CoE202
Fundamentals of Artificial intelligence
<Big Data Analysis and Machine Learning>

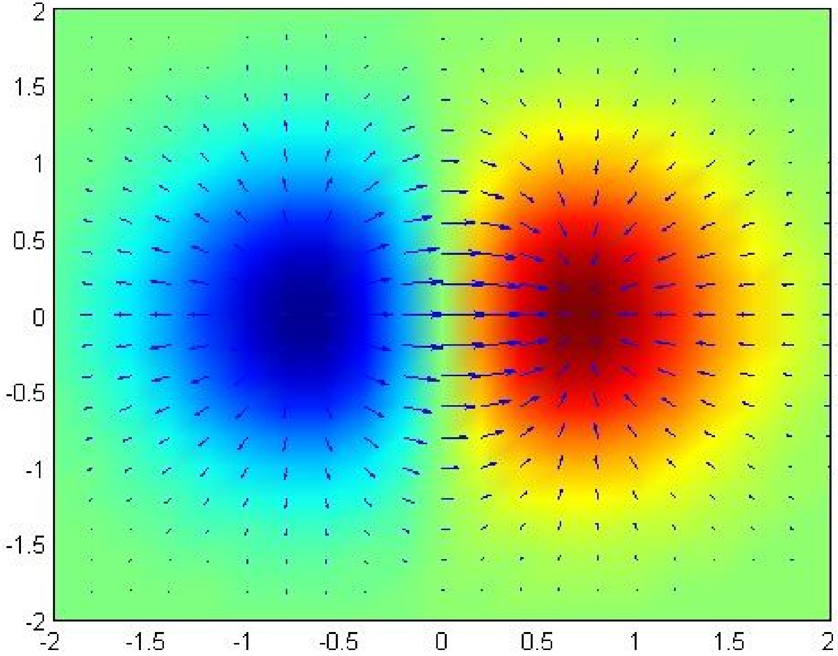
Logistic regression & Linear classification

Prof. Young-Gyu Yoon
School of EE, KAIST

Contents

- Recap
- Classification as supervised learning
- Classification framework
- Softmax and sigmoid functions
- Cross entropy loss
- Gradient descent for training classifier

Recap: Gradient



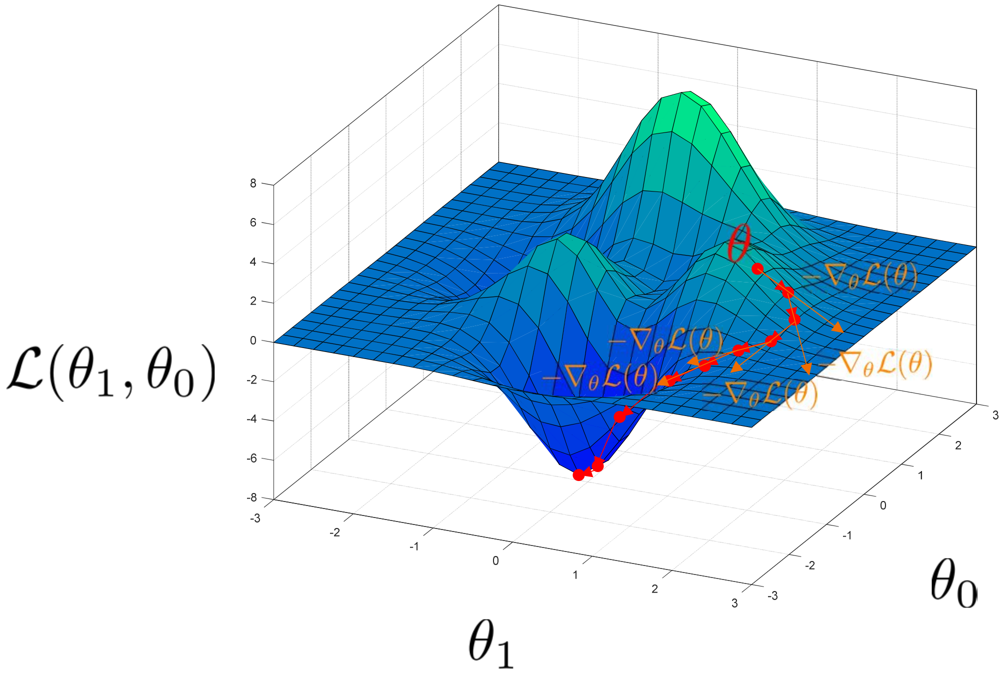
$$\nabla_\theta \mathcal{L}(\theta) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} \mathcal{L}(\theta_1, \theta_0) \\ \frac{\partial}{\partial \theta_1} \mathcal{L}(\theta_1, \theta_0) \end{bmatrix}$$

The gradient at a point is a vector pointing in the direction of the steepest slope at that point.

Recap: Gradient Descent

$$\theta^{(k+1)} = \theta^{(k)} - \gamma \nabla \mathcal{L}(\theta^{(k)})$$

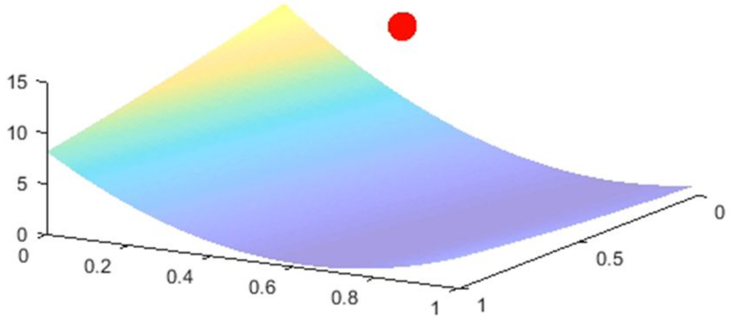
γ : learning rate



Just follow the gradient!

