10-301/601: Introduction to Machine Learning Lecture 9 – Logistic Regression

Henry Chai & Matt Gormley

9/28/22

Q & A:

Where does the $O(n^3)$ computational complexity for matrix inversion come from?

- And that other number you quoted for that matter?

Matrix inversion	One $n \times n$ matrix	One $n \times n$ matrix	Gauss–Jordan elimination	$O(n^3)$
			Strassen algorithm	$O(n^{2.807})$
			Coppersmith–Winograd algorithm	$O(n^{2.376})$
			Optimized CW-like algorithms	$O(n^{2.373})$

Source: https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations

Front Matter

- Announcements:
  - HW3 released 9/21, due 9/28 (today!) at 11:59 PM
    - **Only two grace days allowed on HW3**
    - HW3 exit poll has also been released: you have until one week from the due date to complete it
  - Exam 1 on 10/4 (one week from yesterday) from 6:30 PM - 8:30 PM
    - Exam 1 practice problems released on the course website, under Coursework
  - Lecture instead of recitation this Friday (9/30)

Probabilistic Learning

- Previously:
  - (Unknown) Target function, $c^*: \mathcal{X} \rightarrow \mathcal{Y}$
  - Classifier, $h : \mathcal{X} \rightarrow \mathcal{Y}$
  - Goal: find a classifier, h, that best approximates c^*
- Now:
  - (Unknown) Target *distribution*, $y \sim p^*(Y|\boldsymbol{x})$
  - Distribution, $p(Y|\boldsymbol{x})$
  - Goal: find a distribution, p, that best approximates p^*

Likelihood

- Given N independent, identically distribution (iid) samples $\mathcal{D} = \{x^{(1)}, \ldots, x^{(N)}\}$ of a random variable X
  - If X is discrete with probability mass function (pmf) $p(X|\theta)$, then the *likelihood* of $\mathcal{D}$ is

$$L(\theta) = \prod_{i=1}^{N} p\left(x^{(i)}|\theta\right)$$

  - If X is continuous with probability density function (pdf) $f(X|\theta)$, then the *likelihood* of $\mathcal{D}$ is

$$L(\theta) = \prod_{i=1}^{N} f\left(x^{(i)}|\theta\right)$$

Log-Likelihood

- Given N independent, identically distribution (iid) samples $\mathcal{D} = \{x^{(1)}, \ldots, x^{(N)}\}$ of a random variable X
  - If X is discrete with probability mass function (pmf) $p(X|\theta)$, then the *log-likelihood* of $\mathcal{D}$ is

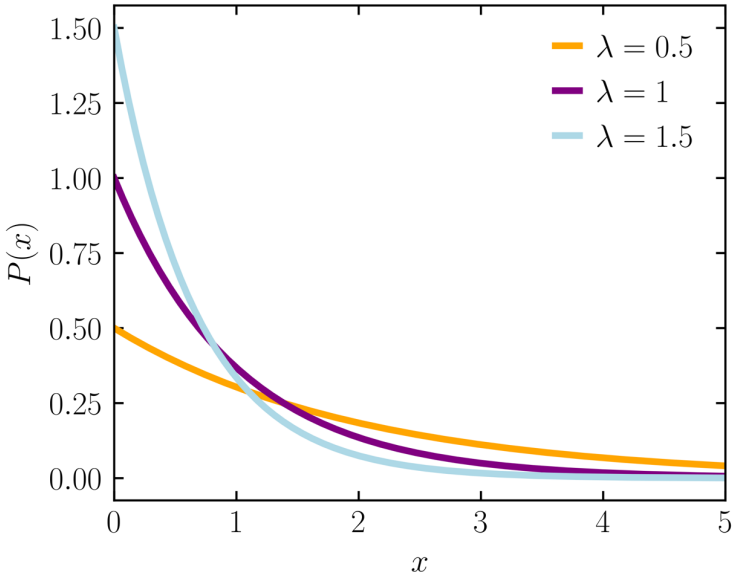
$$\ell(\theta) = \log \prod_{i=1}^{N} p\left(x^{(i)}|\theta\right) = \sum_{i=1}^{N} \log p\left(x^{(i)}|\theta\right)$$

  - If X is continuous with probability density function (pdf) $f(X|\theta)$, then the *log-likelihood* of $\mathcal{D}$ is

$$\ell(\theta) = \log \prod_{i=1}^{N} f\left(x^{(i)}|\theta\right) = \sum_{i=1}^{N} \log f\left(x^{(i)}|\theta\right)$$

Maximum Likelihood Estimation (MLE)

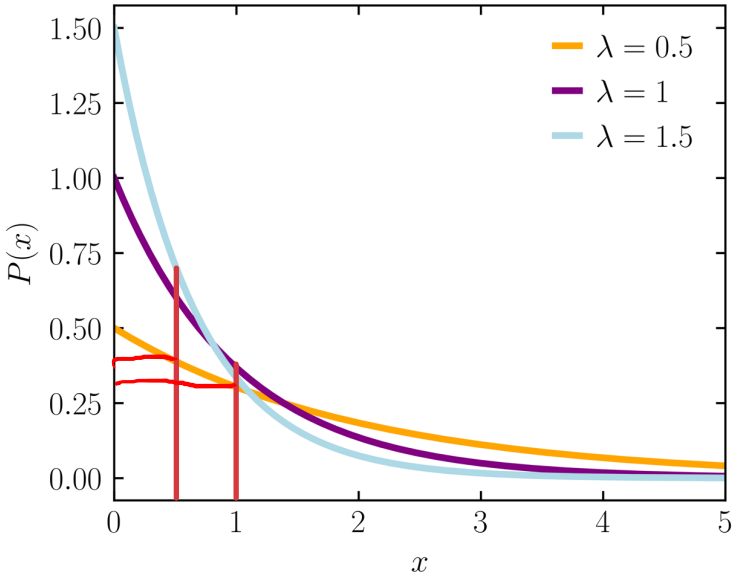
- Insight: every valid probability distribution has a finite amount of probability mass as it must sum/integrate to 1
- Idea: set the parameter(s) so that the likelihood of the samples is maximized
- Intuition: assign as much of the (finite) probability mass to the observed data *at the expense of unobserved data*
- Example: the exponential distribution



Source: https://en.wikipedia.org/wiki/Exponential_distribution#/media/File:Exponential_probability_density.svg

Maximum Likelihood Estimation (MLE)

- Insight: every valid probability distribution has a finite amount of probability mass as it must sum/integrate to 1
- Idea: set the parameter(s) so that the likelihood of the samples is maximized
- Intuition: assign as much of the (finite) probability mass to the observed data *at the expense of unobserved data*
- Example: the exponential distribution

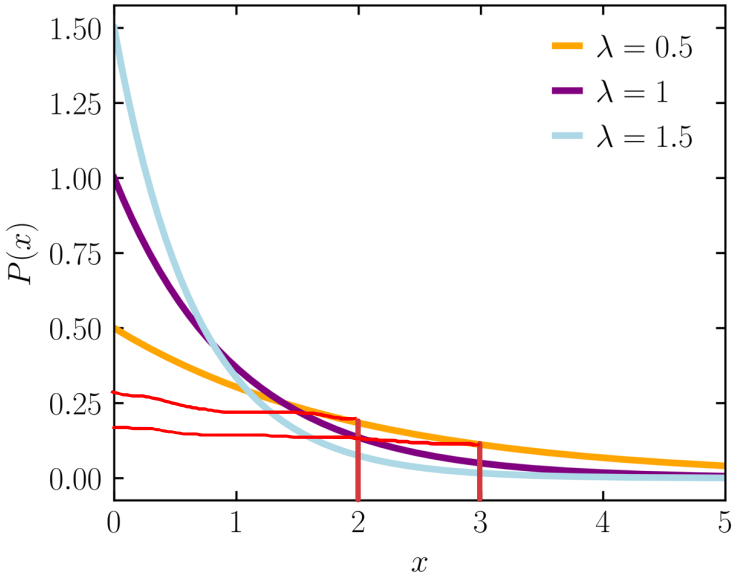


$\{x^{(1)} = 0.5, x^{(2)} = 1\}$

Source: https://en.wikipedia.org/wiki/Exponential_distribution#/media/File:Exponential_probability_density.svg

