# DS 4400

## Machine Learning and Data Mining I

Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

October 8 2020

# Announcements

- Released solutions for Homework 1
- Homework 2 is due on Tuesday, Oct. 13 at midnight
  - University holiday on Monday, Oct. 12
- Projects
  - Start thinking about theme, dataset, and topic
  - Look at shared resources and project examples in Piazza
  - Fill in a form with area preferences on Friday
  - Participate in discussion next week

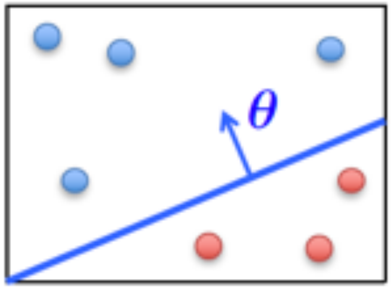
# Outline

- Linear classifiers
- Perceptron wrap up
- Logistic regression
  - Classification based on probability
- Maximum Likelihood Estimation
  - Application to logistic regression
  - Cross-entropy objective
- Gradient descent for logistic regression

# Linear Classifiers

- **Linear classifiers**: represent decision boundary by hyperplane

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \boldsymbol{x}^\mathsf{T} = \begin{bmatrix} 1 & x_1 & \dots & x_d \end{bmatrix}$$