Recap: Gradient Descent

$$\theta^{(k+1)} = \theta^{(k)} - \gamma \nabla \mathcal{L}(\theta^{(k)})$$



- **Gradient descent** is an iterative algorithm for finding a local minimum of a differentiable function
- It requires only the gradient value at one point at each iteration step (does not require closed-form gradient function)

Recap: GD vs. SGD

- **Gradient descent (GD)**

For $k = 1, 2, \cdots, M$:

$$\theta^{(k+1)} = \theta^{(k)} - \gamma \nabla \mathcal{L}(\theta^{(k)})$$

- **Stochastic gradient descent (SGD)**

For $k = 1, 2, \cdots, M$:

For $i = 1, 2, \cdots, n$:

$$\theta^{(k'+1)} = \theta^{(k')} - \gamma \nabla \mathcal{Q}_i(\theta^{(k')}) \qquad (k' = n * (k-1) + i)$$

where $\mathcal{Q}_i(\theta)$ is the loss function for a part of the data set

That is, $\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=0}^{n} \mathcal{Q}_i(\theta)$

Supervised learning: image classification

Seeks a function $f : X \rightarrow Y$

$f\Bigg($  $\Bigg) = \begin{bmatrix}1\\0\end{bmatrix}$

$f\Bigg($  $\Bigg) = \begin{bmatrix}0\\1\end{bmatrix}$

Supervised learning

- **Supervised learning**: learning a function that maps an input to an output based on example input-output pairs

For a data set $\mathcal{D} = \{(\vec{x_1}, \vec{y_1}), (\vec{x_2}, \vec{y_2}), \cdots, (\vec{x_N}, \vec{y_N})\}$

Seeks a function $f : X \to Y$

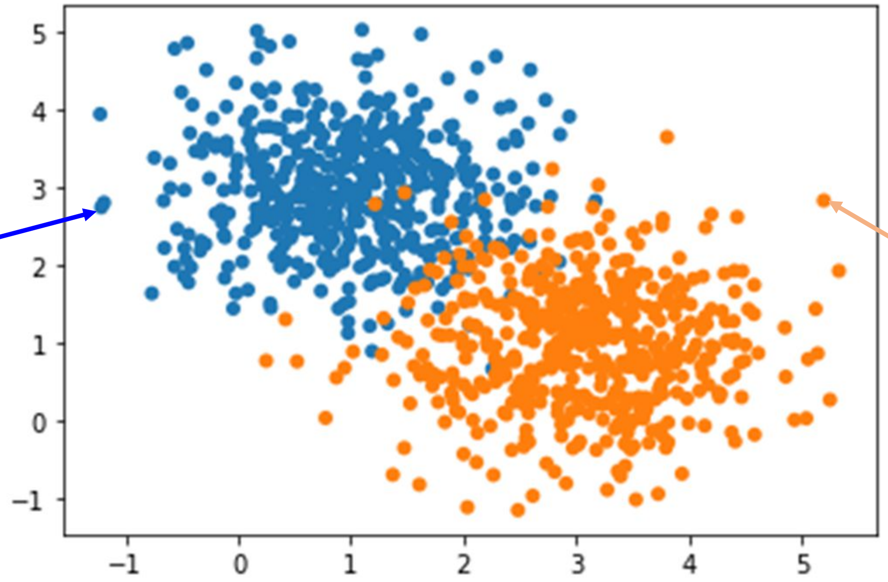
Such that a loss function $\mathcal{L} : X \times Y \to \mathcal{R}$ is minimized

Linear classification example

For a data set $\mathcal{D} = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), \cdots, (\vec{x_N}, y_N)\}$

$$\overrightarrow{x_{10}} = \begin{bmatrix} -1.21 \\ 2.85 \end{bmatrix}$$

$$y_{10} = 1$$

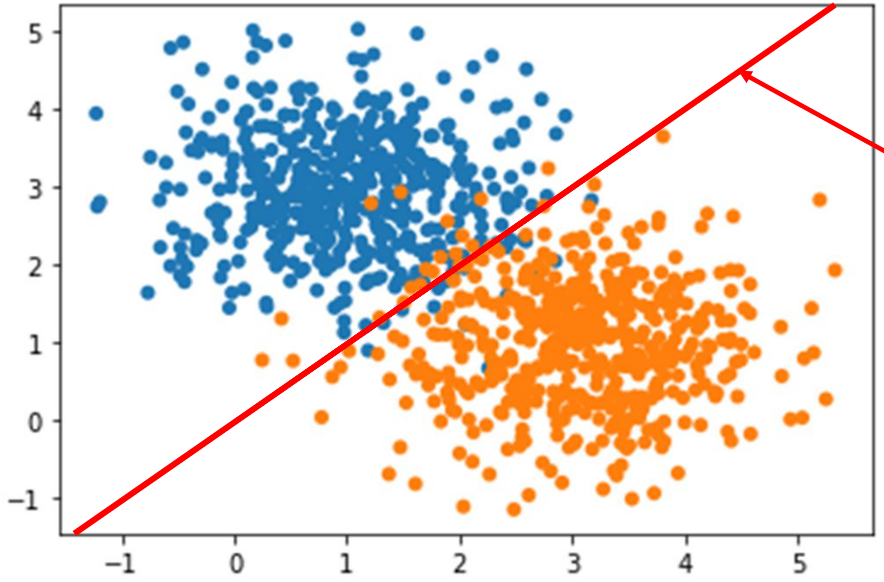


$$\overrightarrow{x_{98}} = \begin{bmatrix} 5.17 \\ 2.91 \end{bmatrix}$$

$$y_{98} = 0$$

Linear classification example

$$\text{Seeks a function } f(\vec{x}; \theta_0, \vec{\theta}) = \vec{\theta} \cdot \vec{x} + \theta_0$$

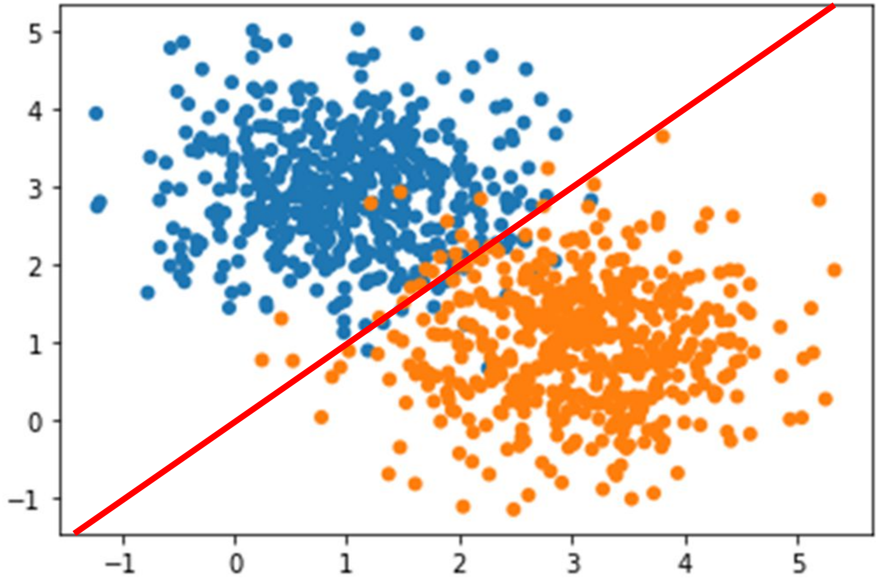


This line is defined by the parameters

Classification can be done by checking if $\theta \cdot x + \theta_0 > 0$

Linear classification example

Such that a loss function $\mathcal{L} : X \times Y \rightarrow \mathcal{R}$ is minimized



What loss function do we want to use?

