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

Hoda Heidari, Henry Chai & Matt Gormley

2/14/24

Front Matter

- Announcements:
  - Exam 1 on 2/19 from 7 PM – 9 PM
    - Exam 1 practice problems released on the course website, under Coursework

Q & A:

Man, I've really been struggling with the homeworks in this class, especially the programming…

- … where can I turn for help?
- First off, I'm really sorry to hear that…
- … but I'm glad you're asking the right questions: we would love to help you!
  - Your TAs would love to help you in OH!
  - Your instructors would love to help you!
  - We all would love to help you on Piazza!
  - Your peers would (probably) love to help you too (stay tuned for more on this as well)!
- **We would not love it if you violated academic integrity by breaking our collaboration policy**

http://www.cs.cmu.edu/~mgormley/courses/10601/syllabus.html

Recall: Collaboration Policy

- Collaboration on homework assignments is *encouraged* but must be *documented*
- **You must always write your own code/answers**
  - You may not re-use code/previous versions of the homework, whether your own or otherwise
  - **You may not use generative AI tools to create any content for any assessment, including (but not limited to) code**
- Our suggested approach to collaborating:
  1. Collectively sketch pseudocode on an impermanent surface, then
  2. Disperse, erase all notes and start from scratch

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^*

$P(A \cap B) = P(A)P(B)$ (if A and B are independent)

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

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

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

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

Log-Likelihood

- Given N independent, identically distribution (iid) samples $\mathcal{D} = \{x^{(1)}, \dots, x^{(N)}\}$ 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

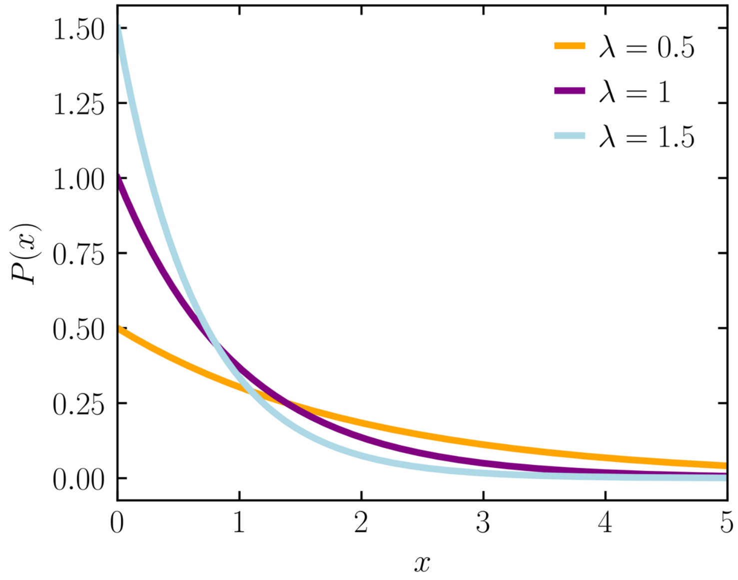
$$\ell(\theta) = \log \prod_{n=1}^{N} p(x^{(n)}|\theta) = \sum_{n=1}^{N} \log p(x^{(n)}|\theta)$$

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

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

Maximum Likelihood Estimation (MLE)

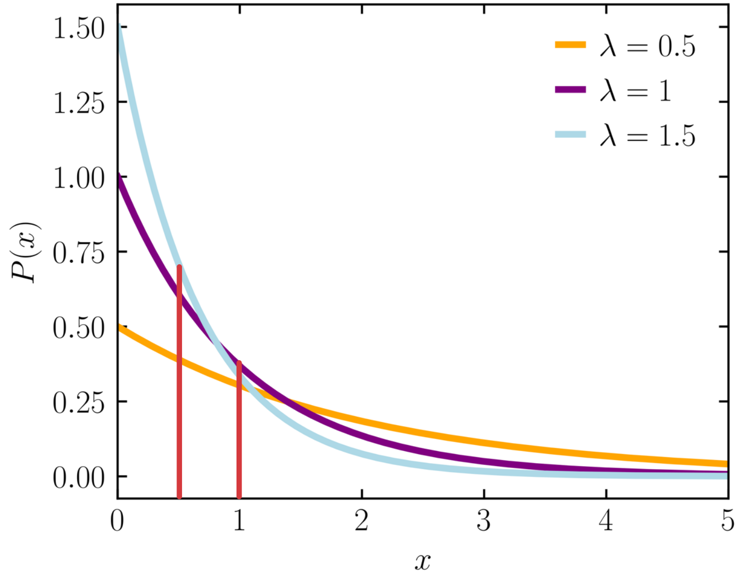
- Insight: every valid probability distribution has a finite amount of probability mass as it must sum/integrate to 1
- Idea: set the parameter(s) so that the likelihood of the samples is maximized
- Intuition: assign as much of the (finite) probability mass to the observed data *at the expense of unobserved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

