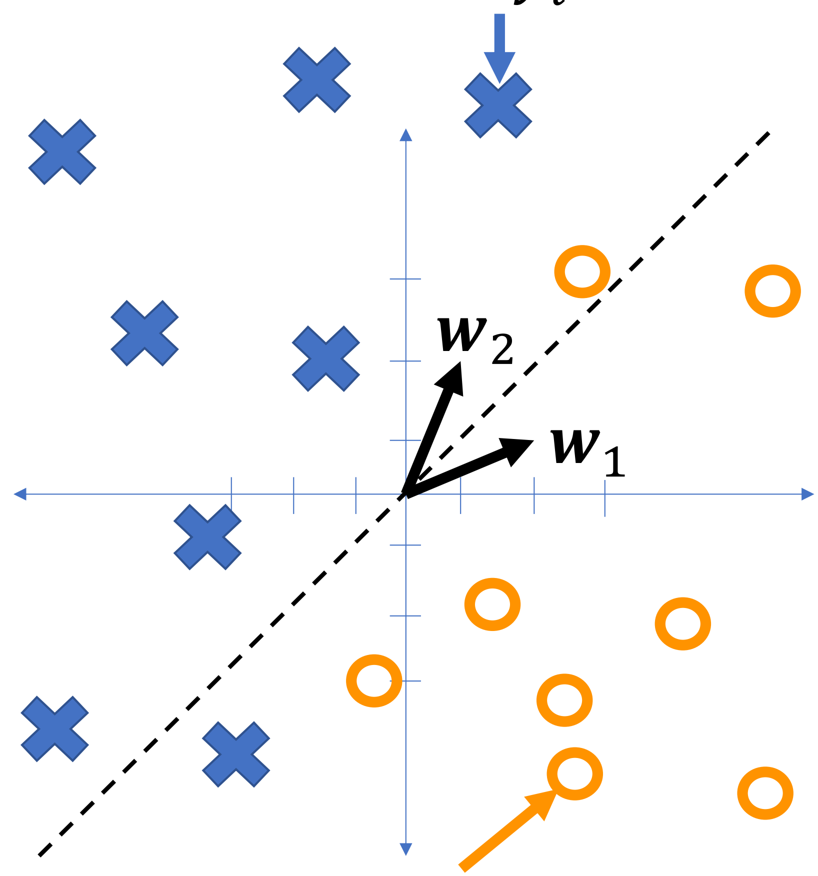
Suppose we have training tokens (x_i, y_i), and we have some initial class vectors w_1 and w_2. We want to update them as

$$\boldsymbol{w}_1 \leftarrow \boldsymbol{w}_1 - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_1}$$

$$\boldsymbol{w}_2 \leftarrow \boldsymbol{w}_2 - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_2}$$

…where $\mathcal{L}$ is some loss function. What loss function makes sense?

Training token $\boldsymbol{x}_i$ of class $y_i = 2$



Training token $\boldsymbol{x}_i$ of class $y_i =$

Loss = negative log probability of the training set

Suppose we have a softmax output, so we want $f_c(\boldsymbol{x}) = \Pr(Y = c|\boldsymbol{x})$. We can train this by learning $\boldsymbol{W}$ to maximize the probability of the training corpus. If we assume all training tokens are independent, we get:

$$\boldsymbol{W} = \underset{\boldsymbol{W}}{\operatorname{argmax}} \prod_{i=1}^{n} \Pr(Y = y_i|\boldsymbol{x}_i)$$