Maximum Likelihood Estimation (MLE)

- Insight: every valid probability distribution has a finite amount of probability mass as it must sum/integrate to 1
- Idea: set the parameter(s) so that the likelihood of the samples is maximized
- Intuition: assign as much of the (finite) probability mass to the observed data *at the expense of unobserved data*
- Example: the exponential distribution



$\{x^{(1)} = 2, x^{(2)} = 3\}$

Source: https://en.wikipedia.org/wiki/Exponential_distribution#/media/File:Exponential_probability_density.svg

Exponential Distribution MLE

- The pdf of the exponential distribution is

$$f(x|\lambda) = \lambda e^{-\lambda x}$$

- Given N iid samples $\{x^{(1)}, \ldots, x^{(N)}\}$, the likelihood is

$$L(\lambda) = \prod_{i=1}^{N} f\left(x^{(i)}|\lambda\right) = \prod_{i=1}^{N} \lambda e^{-\lambda x^{(i)}}$$

Okay, but now what?

How do we actually find the best value for the parameter, λ?

Exponential Distribution MLE

- The pdf of the exponential distribution is

$$f(x|\lambda) = \lambda e^{-\lambda x}$$

- Given N iid samples $\{x^{(1)}, \ldots, x^{(N)}\}$, the log-likelihood is

$$\ell(\lambda) = \sum_{i=1}^{N} \log f\left(x^{(i)}|\lambda\right) = \sum_{i=1}^{N} \log \lambda e^{-\lambda x^{(i)}}$$

$$= \sum_{i=1}^{N} \left(\log \lambda + \log e^{-\lambda x^{(i)}}\right) = \sum_{i=1}^{N} \log \lambda - \lambda x^{(i)}$$

Take the partial derivative, set equal to 0 and solve

$$\frac{\partial \ell}{\partial \lambda} = \sum_{i=1}^{N} \frac{1}{\lambda} - x^{(i)} = \frac{N}{\lambda} - \sum_{i=1}^{N} x^{(i)}$$

$$\Rightarrow \frac{N}{\hat{\lambda}} - \sum_{i=1}^{N} x^{(i)} = 0 \Rightarrow \frac{N}{\hat{\lambda}} = \sum_{i=1}^{N} x^{(i)} \Rightarrow \hat{\lambda} = \frac{N}{\sum_{i=1}^{N} x^{(i)}}$$

Building a Probabilistic Classifier

- Define a decision rule
  - Given a test data point $\boldsymbol{x}'$, predict its label $\hat{y}$ using the posterior distribution $P(Y = y|\boldsymbol{x}')$
  - Common choice: $\hat{y} = \underset{y}{\operatorname{argmax}}\, P(Y = y|\boldsymbol{x}')$
- Idea: model $P(Y|\boldsymbol{x})$ as some parametric function of $\boldsymbol{x}$

Modelling the Posterior

- Suppose we have binary labels $y \in \{0,1\}$ and D-dimensional inputs $\boldsymbol{x} = [1, x_1, \ldots, x_D]^T \in \mathbb{R}^{D+1}$

  1 prepended to $\boldsymbol{x}$

- **Assume**

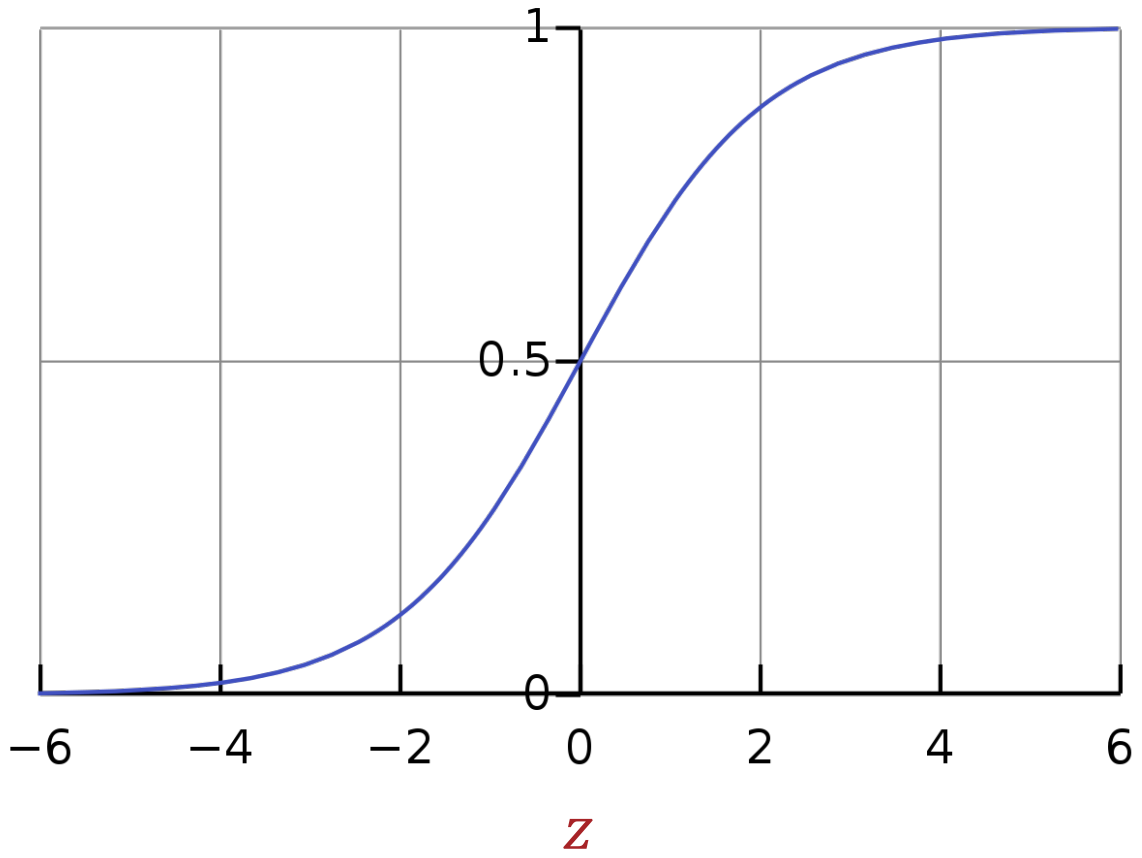
$$P(Y=1 \mid x, \theta) = \sigma(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)} = \frac{\exp(\theta^T x)}{\exp(\theta^T x)+1}$$

This implies 2 things

1. $P(Y=0 \mid x, \theta) = 1 - P(Y=1 \mid x, \theta) = \frac{1}{\exp(\theta^T x)+1}$
2. $\frac{P(Y=1 \mid x, \theta)}{P(Y=0 \mid x, \theta)} = \exp(\theta^T x) \Rightarrow \log \text{odds} = \theta^T x$