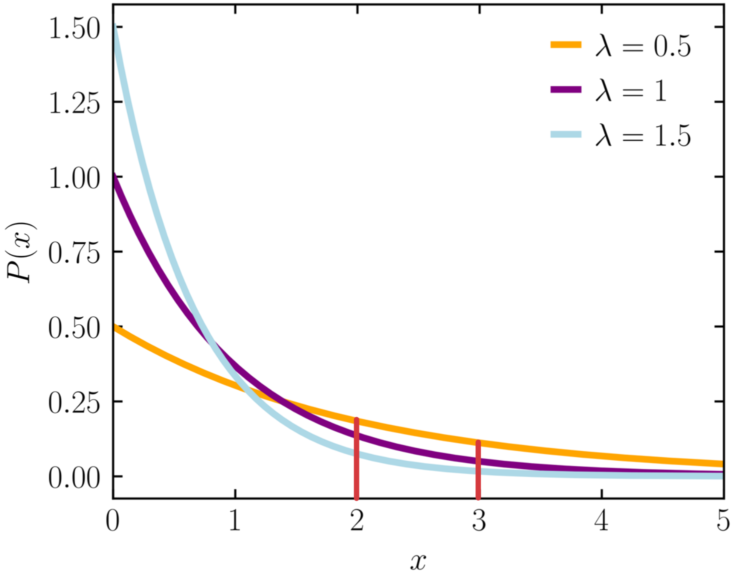


$\{x^{(1)} = 0.5, x^{(2)} = 1\}$

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 $\{x^{(1)}, \dots, x^{(N)}\}$, the likelihood is

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

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

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

$$= \sum_{n=1}^{N} \log \lambda + (-\lambda x^{(n)})$$

$$= N \log \lambda - \sum_{n=1}^{N} \lambda x^{(n)}$$

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

$$\Rightarrow \frac{\partial^2 \ell}{\partial \lambda^2} = -\frac{N}{\lambda^2}$$

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

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: $\hat{y} = \operatorname{argmax}_y P(Y = y|\boldsymbol{x}')$
- 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 $\boldsymbol{x} = [1, x_1, \dots, x_D]^T \in \mathbb{R}^{D+1}$

  1 prepended to $\boldsymbol{x}$

- **Assume**

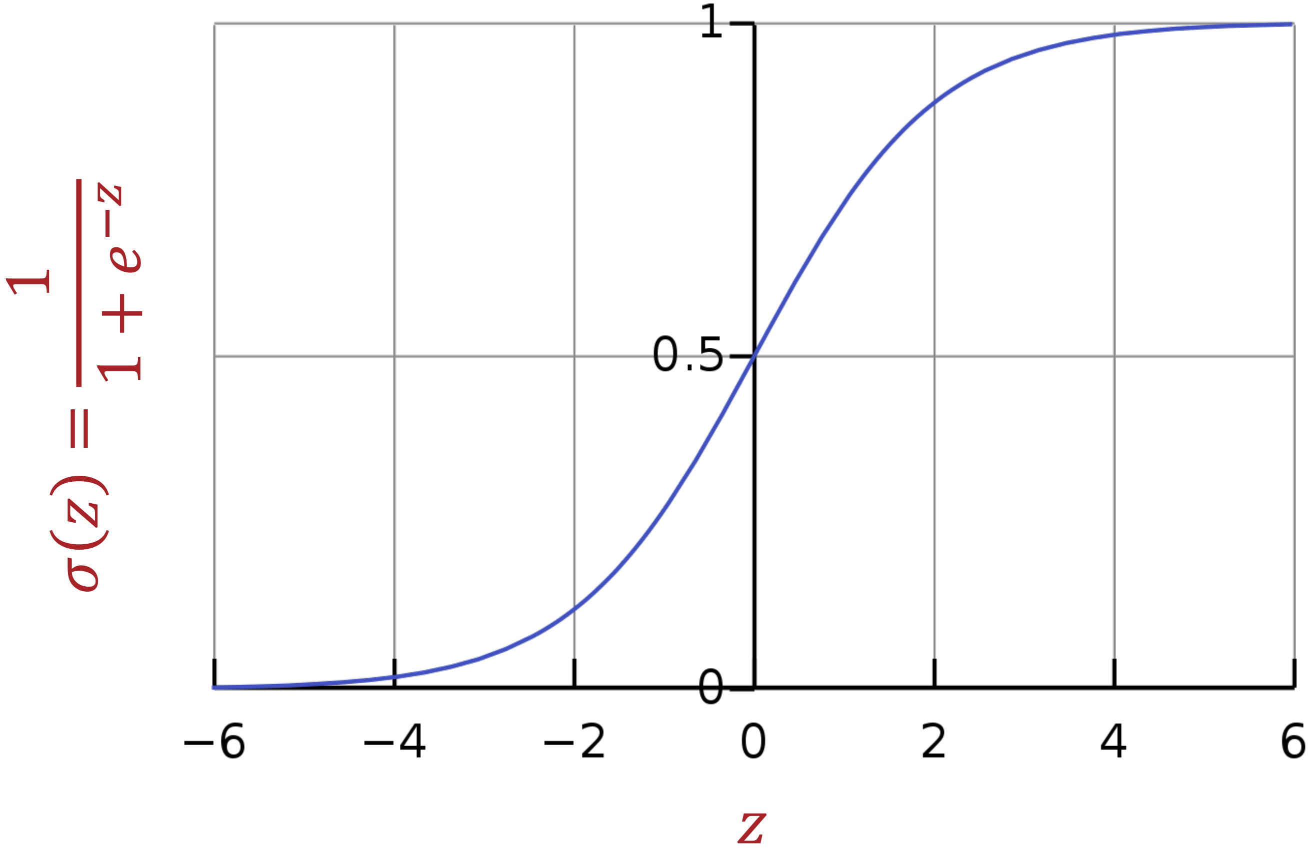
$$P(Y=1 \mid x, \theta) = \sigma(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)} = \frac{\exp(\theta^T x)}{\exp(\theta^T x)+1}$$