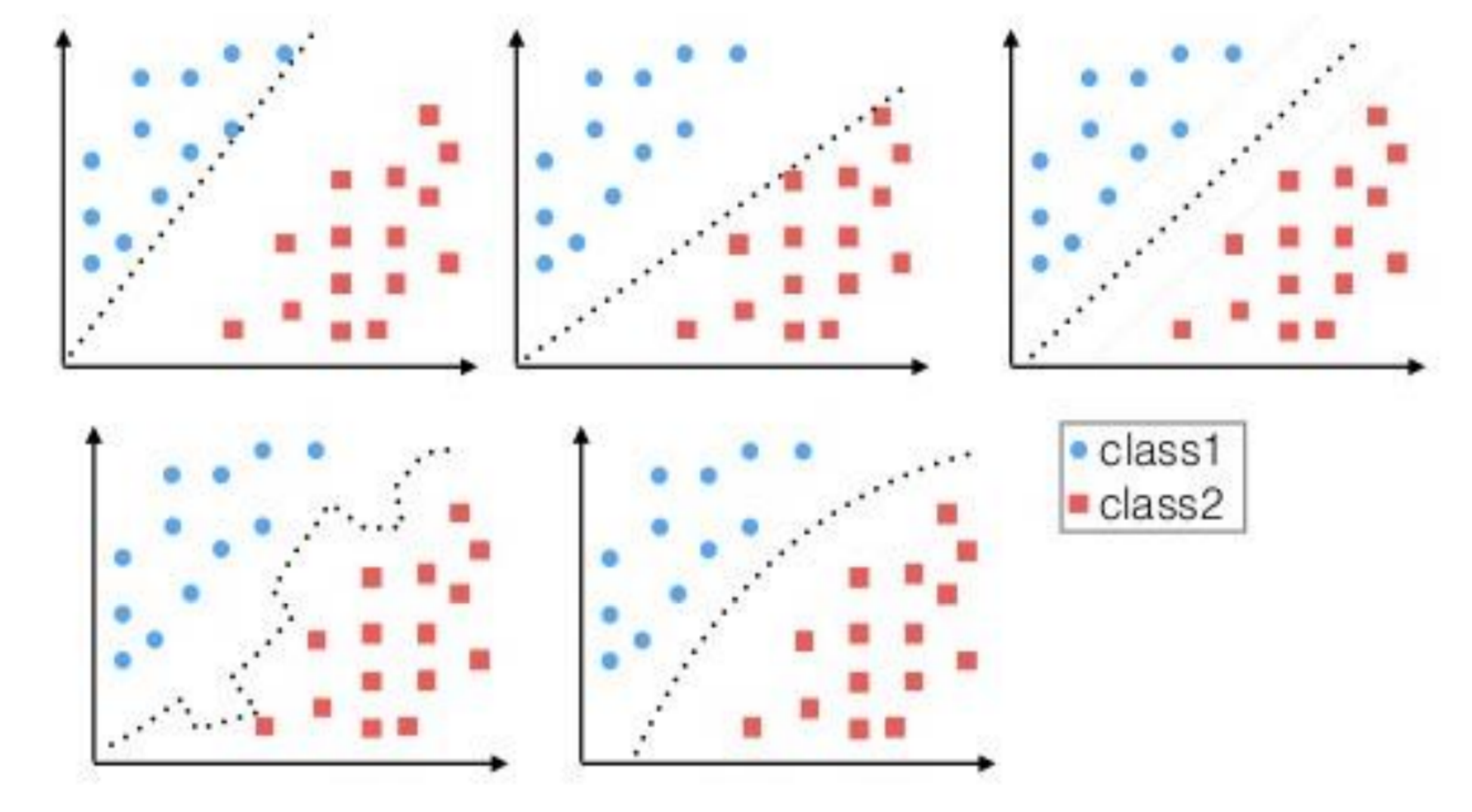



$h_\theta(x) = f(\theta^T x)$ linear function

- If $\theta^T x > 0$ classify "Class A"
- If $\theta^T x < 0$ classify "Class B"

All the points x on the hyperplane satisfy: $\theta^T x = 0$

# Linear vs Non-Linear Classifiers



# Online Perceptron

Let $\theta \leftarrow$[0,0,…,0]
Repeat:
    Receive training example $(x_i, y_i)$
    If $y_i \theta^T x_i \leq 0$ // prediction is incorrect
        $\theta \leftarrow \theta + y_i\, x_i$

**Online learning** – the learning mode where the model update is performed each time a single observation is received

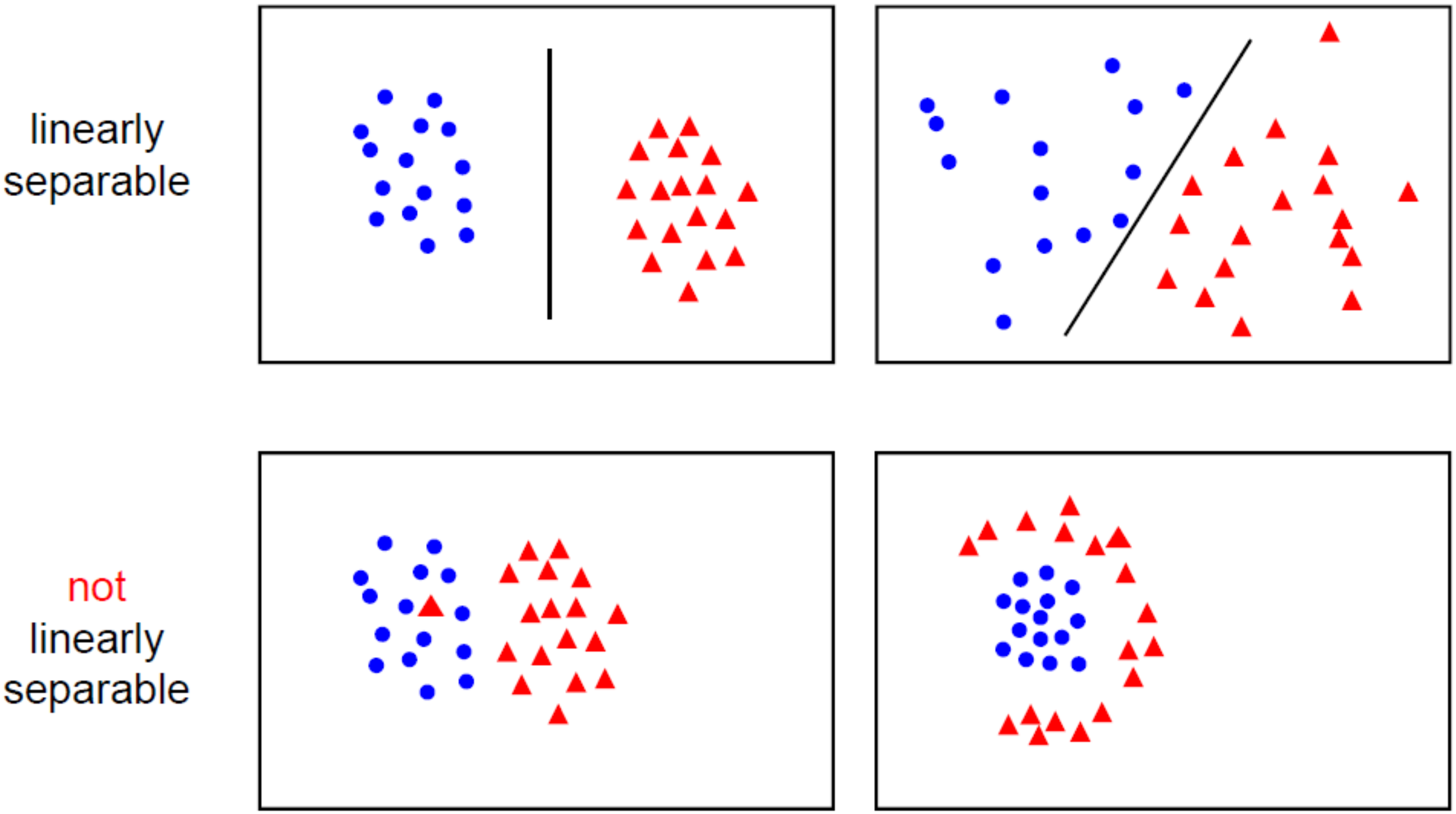
**Batch learning** – the learning mode where the model update is performed after observing the entire training set