Back to image classification

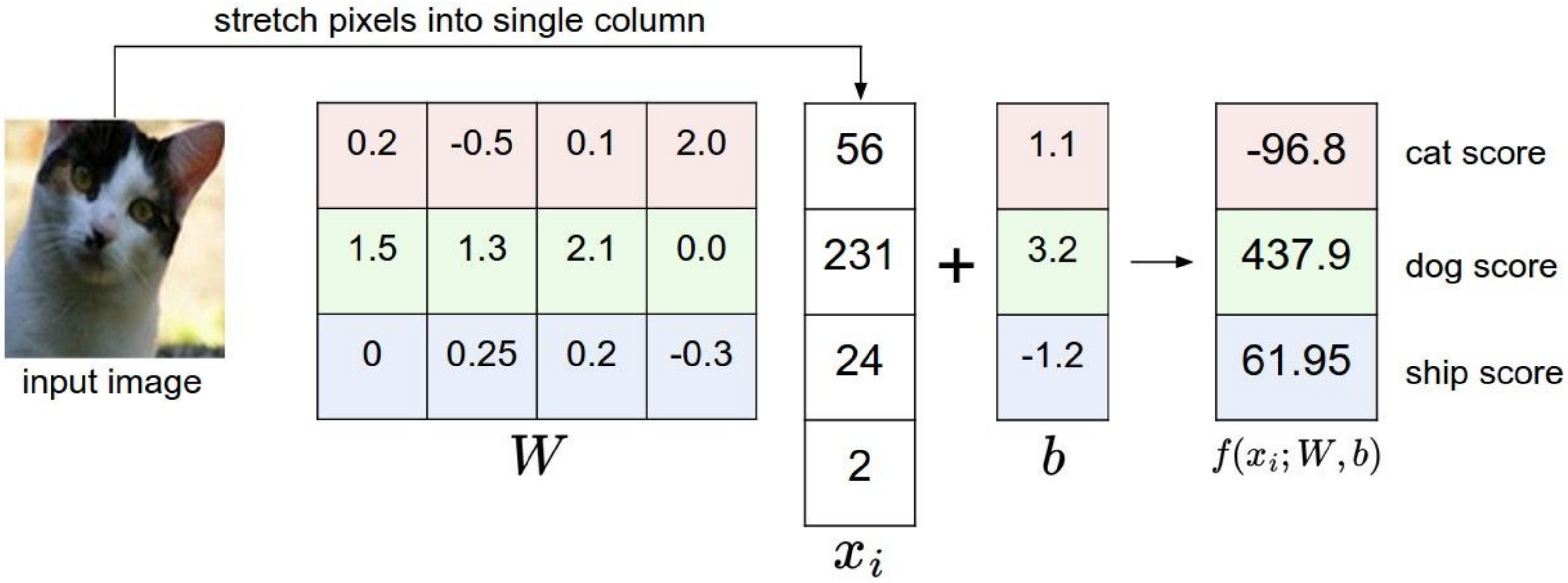


Image classification can use the same framework

Figure from cs231n

Back to image classification

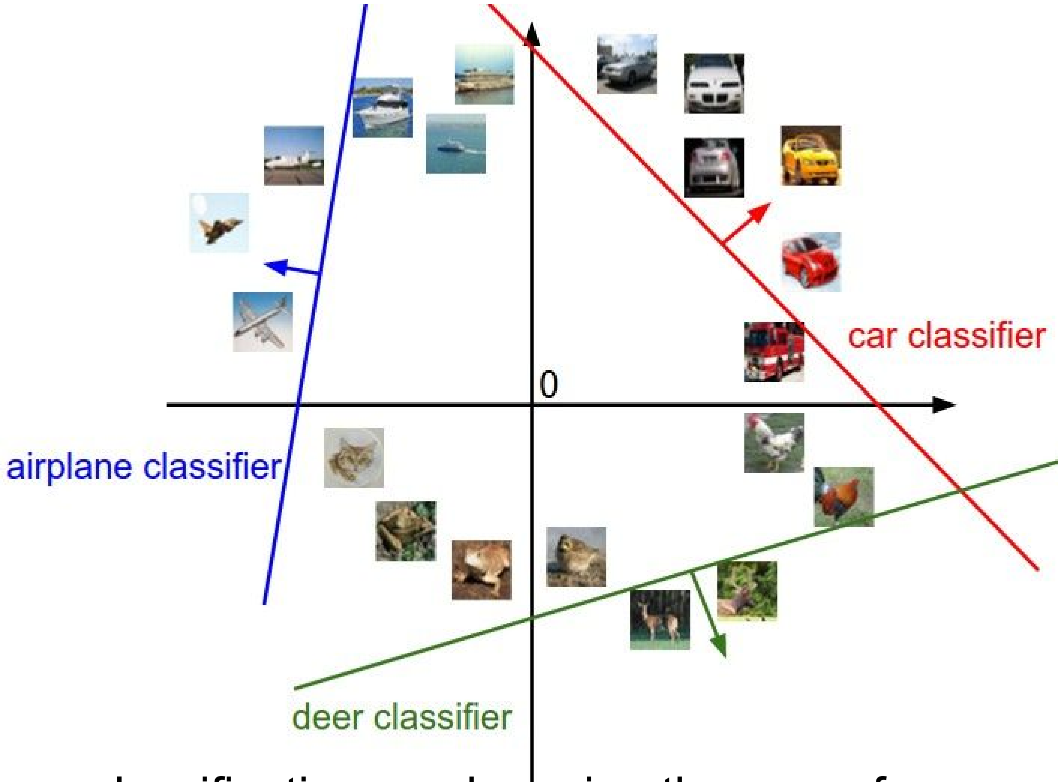
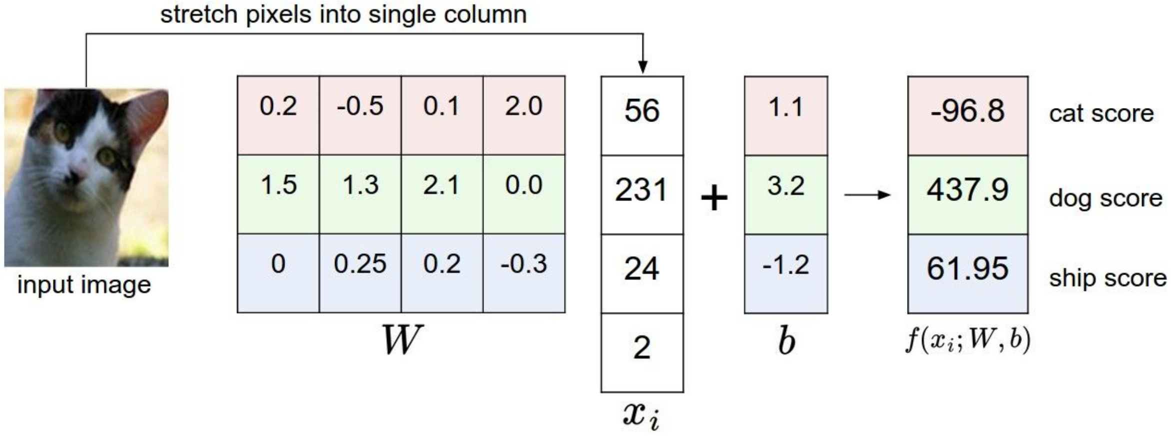