Logistic Function

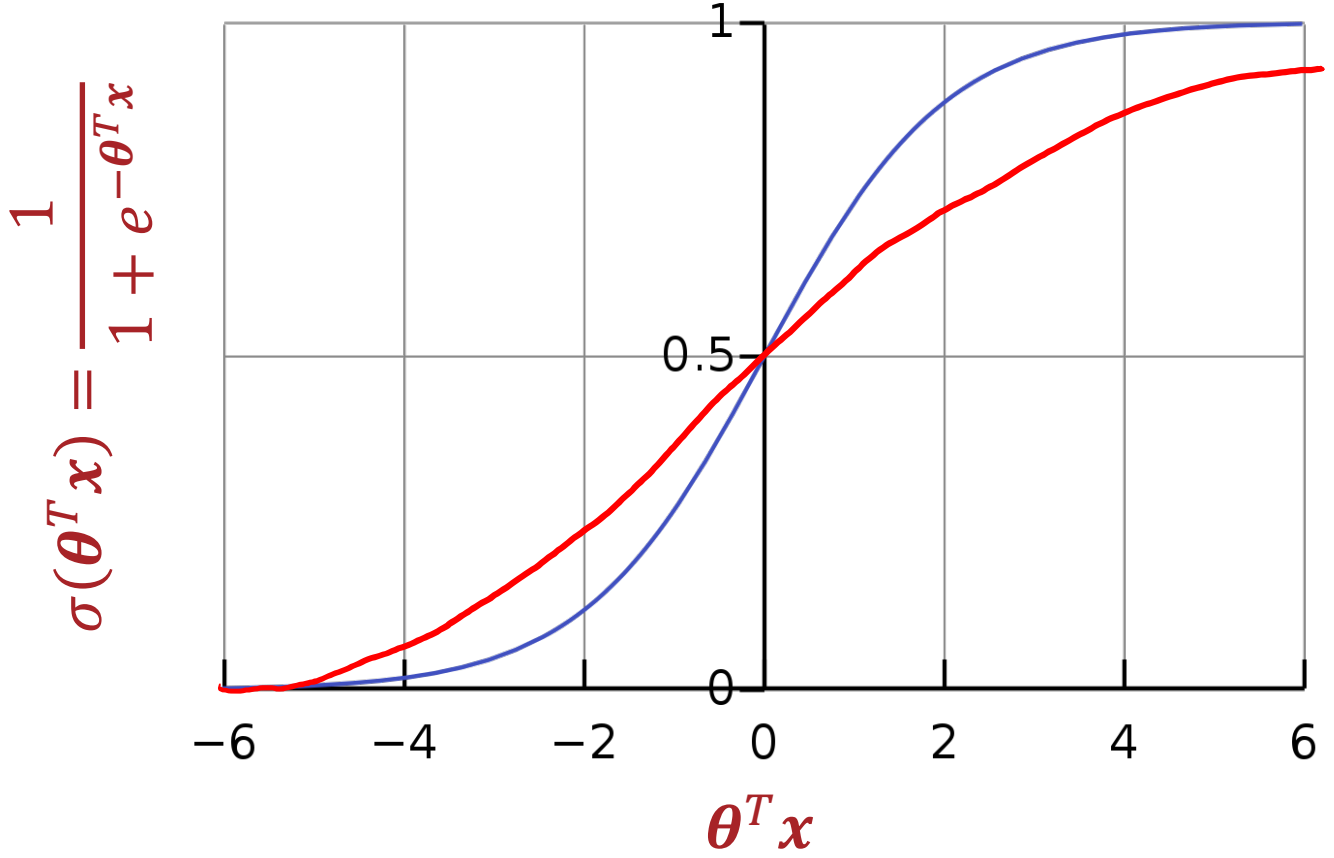
$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Source: https://en.wikipedia.org/wiki/Logistic_function#/media/File:Logistic-curve.svg

Why use the Logistic Function?

- The resulting decision boundary is linear!



- $\sigma : \mathbb{R} \mapsto (0,1)$
- It's differentiable everywhere
- Monotonically increasing
- symmetric about

Source: https://en.wikipedia.org/wiki/Logistic_function#/media/File:Logistic-curve.svg

Logistic Regression Decision Boundary

$$\hat{y} = \begin{cases} 1 & \text{if } P(Y=1|x,\theta) \geq 1/2 \\ 0 & \text{otherwise} \end{cases}$$

$$P(Y=1|x,\theta) = \frac{1}{2} \Rightarrow \frac{1}{1+\exp(-\theta^T x)} = \frac{1}{2}$$