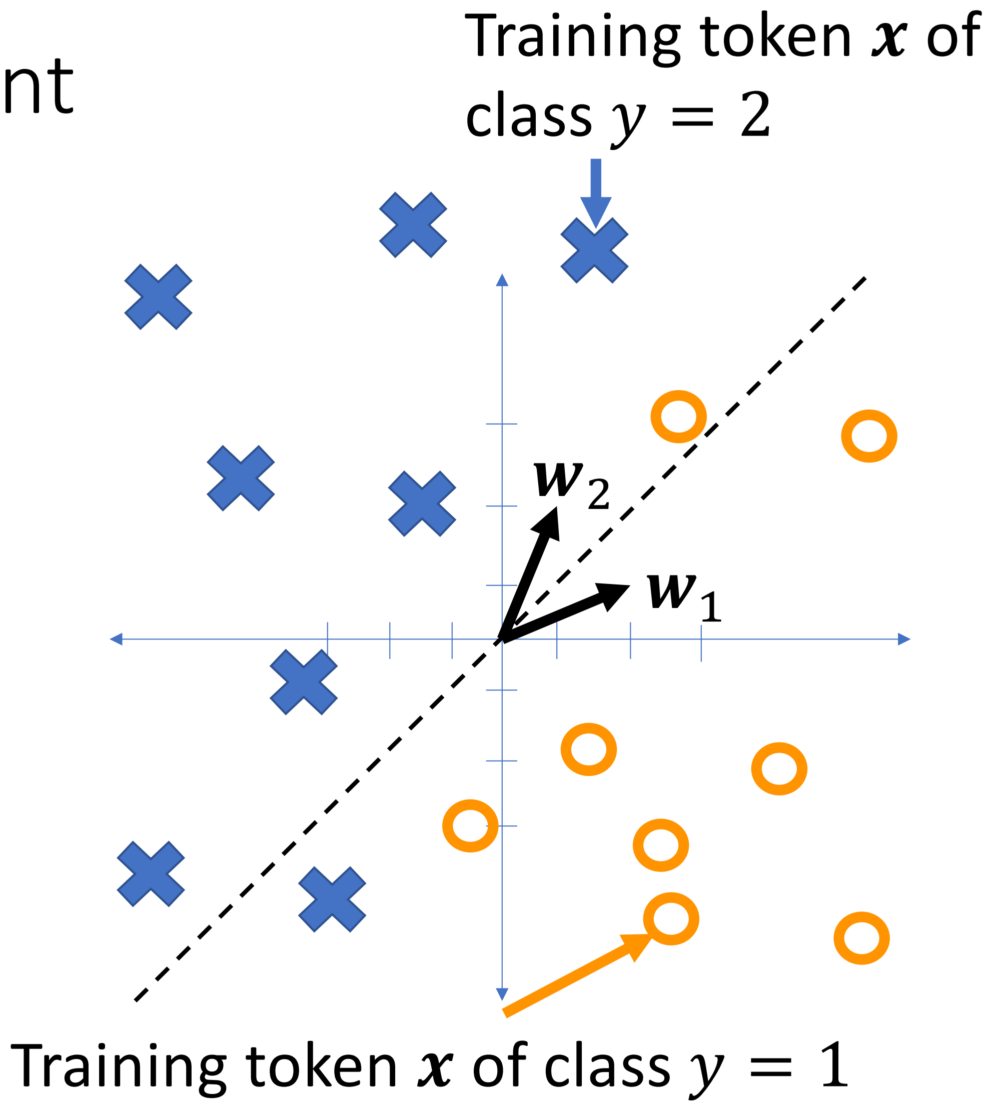
$$= \underset{\boldsymbol{W}}{\operatorname{argmax}} \frac{1}{n} \sum_{i=1}^{\boldsymbol{n}} \ln \Pr(Y = y_i|\boldsymbol{x}_i)$$

$$= \underset{\boldsymbol{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i \,, \qquad \mathcal{L}_i = -\ln f_{y_i}(\boldsymbol{x}_i)$$

Stochastic gradient descent

Suppose we have a training example $(\boldsymbol{x}, y)$. We want to find

$$\boldsymbol{w}_c \leftarrow \boldsymbol{w}_c - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_c}$$