Image classification can be using the same framework

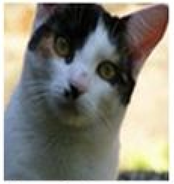
Figure from cs231n

Framework of classification



- The framework above gives us the “score” for each category
- We can pick a category with the highest score for classification
- However, these scores, on their own, do not have any meaning
  - Is 150 a high score? We cannot answer this before comparing to other scores
  - What is a good score to target? Do we want 50? 150? 50,000? Infinite?

Framework of classification



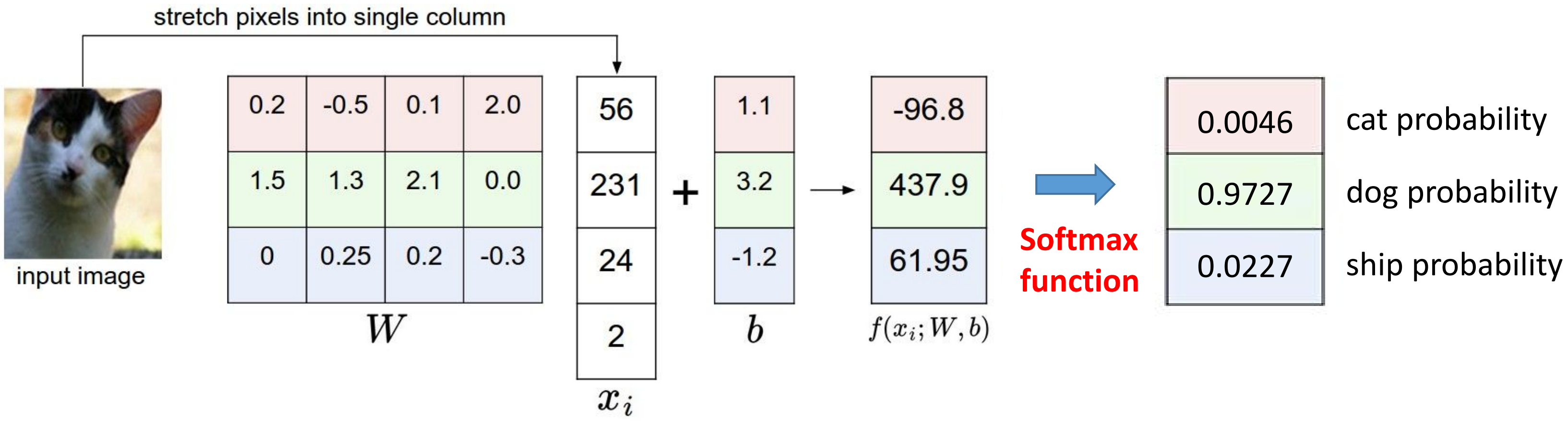
input image

0.0046	cat probability
0.9727	dog probability
0.0227	ship probability

- We can adopt the “**probability**” instead of score (*Bayesian probability)
- Similarly, we can pick a category with the highest probability for classification
- Compared to score, probability is a very intuitive measure
- The sum of the probability, across all possible classes, should be one
- Target probability will be one for the correct label and zero for incorrect labels

*reasonable expectation or quantification of a personal belief

Framework of classification



- We can use a deterministic function, called softmax function, to convert scores to probabilities
- It is a fixed function (no need to train this part)

Softmax function: normalized exponential function

Standard softmax

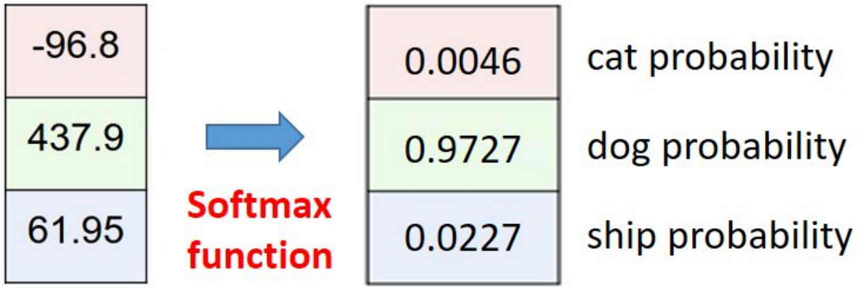
$$\boldsymbol{\sigma}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Softmax with a smoothness parameter

$$\boldsymbol{\sigma}(z_i) = \frac{e^{\beta z_i}}{\sum_{j=1}^{K} e^{\beta z_j}}$$

Vector representation of above

$$\boldsymbol{\sigma}(\vec{z}) = \frac{e^{\beta \vec{z}}}{\sum_{j=1}^{K} e^{\beta z_j}}$$