$$\Rightarrow 2 = 1 + \exp(-\theta^T x)$$

$$\Rightarrow 1 = \exp(-\theta^T x)$$

$$\Rightarrow 0 = -\theta^T x$$

$$\Rightarrow \theta^T x = 0$$

Logistic Regression Decision Boundary

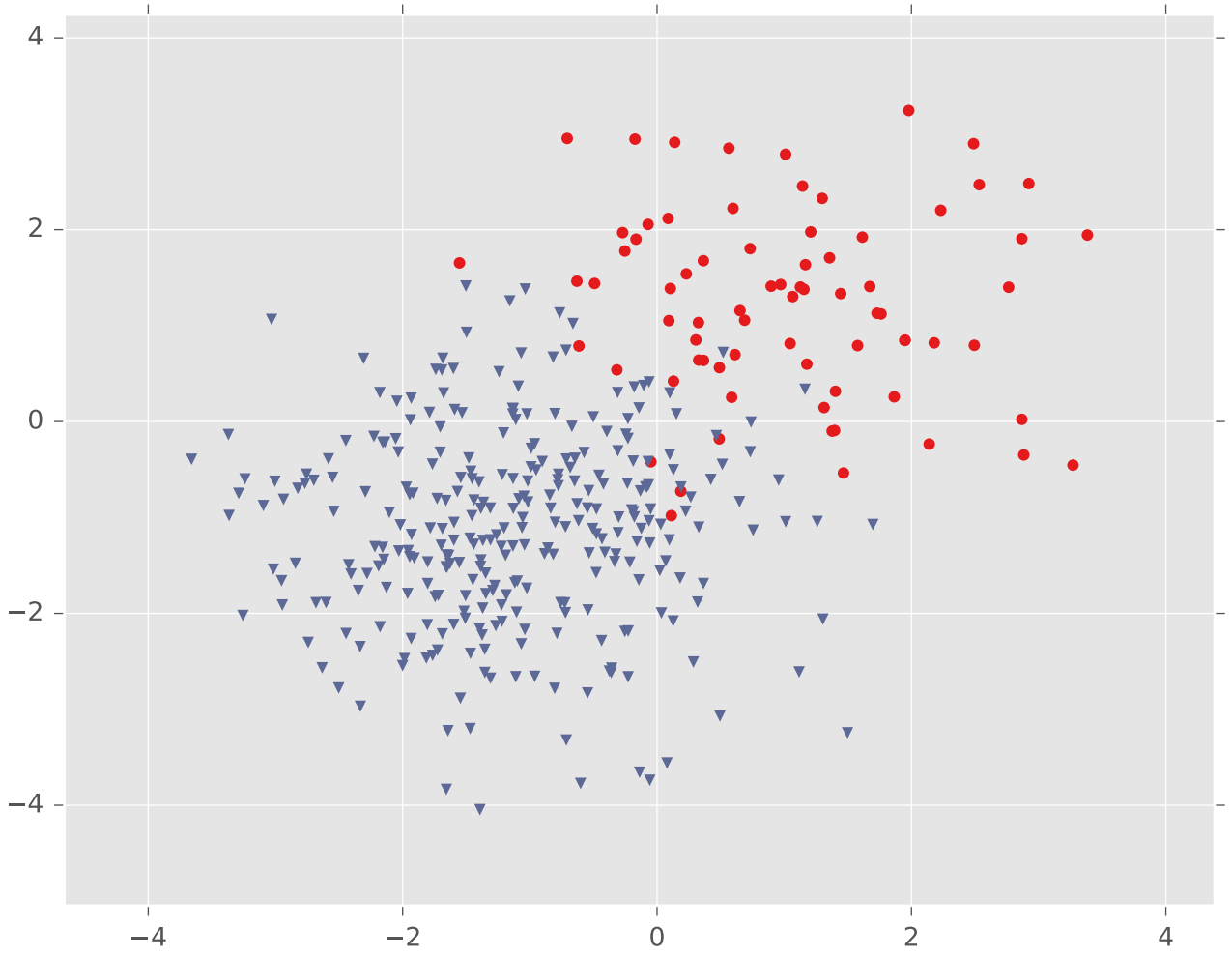


Figure courtesy of Matt Gormley

Logistic Regression Decision Boundary

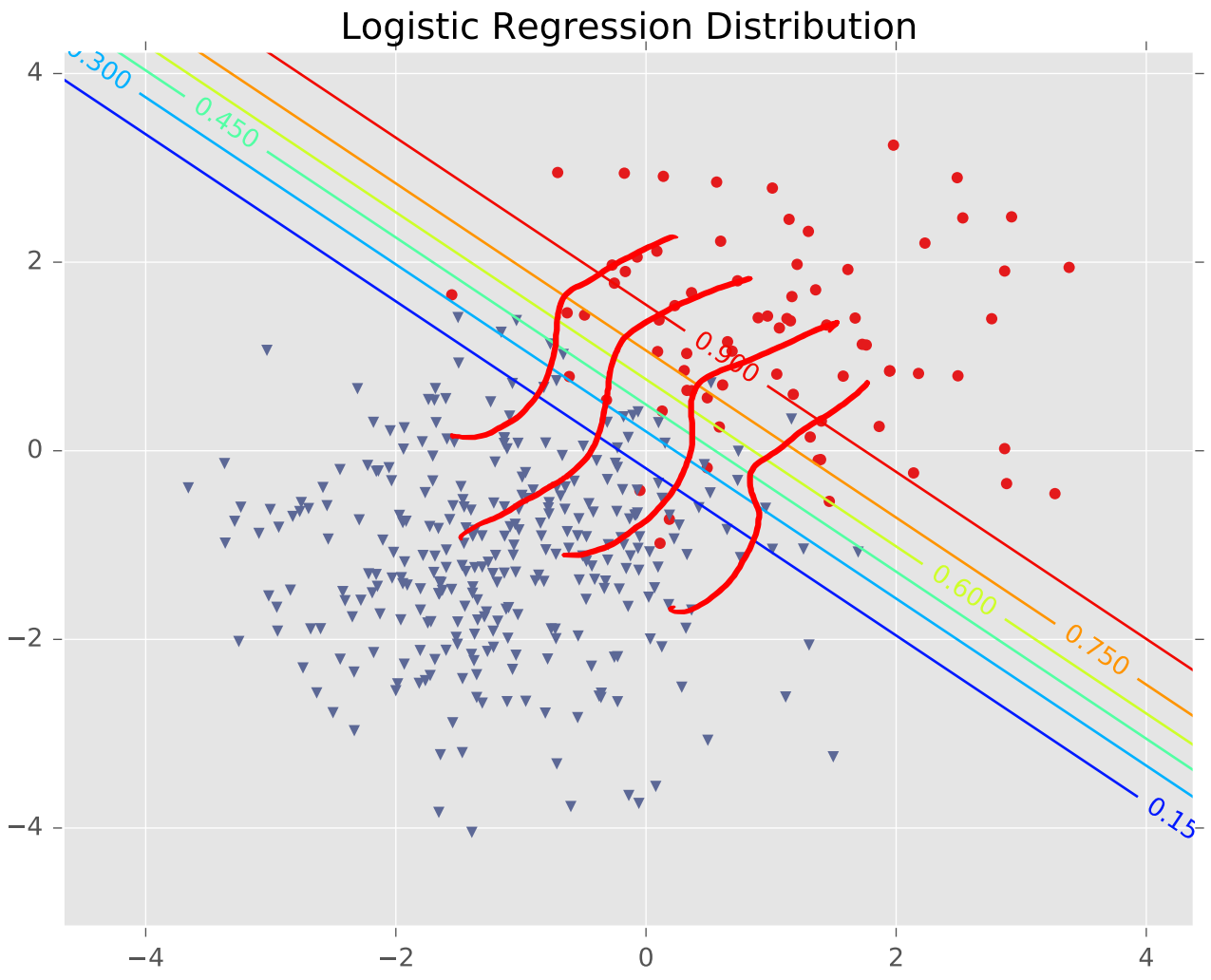


Figure courtesy of Matt Gormley

Logistic Regression Decision Boundary

why $1/2$?