# Batch Perceptron

Given training data $\{(x_i, y_i)\}_{i=1}^{n}$
Let $\boldsymbol{\theta} \leftarrow [0, 0, \ldots, 0]$
Repeat:
    Let $\boldsymbol{\Delta} \leftarrow [0, 0, \ldots, 0]$
    for $i = 1 \ldots n$, do
        if $y_i \theta^T x_i \leq 0$    // prediction for i$^{th}$ instance is incorrect
            $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta} + y_i\, x_i$
    $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta}/n$    // compute average update
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\Delta}$
Until $\|\boldsymbol{\Delta}\|_2 < \epsilon$

Guaranteed to find separating hyperplane if data is linearly separable

# Linear separability




- For linearly separable data, can prove bounds on perceptron error (depends on how well separated the data is)

# The perceptron

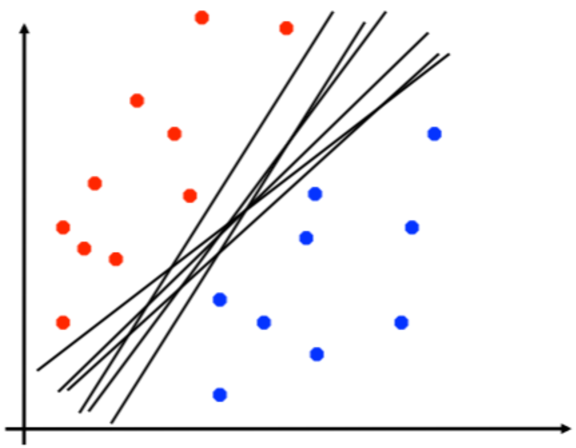
$$h_\theta(x) = f(\theta^T x)$$

- Linear classifier
- $f$ is the sign function for the perceptron
- Pros
  - Very compact model (size *d*)

- Cons of the perceptron
  - Perceptron depends on the order of training data and it could take many steps for convergence
  - Only classifies well data that is linearly separable

# Perceptron Limitations

- Is dependent on starting point
- It could take many steps for convergence
- Perceptron can overfit
  - Move the decision boundary for every example



Which of this is optimal?

# Improving the Perceptron

- The Perceptron produces many $\theta$'s during training
- The standard Perceptron simply uses the final $\theta$ at test time
  - This may sometimes not be a good idea!
  - Some other $\theta$ may be correct on 1,000 consecutive examples, but one mistake ruins it!

- **Idea:** Use a combination of multiple perceptrons
  - (i.e., neural networks!)
- **Idea:** Use the intermediate $\theta$'s
  - **Voted Perceptron**: vote on predictions of the intermediate $\theta$'s
  - **Averaged Perceptron**: average the intermediate $\theta$'s

# Classification Based on Probability

- Instead of just predicting the class, give the *probability of the instance being in that class*
  - Learn $P(Y|X)$
- Consider binary classifier with classes 0 and 1
  - $P(Y = 1|X) + P(Y = 0|X) = 1$
  - Sufficient to learn $P(Y = 1|X)$
- Advantages: interpretability and confidence of output

# Logistic Regression

- Setup
  - Training data: $\{x_i, y_i\}$, for $i = 1, \ldots, N$
  - Labels: $y_i \in \{0,1\}$
- Goals
  - Learn $P(Y = 1|X = x)$
- Highlights
  - Probabilistic output
  - At the basis of more complex models (e.g., neural networks)
  - Supports regularization (Ridge, Lasso)
  - Can be trained with Gradient Descent