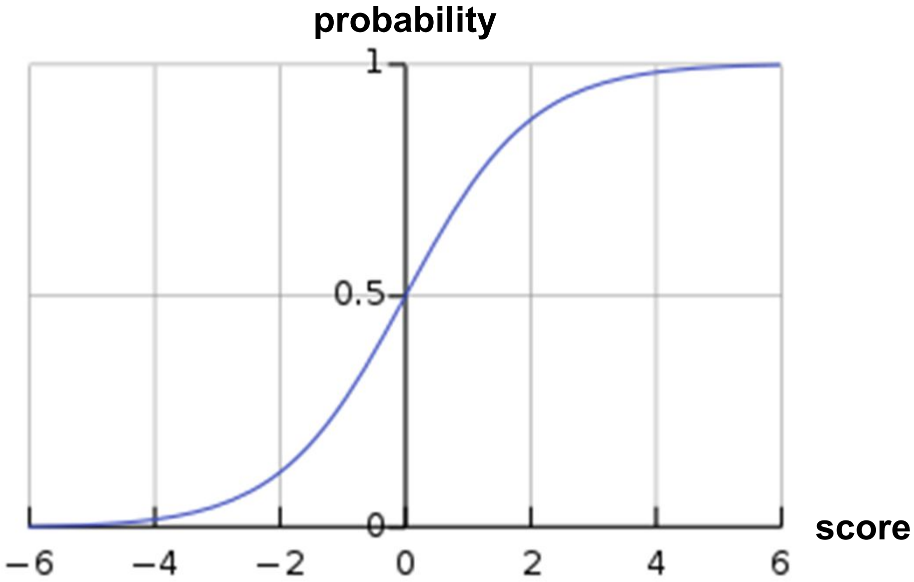


- This softmax function allows us to convert a meaningless set of scores to a set of probabilities (which is a lot more interpretable!)

Softmax function for binary classification (sigmoid)

Sigmoid function

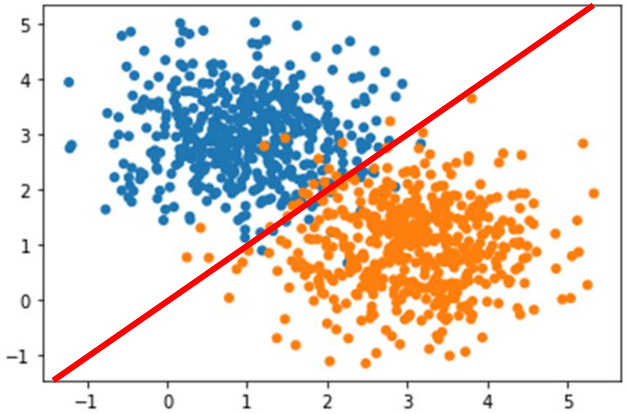
$$S(z) = \frac{e^z}{e^z+1} = \frac{1}{1+e^{-z}}$$



- Sigmoid function
  - is monotonic
  - maps all real values to (0,1) → can be used to represent a probability

What loss function do we want to use?

Such that a loss function $\mathcal{L} : X \times Y \rightarrow \mathcal{R}$ is minimized



- With softmax, our output Y is now a probability, which means **we need a measure to compare two probabilities** (one probability from data and the other one from the classifier output)

Loss for probability measures (binary)

- With softmax, our output is now a probability, which means **we need a measure to quantify the relation between two probabilities** (one probability from data and the other one from the classifier output) to define our loss function

$f\Big($  $\Big) = 0.87$

output of the binary classifier (cat probability)

We want this value to be close to the label Y

Binary cross entropy loss (for binary classification)

- **Binary cross entropy (BCE) loss**

$$\mathcal{L} = -(y\, log(f(x)) + (1-y)log(1-f(x)))$$

  - measures the similarity of two probabilities
  - Consider two cases
    - When $y_i=0$, then $f(x_i)=0$ & $y_i=1$, then $f(x_i)=1$
    - When $y_i=0$, then $f(x_i)=1$ & $y_i=1$, then $f(x_i)=0$

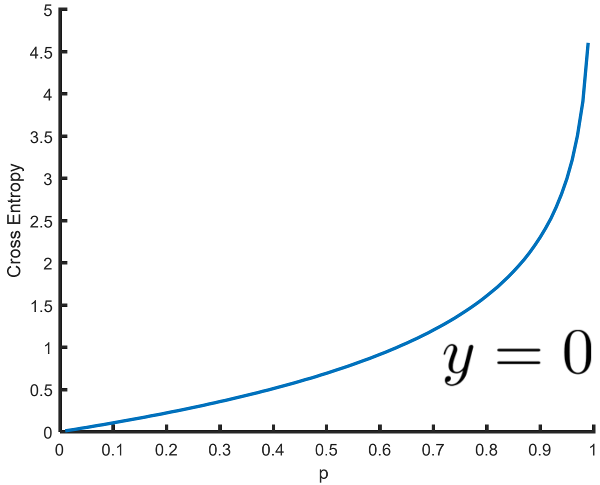
- **BCE loss of multiple data samples**

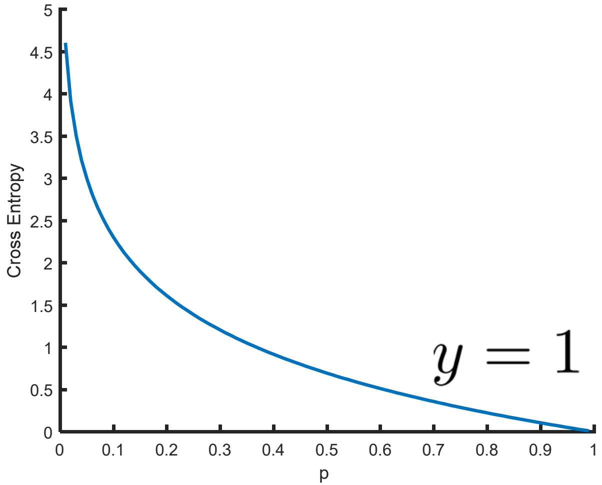
$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}(y_i\, log(f(x_i)) + (1-y_i)log(1-f(x_i)))$$