Stochastic gradient descent

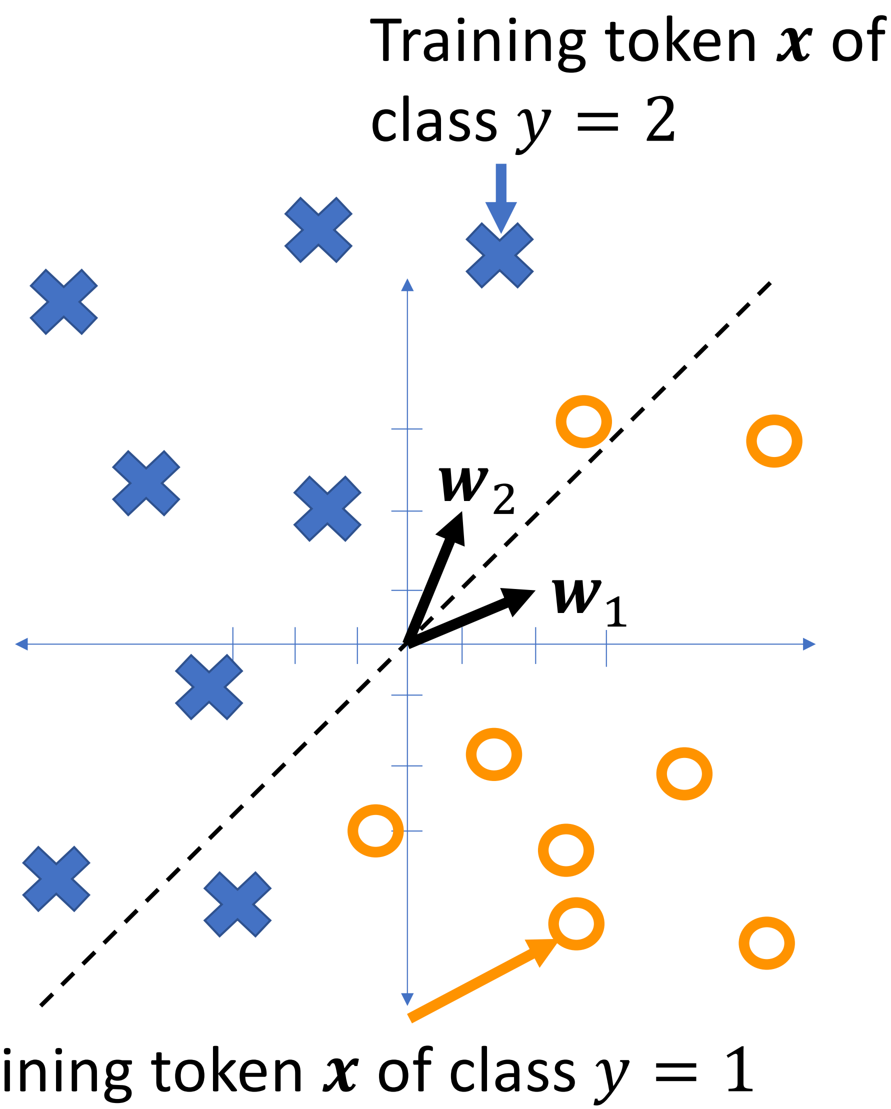
Suppose we have a training example $(\boldsymbol{x}, y)$. We want to find

$$\boldsymbol{w}_c \leftarrow \boldsymbol{w}_c - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_c}$$

$$\mathcal{L} = -\ln f_y(\boldsymbol{x}) = -\ln\left(\frac{e^{\boldsymbol{w}_y^T \boldsymbol{x}}}{\sum_{k=1}^{v} e^{\boldsymbol{w}_k^T \boldsymbol{x}}}\right)$$

…and after some work, we find out that:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_c} = \left(f_c(\boldsymbol{x}) - \mathbb{1}_{y=c}\right)\boldsymbol{x}$$



Stochastic gradient descent

Suppose we have a training example $(\boldsymbol{x}, y)$.