# Interpretation of Model Output

$h_{\boldsymbol{\theta}}(\boldsymbol{x})$ = estimated $P(Y = 1|X; \theta)$

Example: Cancer diagnosis from tumor size

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0.7$$

→ Tell patient that 70% chance of tumor being malignant

Note that: $P(Y = 0|X; \theta) + P(Y = 1|X; \theta) = 1$

Therefore, $P(Y = 0|X; \theta) = 1 - P(Y = 1|X; \theta)$

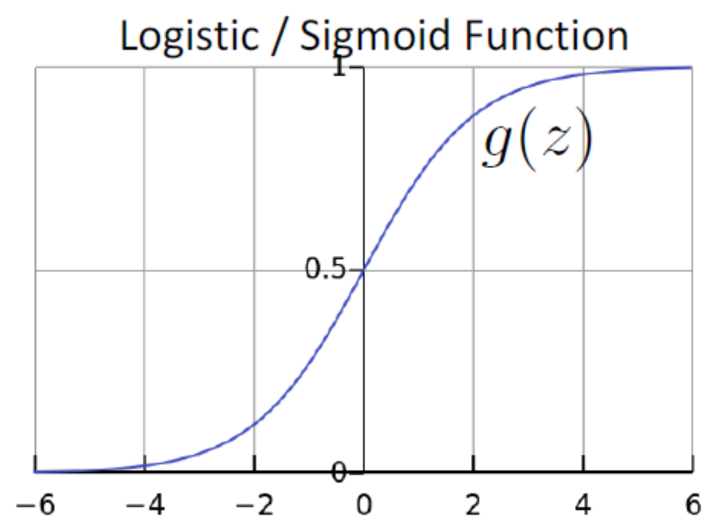
# Logistic Regression

- Takes a probabilistic approach to learning discriminative functions (i.e., a classifier)
- $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ should give $P(Y = 1|X; \theta)$
  - Want $0 \leq h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 1$
- Logistic regression model:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

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

$$\boxed{h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}}}}$$



# LR is a Linear Classifier!

- Predict $Y = 1$ if:
  - $\mathrm{P}[Y = 1 | X = x; \theta] > \mathrm{P}[Y = 0 | X = x; \theta]$
  - $\mathrm{P}[Y = 1 | X = x; \theta] > \frac{1}{2}$

$$\frac{1}{1 + e^{-\theta^T x}} > \frac{1}{2}$$

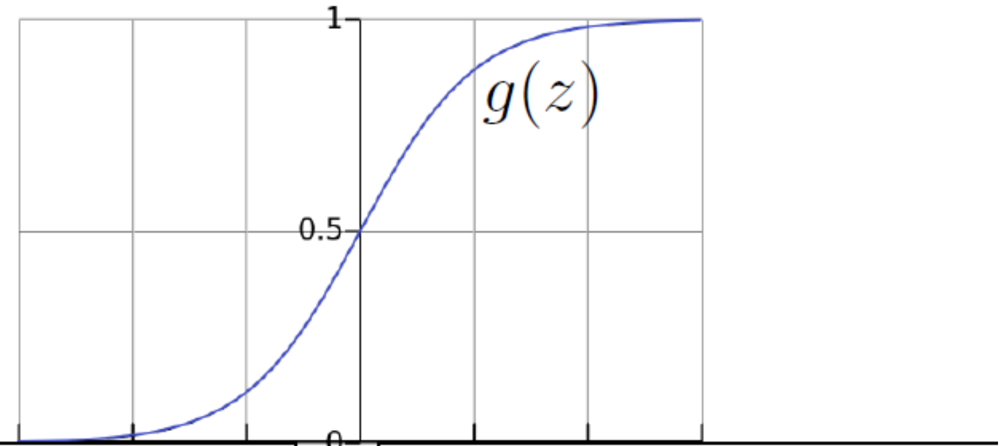
- Equivalent to:
  - $e^{\theta_0 + \sum_{j=1}^{d} \theta_j x_j} > 1$
  - $\theta_0 + \sum_{j=1}^{d} \theta_j x_j > 0$

Logistic Regression is a linear classifier!