Binary cross entropy loss

- **Binary cross entropy (BCE) loss**

$$\mathcal{L} = -(y\, log(f(x)) + (1-y) log(1-f(x)))$$



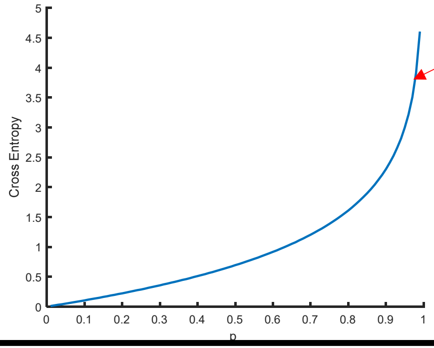


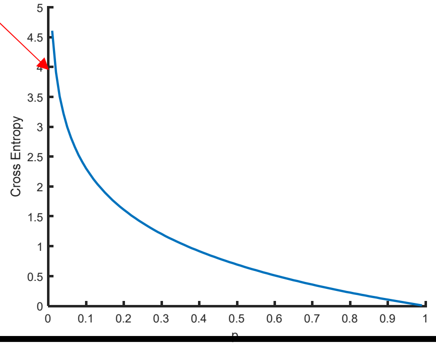
BCE vs. MSE

Q) Can we just use MSE?
A) Yes, but CE is better

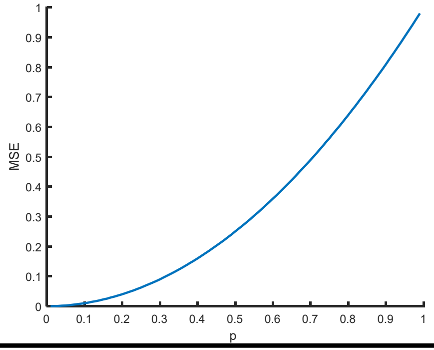
$y = 0$ $$-(y\,logp + (1-y)log(1-p))$$ $y = 1$

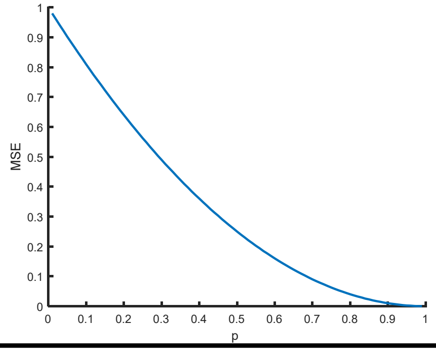
Notice this large slope





$y = 0$ $$(y-p)^2$$ $y = 1$





Cross-entropy loss (for multi-class classification)

- **Cross entropy (CE) loss for K-class classification**

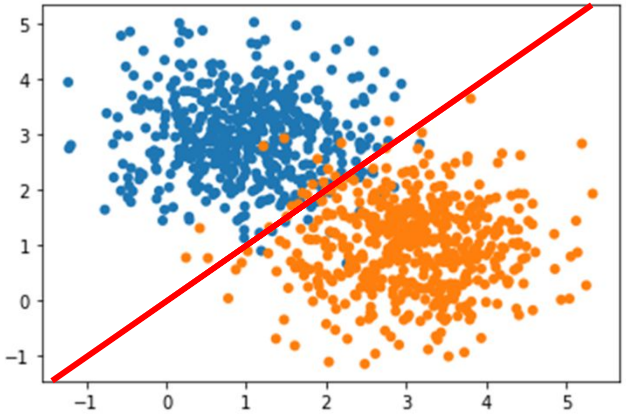
$$\mathcal{L} = -\sum_{j=1}^{K} y_j \, log(f(x)_j)$$

Why is this here?

- **When we have multiple data samples, we take an average**

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij} \, log(f(x_i)_j)$$

Back to our problem: linear classification



For a data set $\mathcal{D} = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), \cdots, (\vec{x_N}, y_N)\}$

Seeks a function $f(\vec{x}; \theta_0, \vec{\theta}) = \boldsymbol{S}(\vec{\theta} \cdot \vec{x} + \theta_0)$

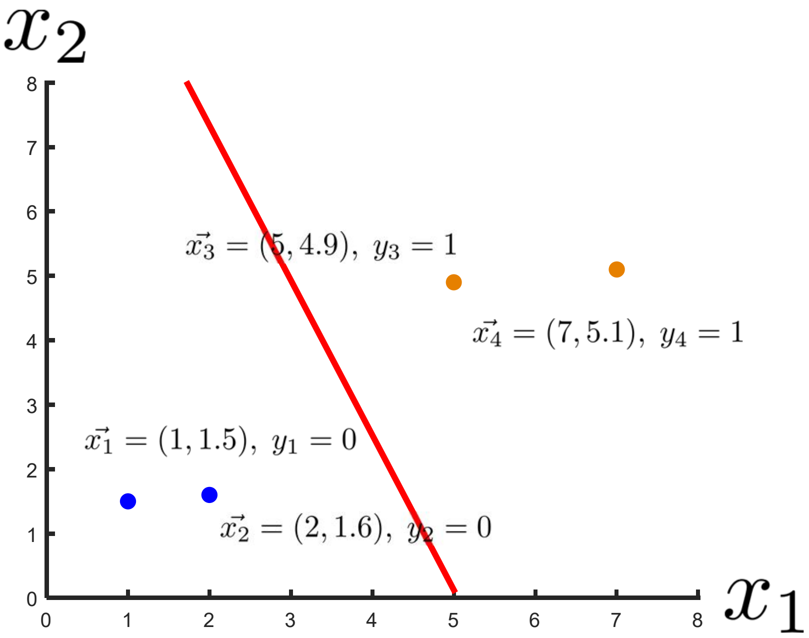
Such that a loss function

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} (y_i \, log(f(x_i)) + (1 - y_i) log(1 - f(x_i)))$$

is minimized

- Now, training our classifier has become a simple problem
  - Just do gradient descent to find the parameters θ and θ_0 that minimize to loss

Simple linear classification



For a data set $\mathcal{D} = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), \cdots, (\vec{x_N}, y_N)\}$

Seeks a function $f(\vec{x}; \theta_0, \vec{\theta}) = \boldsymbol{S}(\vec{\theta} \cdot \vec{x} + \theta_0)$

Such that a loss function

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} (y_i\, log(f(x_i)) + (1 - y_i) log(1 - f(x_i)))$$

is minimized

$$f(x) = \frac{1}{1+e^{-(\vec{\theta}\cdot\vec{x}+\theta_0)}}$$

$$\vec{x_1} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \end{bmatrix}$$

sample index

axis

Gradient calculation for simple linear classification

$$\mathcal{L} = -(y\,log(f(x)) + (1-y)log(1-f(x)))$$

$$f(x) = \hat{y} = \frac{1}{1+e^{-(\theta_2 x_2 + \theta_1 x_1 + \theta_0)}}$$

$$\mathcal{L} = -(y\,log\hat{y} + (1-y)log(1-\hat{y}))$$

$$\hat{y} = \frac{1}{1+e^{-(\theta_2 x_2 + \theta_1 x_1 + \theta_0)}} = \boldsymbol{S}(z)$$

$$\boldsymbol{S}(z) = \frac{e^z}{e^z+1} = \frac{1}{1+e^{-z}}$$

$$z = \vec{\theta} \cdot \vec{x} + \theta_0 = \theta_2 x_2 + \theta_1 x_1 + \theta_0$$