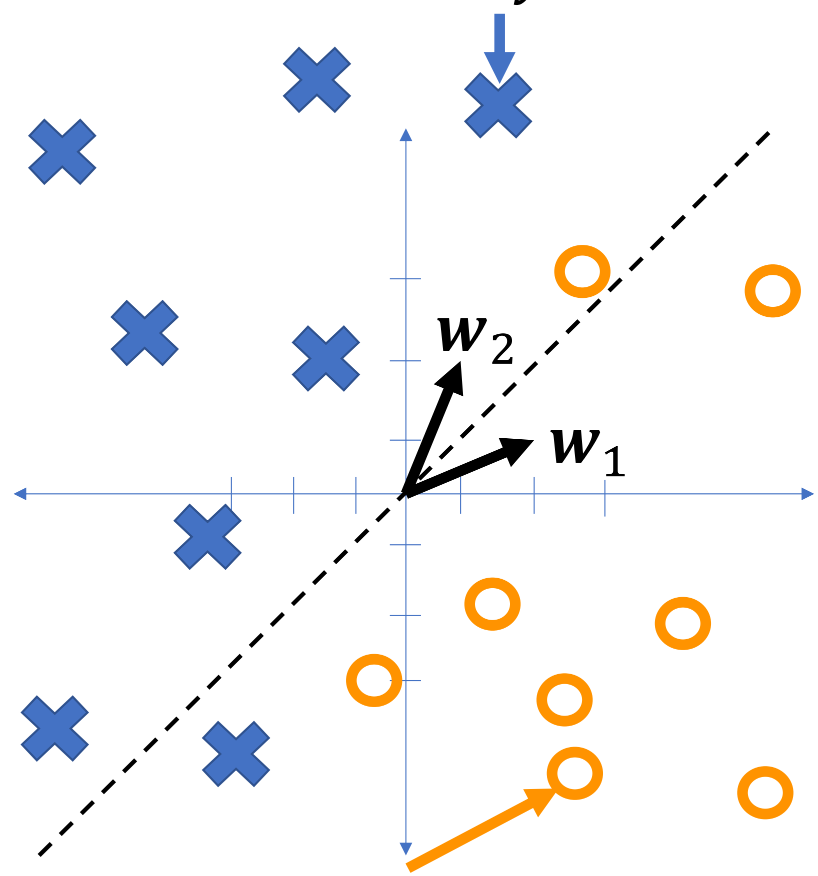
- Shift $\boldsymbol{w}_y$ in the direction of $\boldsymbol{x}$:

$$\boldsymbol{w}_c \leftarrow \boldsymbol{w}_c + \eta\left(1 - f_y(\boldsymbol{x})\right)\boldsymbol{x}$$

- For all other classes, $c \neq y$, shift them in the direction opposite $\boldsymbol{x}$:

$$\boldsymbol{w}_c \leftarrow \boldsymbol{w}_c + \eta\left(0 - f_y(\boldsymbol{x})\right)\boldsymbol{x}$$

Training token $\boldsymbol{x}$ of class $y = 2$



Training token $\boldsymbol{x}$ of class $y = 1$

Summary

- Sigmoid review

$$f(\boldsymbol{x}) = \sigma(\boldsymbol{x}^T \boldsymbol{w}), \qquad \boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \big(y_i - f(\boldsymbol{x})\big)\boldsymbol{x}_i$$

- Multi-class linear classifiers

$$f(\boldsymbol{x}) = \operatorname{argmax} \boldsymbol{W}\boldsymbol{x}$$

- One-hot vectors

$$f_c(\boldsymbol{x}) = \mathbb{1}_{\operatorname{argmax} \boldsymbol{W}\boldsymbol{x}=c}$$

- Softmax nonlinearity

$$f_c(\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_c^T \boldsymbol{x})}{\sum_{k=1}^{v} \exp(\boldsymbol{w}_k^T \boldsymbol{x})}$$

- Derivative of the log softmax

$$\boldsymbol{w}_c \leftarrow \boldsymbol{w}_c + \eta \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \left(\mathbb{1}_{y_i=c} - f_c(\boldsymbol{x})\right) \boldsymbol{x}_i$$