# Logistic Regression

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

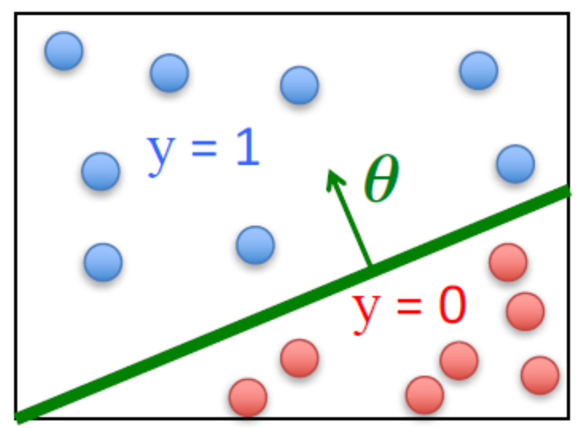
$$g(z) = \frac{1}{1 + e^{-z}}$$




$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large <u>negative</u> values for negative instances

$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large <u>positive</u> values for positive instances

- Assume a threshold and...
  - Predict $Y = 1$ if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5$
  - Predict $Y = 0$ if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$




**Logistic Regression is a linear classifier!**

# How to Pick Loss Function?

# Maximum Likelihood Estimation (MLE)

Given training data $\boldsymbol{X} = \{x_1, \ldots, x_N\}$ with labels $\boldsymbol{Y} = \{y_1, \ldots, y_N\}$

What is the likelihood of training data for parameter $\theta$?

Define likelihood function

$$Max_\theta \, L(\theta) = P[\boldsymbol{Y}|\boldsymbol{X}; \theta]$$

Assumption: training points are independent

$$L(\theta) = \prod_{i=1}^{N} P[Y = y_i | X = x_i; \theta]$$

General probabilistic method for classifier training

# Log Likelihood

- Max likelihood is equivalent to maximizing log of likelihood

$$L(\theta) = \prod_{i=1}^{N} P[Y = y_i | X = x_i; \theta]$$

$$\log L(\theta) = \sum_{i=1}^{N} \log P[Y = y_i | X = x_i; \theta]$$

- They both have the same maximum $\theta_{MLE}$

# MLE for Logistic Regression

$$\mathrm{P}(Y = y_i | X = x_i; \theta) = h_\theta(x_i)^{y_i} \left(1 - h_\theta(x_i)\right)^{1-y_i}$$

$$\theta_{MLE} = \mathrm{argmax}_\theta \sum_{i=1}^{N} \log P[Y = y_i | X = x_i; \theta]$$

$$= \mathrm{argmax}_\theta \sum_{i=1}^{N} y_\mathrm{i} \log h_\theta(x_i) + (1 - y_i) \log\left(1 - h_\theta(x_i)\right)$$

Logistic regression objective

$$\min_\theta J(\theta)$$

$$J(\theta) = -\sum_{i=1}^{N} [y_\mathrm{i} \log h_\theta(x_i) + (1 - y_i) \log\left(1 - h_\theta(x_i)\right)]$$

# Cross-Entropy Objective

$$J(\theta) = -\sum_{i=1}^{N}[y_{\mathrm{i}}\log h_\theta(x_i) + (1 - y_i)\log\left(1 - h_\theta(x_i)\right)]$$

- Cost of a single instance:

$$\text{cost}\ (h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

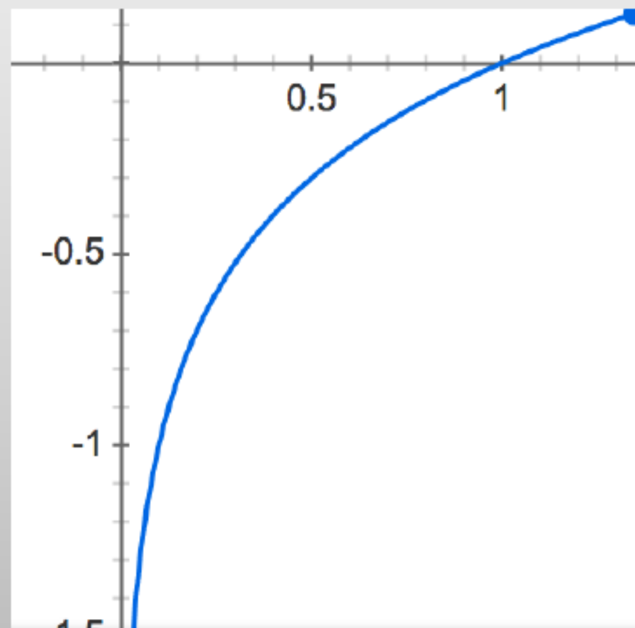
- Can re-write objective function as

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \underbrace{\text{cost}\left( h_\theta(x_i), y_i \right)}_{\text{Cross-entropy loss}}$$