These are NOT sample indices

Gradient calculation for simple linear classification

$$\mathcal{L} = -(y\,log\hat{y} + (1-y)log(1-\hat{y}))$$

$$\hat{y} = \frac{1}{1+e^{-(\theta_2 x_2 + \theta_1 x_1 + \theta_0)}} = \boldsymbol{S}(z)$$

$$\boldsymbol{S}(z) = \frac{e^z}{e^z+1} = \frac{1}{1+e^{-z}}$$

$$z = \theta_2 x_2 + \theta_1 x_1 + \theta_0$$

$$\nabla_\theta \mathcal{L}(\theta) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \theta_0} \\ \frac{\partial \mathcal{L}}{\partial \theta_1} \\ \frac{\partial \mathcal{L}}{\partial \theta_2} \end{bmatrix}$$

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial \theta_0}$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial \theta_1}$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial \theta_2}$$

Gradient calculation for simple linear classification

$$\mathcal{L} = -(y\, log\hat{y} + (1-y)log(1-\hat{y}))$$

$$\hat{y} = \frac{1}{1+e^{-(\theta_2 x_2 + \theta_1 x_1 + \theta_0)}} = \boldsymbol{S}(z)$$

$$\boldsymbol{S}(z) = \frac{e^z}{e^z+1} = \frac{1}{1+e^{-z}}$$

$$z = \theta_2 x_2 + \theta_1 x_1 + \theta_0$$

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = -\frac{\partial}{\partial \hat{y}}(y\, log\hat{y} + (1-y)log(1-\hat{y})) = -(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}) = -(\frac{y-\hat{y}}{\hat{y}(1-\hat{y})})$$

$$\frac{\partial \hat{y}}{\partial z} = (\frac{1}{1+e^{-z}})(\frac{-e^{-z}}{1+e^{-z}}) = \hat{y}(1-\hat{y})$$

$$\frac{\partial z}{\partial \theta_0} = 1 \qquad \frac{\partial z}{\partial \theta_1} = x_1 \qquad \frac{\partial z}{\partial \theta_2} = x_2$$

Gradient calculation for simple linear classification

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = -\frac{\partial}{\partial \hat{y}}(y\,log\hat{y} + (1-y)log(1-\hat{y})) = -(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}) = -(\frac{y-\hat{y}}{\hat{y}(1-\hat{y})})$$

$$\frac{\partial \hat{y}}{\partial z} = (\frac{1}{1+e^{-z}})(\frac{-e^{-z}}{1+e^{-z}}) = \hat{y}(1-\hat{y})$$

$$\frac{\partial z}{\partial \theta_0} = 1 \qquad \frac{\partial z}{\partial \theta_1} = x_1 \qquad \frac{\partial z}{\partial \theta_2} = x_2$$

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{\partial \mathcal{L}}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z}\frac{\partial z}{\partial \theta_0} = \hat{y} - y$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z}\frac{\partial z}{\partial \theta_1} = (\hat{y} - y)x_1$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = \frac{\partial \mathcal{L}}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z}\frac{\partial z}{\partial \theta_2} = (\hat{y} - y)x_2$$

Gradient calculation for simple linear classification

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial \theta_0} = \hat{y} - y$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial \theta_1} = (\hat{y} - y)x_1$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial \theta_2} = (\hat{y} - y)x_2$$

$$\vec{x_1} = (1, 1.5),\ y_1 = 0$$

$$\vec{x_2} = (2, 1.6),\ y_2 = 0$$

$$\vec{x_3} = (5, 4.9),\ y_3 = 1$$

$$\vec{x_4} = (7, 5.1),\ y_4 = 1$$

- Let’s start with zero parameters → $\theta_0 = \theta_1 = \theta_2 = 0$

$$f(\vec{x_1}) = \frac{1}{1+e^{-(0*1+0*1.5+0)}} = \frac{1}{1+e^{-0}} = 0.5$$

$$f(\vec{x_2}) = \frac{1}{1+e^{-(0*2+0*1.6+0)}} = \frac{1}{1+e^{-0}} = 0.5$$

$$f(\vec{x_1}) = \frac{1}{1+e^{-(0*5+0*4.9+0)}} = \frac{1}{1+e^{-0}} = 0.5$$

$$f(\vec{x_1}) = \frac{1}{1+e^{-(0*7+0*5.1+0)}} = \frac{1}{1+e^{-0}} = 0.5$$

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{1}{4}\sum_{i=1}^{4} \hat{y_i} - y_i = \frac{1}{4}((0.5-0) + (0.5-0) + (0.5-1) + (0.5-1)) = 0$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{1}{4}\sum_{i=1}^{4} (\hat{y_i} - y_i)x_1^{(i)}$$

$$= \frac{1}{4}((0.5-0)1 + (0.5-0)2 + (0.5-1)5 + (0.5-1)7) = -1.125$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = \frac{1}{4}\sum_{i=1}^{4} (\hat{y_i} - y_i)x_2^{(i)}$$

$$= \frac{1}{4}((0.5-0)1.5 + (0.5-0)1.6 + (0.5-1)4.9 + (0.5-1)5.1) = -0.8625$$

Gradient Descent for linear classification

1. Apply chain rule for gradient calculation

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial \theta_0} = \hat{y} - y$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial \theta_1} = (\hat{y} - y)x_1$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial \theta_2} = (\hat{y} - y)x_2$$