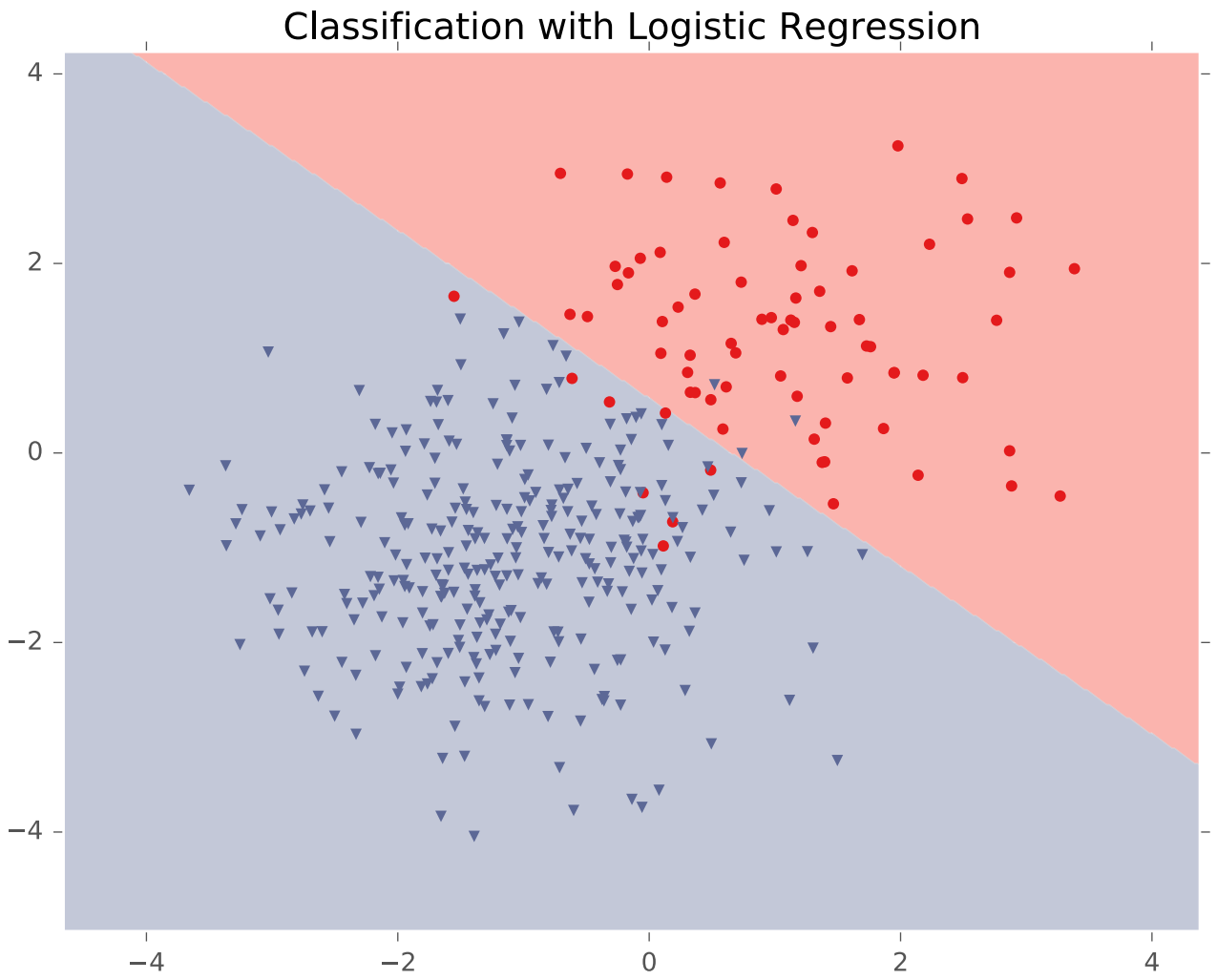


Figure courtesy of Matt Gormley

Setting the Parameters via Minimum Negative Conditional (log-)Likelihood Estimation (MCLE)

- Find $\boldsymbol{\theta}$ that minimizes

assume iid

$$\ell(\boldsymbol{\theta}) = -\log P\left(y^{(1)}, \ldots, y^{(N)} \middle| \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}, \boldsymbol{\theta}\right) = -\log \prod_{i=1}^{N} P\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \boldsymbol{\theta}\right)$$

$$= -\log \prod_{i=1}^{N} \left(P(Y=1|x^{(i)};\theta)\right)^{y^{(i)}} \left(P(Y=0|x^{(i)},\theta)\right)^{1-y^{(i)}}$$

$$= -\sum_{i=1}^{N} \log\left(P(Y=1|x^{(i)};\theta)\right)^{y^{(i)}} + \log\left(P(Y=0|x^{(i)};\theta)\right)^{1-y^{(i)}}$$

$$= -\sum_{i=1}^{N} y^{(i)} \log\left(P(Y=1|x^{(i)};\theta)\right) + (1-y^{(i)}) \log\left(P(Y=0|x^{(i)};\theta)\right)$$

$$= -\sum_{i=1}^{N} y^{(i)} \log \frac{P(Y=1|x^{(i)};\theta)}{P(Y=0|x^{(i)};\theta)} + \log P(Y=0|x^{(i)};\theta)$$

$$= -\sum_{i=1}^{N} y^{(i)} (\theta^T x^{(i)}) + \log\left(\frac{1}{1+\exp(\theta^T x^{(i)})}\right)$$

$$= -\sum_{i=1}^{N} y^{(i)} (\theta^T x^{(i)}) - \log\left(1+\exp(\theta^T x^{(i)})\right)$$

$\Rightarrow J(\theta) = \frac{1}{N}\ell(\theta)$

Minimizing the Negative Conditional (log-)Likelihood

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N} y^{(i)}\boldsymbol{\theta}^T\boldsymbol{x}^{(i)} - \log\left(1+\exp\left(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)}\right)\right)$$

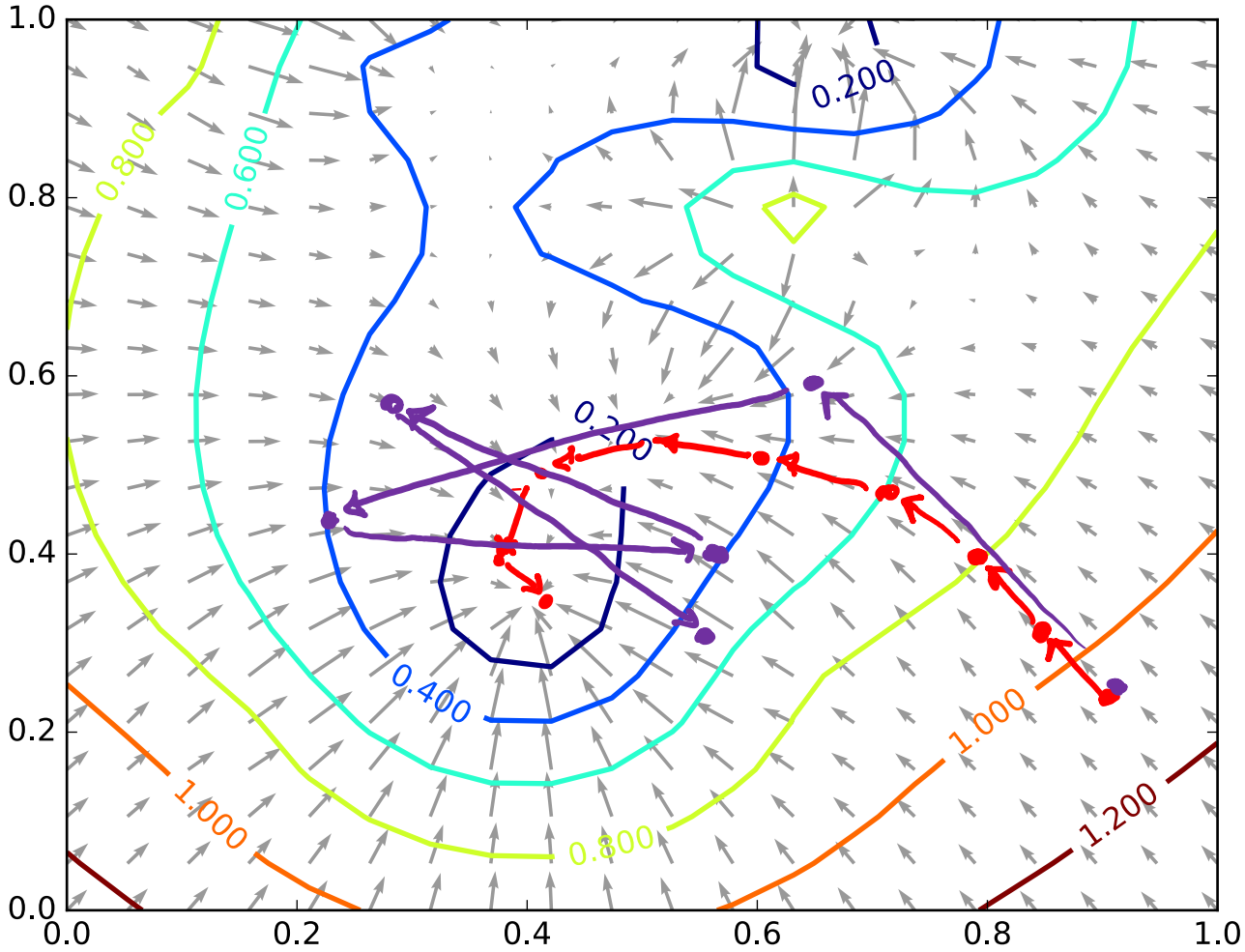
$$\nabla_\theta J(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \nabla_\theta\left(y^{(i)}\theta^T x^{(i)}\right) - \nabla_\theta \log\left(1+\exp\left(\theta^T x^{(i)}\right)\right)$$

$$= -\frac{1}{N}\sum_{i=1}^{N} y^{(i)}x^{(i)} - \frac{1}{1+\exp(\theta^T x^{(i)})}\nabla_\theta\left(1+\exp(\theta^T x^{(i)})\right)$$