# Intuition

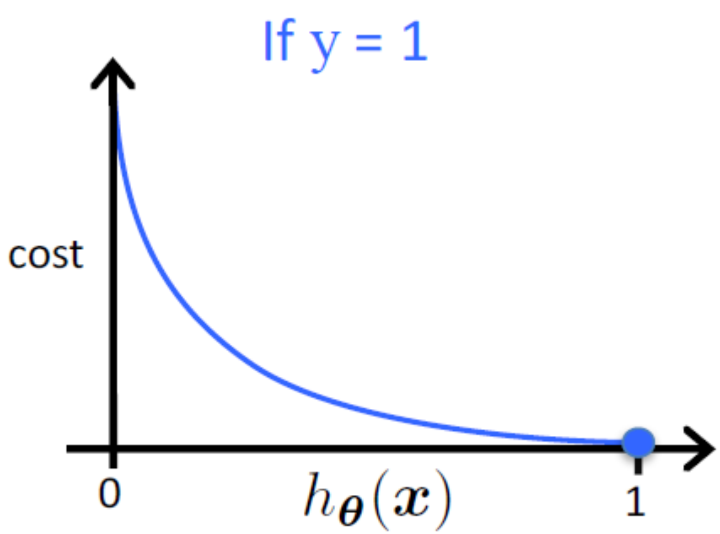
$$\text{cost}\,(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

Aside: Recall the plot of log(z)



# Intuition

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} \boxed{-\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 1} \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 0 \end{cases}$$

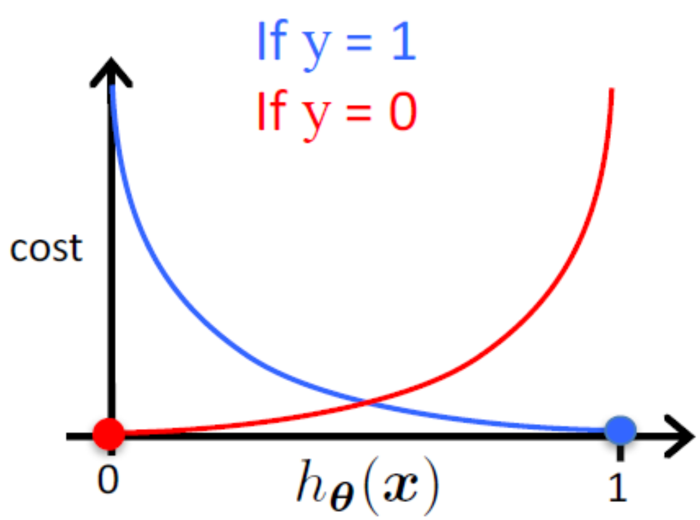



If y = 1

- Cost = 0 if prediction is correct
- As $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \rightarrow 0, \text{cost} \rightarrow \infty$
- Captures intuition that larger mistakes should get larger penalties
  - e.g., predict $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0$, but y = 1

# Intuition

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} \boxed{-\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 1} \\ \boxed{-\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 0} \end{cases}$$




If y = 0

- Cost = 0 if prediction is correct
- As $(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \to 0, \text{cost} \to \infty$
- Captures intuition that larger mistakes should get larger penalties

# Cross-Entropy Objective

$$\mathrm{P}(Y = y_i | X = x_i; \theta) = h_\theta(x_i)^{y_i}\big(1 - h_\theta(x_i)\big)^{1-y_i}$$

$$\theta_{MLE} = \mathrm{argmax}_\theta \textstyle\sum_{i=1}^{N} \log P[Y = y_i | X = x_i; \theta]$$
$$= \mathrm{argmax}_\theta \sum_{i=1}^{N} y_i \log h_\theta(x_i) + (1 - y_i) \log\big(1 - h_\theta(x_i)\big)$$

Logistic regression objective

$$\min_\theta J(\theta)$$

$$J(\theta) = -\sum_{i=1}^{N} \left[y_i \log h_\theta(x_i) + (1 - y_i) \log\big(1 - h_\theta(x_i)\big)\right]$$

# Logistic Regression
# Lab Example

# Review

- Perceptron is the first example of linear classifier
  - Online and batch learning
  - Has several limitations
- Logistic regression is a linear classifier that predicts class probability
- Maximum Likelihood Estimation is a method to estimate model parameters
  - Derive cross-entropy loss function
- Logistic regression can be trained with GD

# Acknowledgements


- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
- Thanks!