2. Forward pass

$$f(x) = \hat{y} = \frac{1}{1+e^{-(\theta_2 x_2 + \theta_1 x_1 + \theta_0)}}$$

Gradient Descent for linear classification

3. Gradient calculation

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{1}{4}\sum_{i=1}^{4} \hat{y_i} - y_i = \frac{1}{4}((0.5-0) + (0.5-0) + (0.5-1) + (0.5-1)) = 0$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{1}{4}\sum_{i=1}^{4} (\hat{y_i} - y_i)x_1^{(i)}$$

$$= \frac{1}{4}((0.5-0)1 + (0.5-0)2 + (0.5-1)5 + (0.5-1)7) = -1.125$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = \frac{1}{4}\sum_{i=1}^{4} (\hat{y_i} - y_i)x_2^{(i)}$$

$$= \frac{1}{4}((0.5-0)1.5 + (0.5-0)1.6 + (0.5-1)4.9 + (0.5-1)5.1) = -0.8625$$

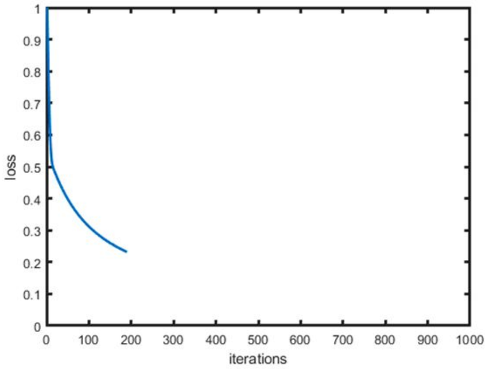
4. Parameter updates

$$\begin{bmatrix} \theta_0^{(k+1)} \\ \theta_1^{(k+1)} \\ \theta_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} \theta_0^{(k)} \\ \theta_1^{(k)} \\ \theta_2^{(k)} \end{bmatrix} - \gamma \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \theta_0} \\ \frac{\partial \mathcal{L}}{\partial \theta_1} \\ \frac{\partial \mathcal{L}}{\partial \theta_2} \end{bmatrix}$$

5. Repeat 2-4

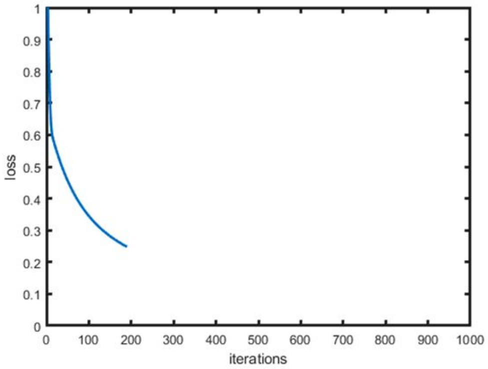
Gradient Descent for linear classification

$\mathcal{L}$

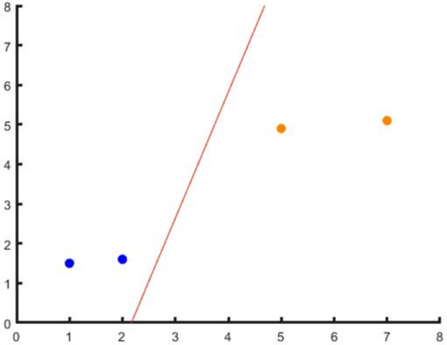


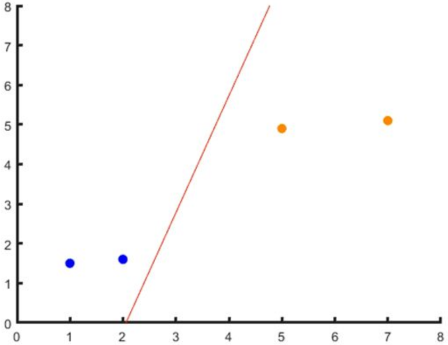
k (iterations)

$\mathcal{L}$



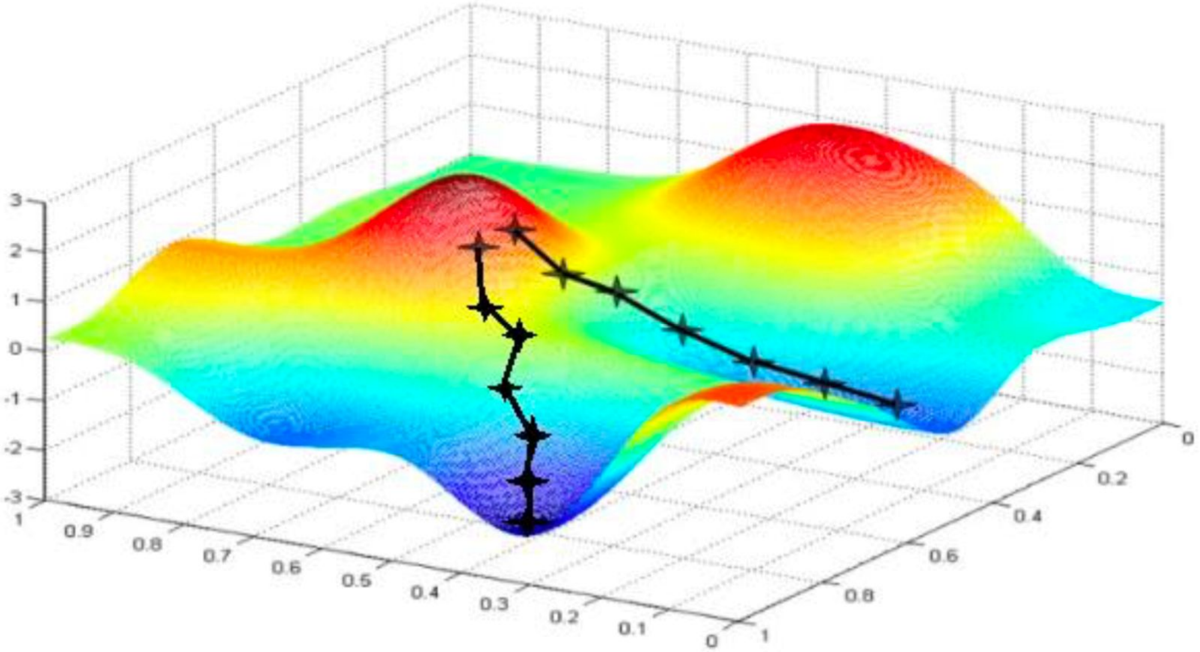
k (iterations)





Two trials with different initial parameters

Gradient Descent for linear classification



Two trials with different initial parameters

Summary

- We can formulate classification task as a supervised learning problem
- We can use softmax and sigmoid functions to convert “scores” to probabilities
- We can use cross entropy as our loss function (to measure the similarity of two probabilities distributions)
- We can use gradient descent to train a linear classifier

References

- Lecture notes
  - CC229 lecture note
    - http://cs229.stanford.edu/notes2020fall/notes2020fall/cs229-notes1.pdf
  - MIT 6.036 Intro to Machine Learning (Chapter 2)
    - https://www.mit.edu/~lindrew/6.036.pdf

- Website
  - CS231n course website: https://cs231n.github.io/