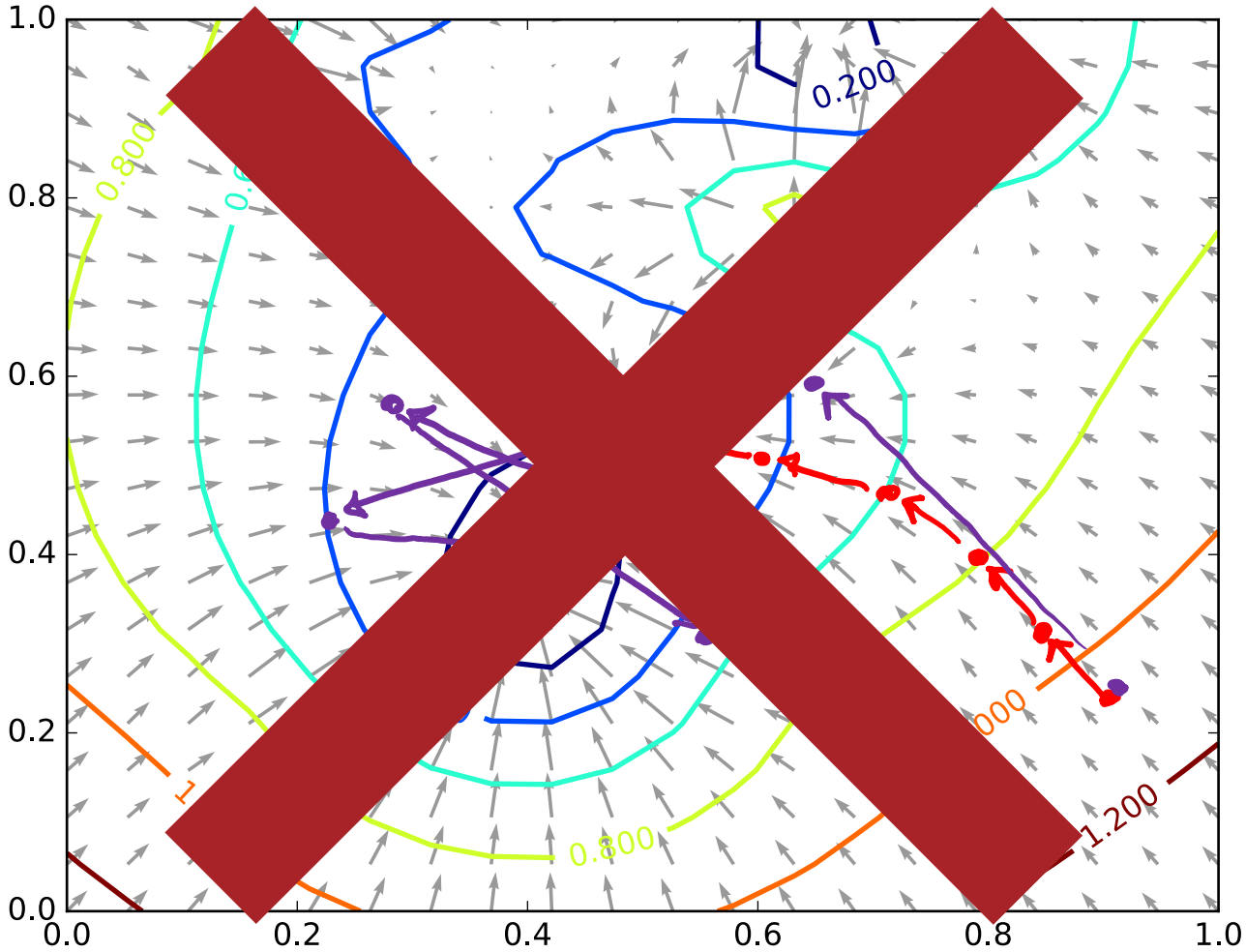
$$= -\frac{1}{N}\sum_{i=1}^{N} y^{(i)}x^{(i)} - \frac{\exp(\theta^T x^{(i)})}{1+\exp(\theta^T x^{(i)})}x^{(i)}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(\underbrace{\frac{\exp(\theta^T x^{(i)})}{1+\exp(\theta^T x^{(i)})}}_{\sigma(\theta^T x^{(i)}) = P(Y=1 \mid x^{(i)}, \theta)} - y^{(i)}\right)x^{(i)}$$

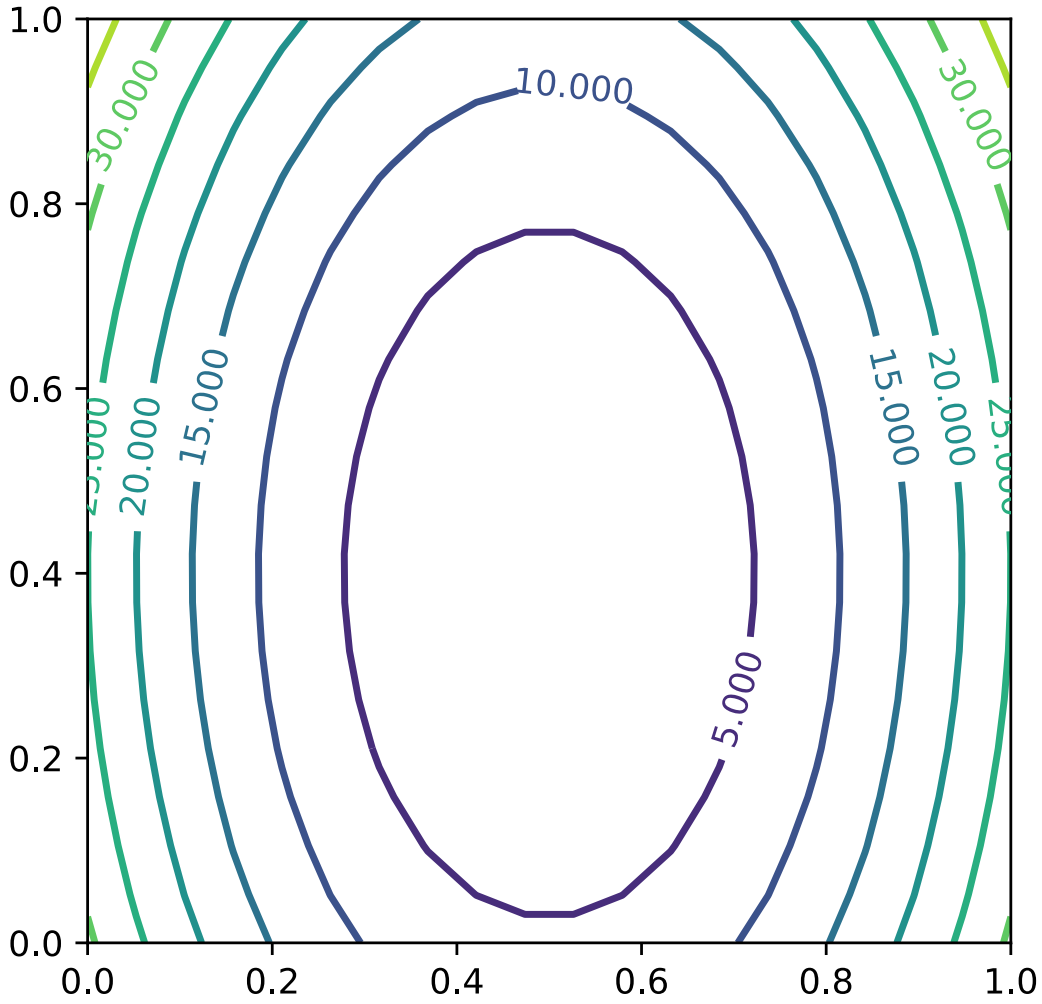
Recall: Gradient Descent



Recall: Gradient Descent



Recall: Gradient Descent



Good news: the negative conditional log-likelihood is convex!

(See HW/recitation)

Gradient Descent

- Input: training dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ and step size γ

1. Initialize $\boldsymbol{\theta}^{(0)}$ to all zeros and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. Compute the gradient:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(t)}) = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}^{(i)} \left(P(Y = 1 | \boldsymbol{x}^{(i)}, \boldsymbol{\theta}^{(t)}) - y^{(i)} \right)$$

   b. Update $\boldsymbol{\theta}$: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \gamma \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(t)})$
   c. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{\theta}^{(t)}$

Poll Question 1:

What is the computational cost of one iteration of gradient descent for logistic regression?

A. $O(1)$ **(TOXIC)** B. $O(N)$ C. $O(D)$ D. $O(ND)$

- Input: training dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ and step size γ

1. Initialize $\boldsymbol{\theta}^{(0)}$ to all zeros and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. Compute the gradient:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(t)}) = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}^{(i)} \left(P(Y = 1 | \boldsymbol{x}^{(i)}, \boldsymbol{\theta}^{(t)}) - y^{(i)} \right)$$

   b. Update $\boldsymbol{\theta}$: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \gamma \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(t)})$
   c. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{\theta}^{(t)}$

Gradient Descent

- Input: training dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ and step size γ

1. Initialize $\boldsymbol{\theta}^{(0)}$ to all zeros and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. Compute the gradient:

$$O(ND) \left\{ \quad \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(t)}) = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}^{(i)} \left(P(Y = 1 | \boldsymbol{x}^{(i)}, \boldsymbol{\theta}^{(t)}) - y^{(i)} \right) \right.$$

   b. Update $\boldsymbol{\theta}$: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \gamma \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(t)})$
   c. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{\theta}^{(t)}$

Stochastic Gradient Descent (SGD)

- Input: training dataset $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$ and step size γ

1. Initialize $\boldsymbol{\theta}^{(0)}$ to all zeros and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. Randomly sample a data point from $\mathcal{D}$, $\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)$
   b. Compute the pointwise gradient:

$$\nabla_{\boldsymbol{\theta}} J^{(i)}\left(\boldsymbol{\theta}^{(t)}\right) = \boldsymbol{x}^{(i)}\left(P\left(Y = 1 \middle| \boldsymbol{x}^{(i)}, \boldsymbol{\theta}^{(t)}\right) - y^{(i)}\right)$$

   c. Update $\boldsymbol{\theta}$: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \gamma \nabla_{\boldsymbol{\theta}} J^{(i)}\left(\boldsymbol{\theta}^{(t)}\right)$
   d. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{\theta}^{(t)}$

Stochastic Gradient Descent (SGD)

- If the example is sampled uniformly at random, the expected value of the pointwise gradient is the same as the full gradient!

$$\mathbb{E}\left[\nabla_\theta J^{(i)}(\theta)\right] = \sum_{i=1}^{N} (\text{probability of choosing } (x^{(i)}, y^{(i)})) \nabla_\theta J^{(i)}(\theta)$$

$$= \sum_{i=1}^{N} \frac{1}{N} \nabla_\theta J^{(i)}(\theta)$$

$$= \nabla_\theta \left(\frac{1}{N} \sum_{i=1}^{N} J^{(i)}(\theta) \right) = \nabla_\theta J(\theta)$$

- In practice, the data set is randomly shuffled then looped through so that each data point is used equally often

Stochastic Gradient Descent (SGD)

- Input: training dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ and step size γ

1. Initialize $\boldsymbol{\theta}^{(0)}$ to all zeros and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. For $i \in \text{shuffle}(\{1, \ldots, N\})$
      i. Compute the pointwise gradient:
      $$\nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta}^{(t)}) = \boldsymbol{x}^{(i)}\left(P\left(Y = 1 \middle| \boldsymbol{x}^{(i)}, \boldsymbol{\theta}^{(t)}\right) - y^{(i)}\right)$$
      ii. Update $\boldsymbol{\theta}$: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \gamma \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta}^{(t)})$
      iii. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{\theta}^{(t)}$

Wait, have we seen something like this before?

- Input: training dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ and step size γ

1. Initialize $\boldsymbol{\theta}^{(0)}$ to all zeros and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. For $i \in \text{shuffle}(\{1, \ldots, N\})$
      i. Compute the pointwise gradient:
      $$\nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta}^{(t)}) = \boldsymbol{x}^{(i)}\left(P\left(Y = 1 \middle| \boldsymbol{x}^{(i)}, \boldsymbol{\theta}^{(t)}\right) - y^{(i)}\right)$$
      ii. Update $\boldsymbol{\theta}$: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \gamma \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta}^{(t)})$
      iii. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{\theta}^{(t)}$

Recall: Perceptron Learning Algorithm

- Input: training dataset $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$

1. Initialize $\boldsymbol{\theta}^{(0)}$ to all zeros and set $t = 0$
2. While NOT CONVERGED
   a. For $t \in \text{shuffle}(\{1, \ldots, N\})$
      i. Predict the label of $\boldsymbol{x}^{(t)}$, $\hat{y} = \text{sign}\left(\boldsymbol{\theta}^T \boldsymbol{x}^{(t)}\right)$
      ii. Observe its true label, $y^{(t)}$
      iii. If we misclassified $\boldsymbol{x}^{(t)}$ ($y^{(t)} \neq \hat{y}$):

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(t)} \boldsymbol{x}^{(t)}$$

This is the negative gradient of the *hinge loss*

$$J^{(t)}(\boldsymbol{\theta}) = \max\left(0, -y^{(t)}\left(\boldsymbol{\theta}^T \boldsymbol{x}^{(t)}\right)\right)$$

Recall: Perceptron Learning Algorithm

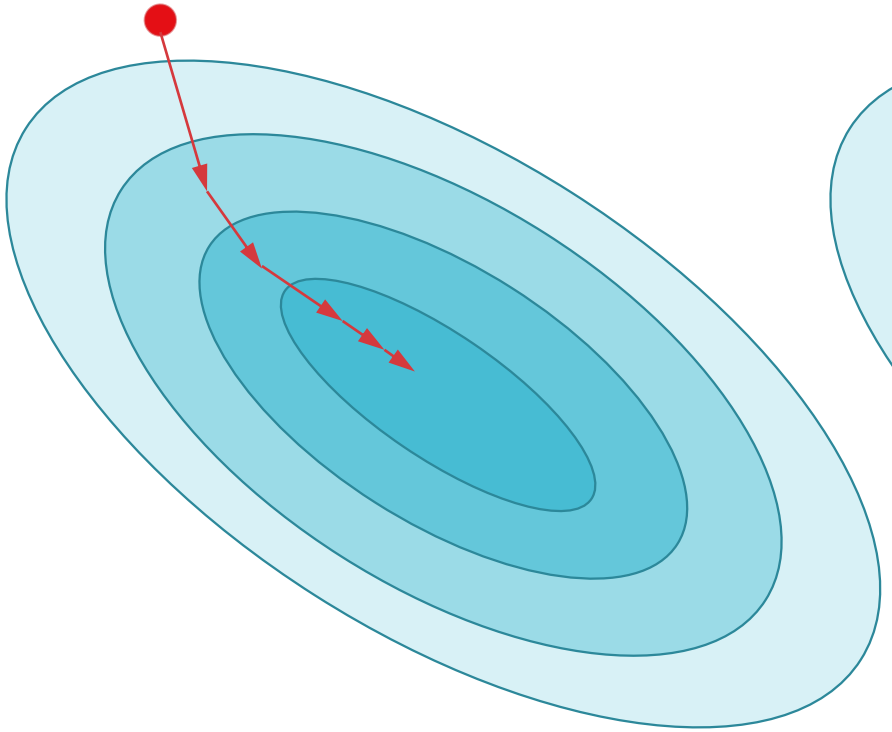
- Input: training dataset $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$

1. Initialize $\boldsymbol{\theta}^{(0)}$ to all zeros and set $t = 0$
2. While NOT CONVERGED
   a. For $t \in \text{shuffle}(\{1, \ldots, N\})$
      i. Predict the label of $\boldsymbol{x}^{(t)}$, $\hat{y} = \text{sign}\left(\boldsymbol{\theta}^T \boldsymbol{x}^{(t)}\right)$
      ii. Observe its true label, $y^{(t)}$
      iii. If we misclassified $\boldsymbol{x}^{(t)}$ ($y^{(t)} \neq \hat{y}$):

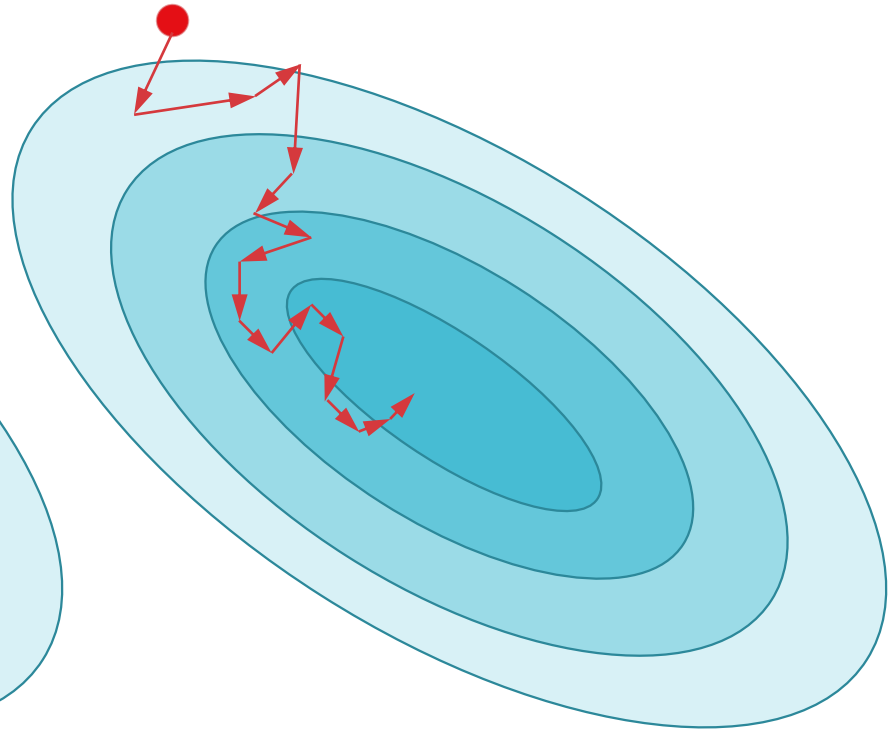
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(t)} \boldsymbol{x}^{(t)}$$

$$\nabla_{\boldsymbol{\theta}} J^{(t)}(\boldsymbol{\theta}) = \begin{cases} 0 & \text{if sign}\left(\boldsymbol{\theta}^T \boldsymbol{x}^{(t)}\right) = y^{(t)} \\ -y^{(t)} \boldsymbol{x}^{(t)} & \text{otherwise} \end{cases}$$

Stochastic Gradient Descent vs. Gradient Descent



Gradient Descent



Stochastic Gradient Descent

Stochastic Gradient Descent vs. Gradient Descent

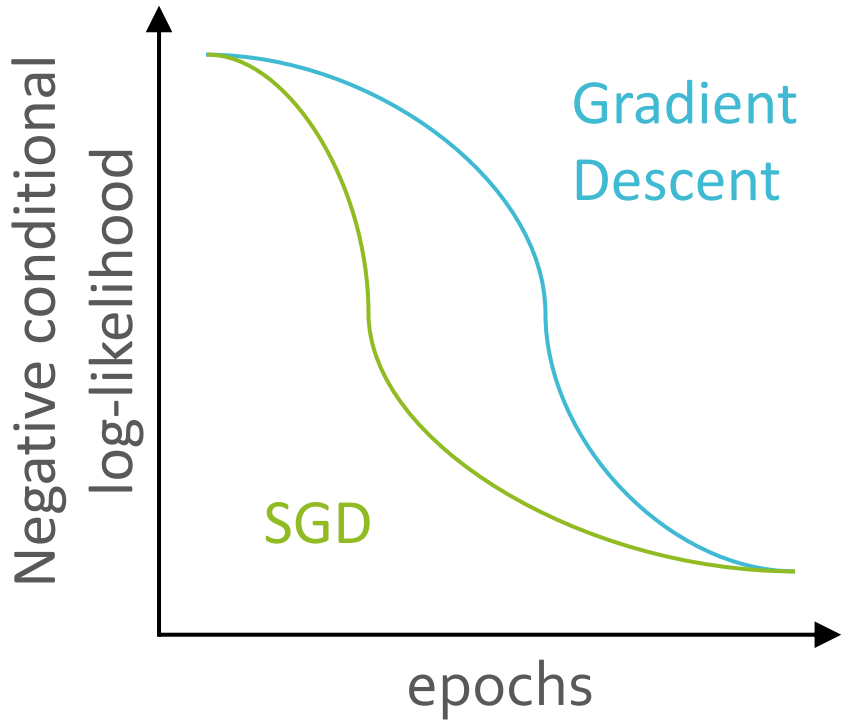
- An *epoch* is a single pass through the entire training dataset
  - Gradient descent updates the parameters once per epoch
  - SGD updates the parameters N times per epoch
- Theoretical comparison:
  - Define convergence to be when $J(\boldsymbol{\theta}^{(t)}) - J(\boldsymbol{\theta}^*) < \epsilon$

Method	Steps to Convergence	Computation per Step
Gradient descent	$O(\log 1/\epsilon)$	$O(ND)$
SGD	$O(1/\epsilon)$	$O(D)$

(with high probability under certain assumptions)

Stochastic Gradient Descent vs. Gradient Descent

- An *epoch* is a single pass through the entire training dataset
  - Gradient descent updates the parameters once per epoch
  - SGD updates the parameters N times per epoch



Empirically, SGD reduces the negative conditional log-likelihood much faster than gradient descent

Optimization for ML Learning Objectives

You should be able to…

- Apply gradient descent to optimize a function
- Apply stochastic gradient descent (SGD) to optimize a function
- Apply knowledge of zero derivatives to identify a closed-form solution (if one exists) to an optimization problem
- Distinguish between convex, concave, and nonconvex functions
- Obtain the gradient (and Hessian) of a (twice) differentiable function

Logistic Regression Learning Objectives

You should be able to…

- Apply the principle of maximum likelihood estimation (MLE) to learn the parameters of a probabilistic model
- Given a discriminative probabilistic model, derive the conditional log-likelihood, its gradient, and the corresponding Bayes Classifier
- Explain the practical reasons why we work with the log of the likelihood
- Implement logistic regression for binary (and multiclass) classification
- Prove that the decision boundary of binary logistic regression is linear