Implies

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

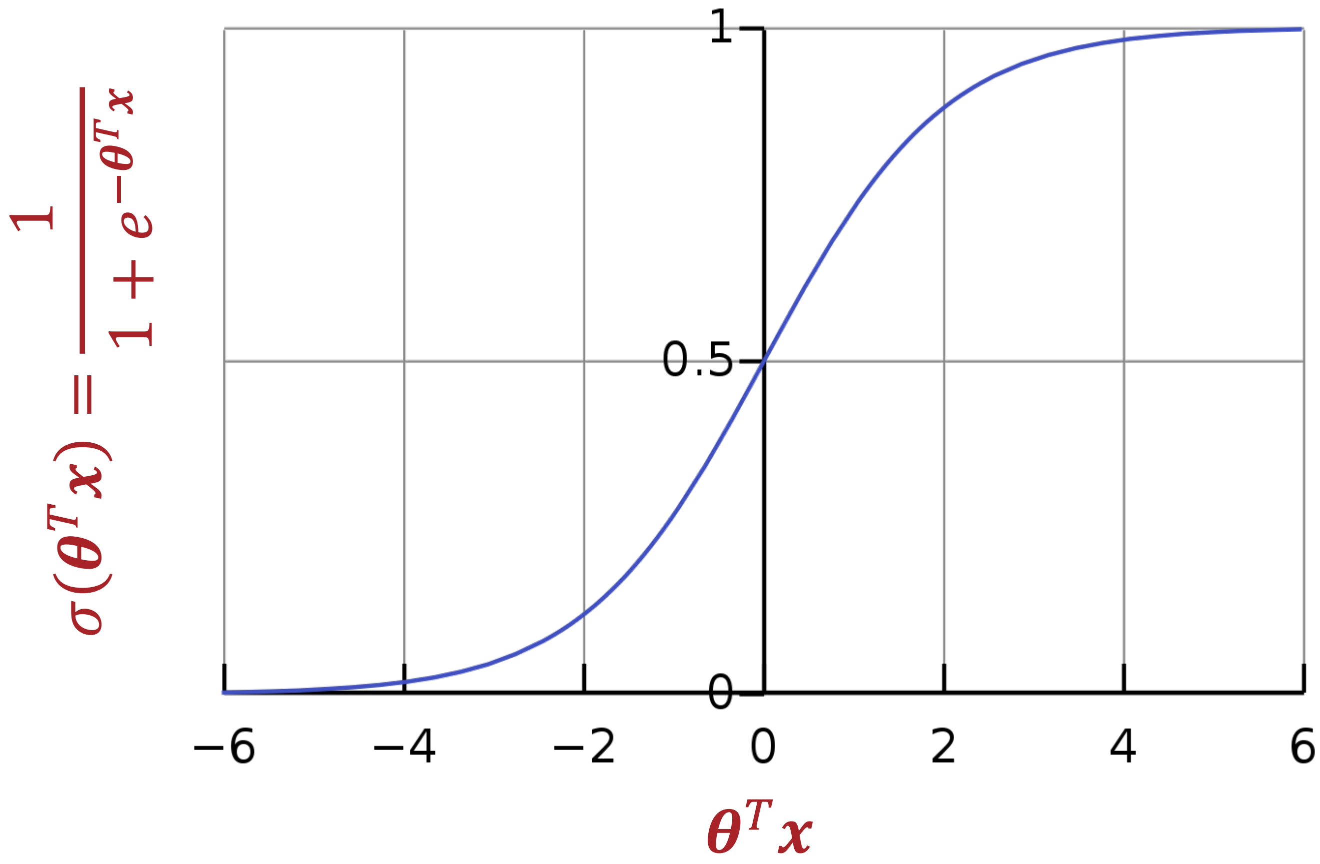
2. $\frac{P(Y=1 \mid x, \theta)}{P(Y=0 \mid x, \theta)} = \exp(\theta^T x) \Rightarrow$ log odds are linear in my inputs, x

Logistic Function



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

Why use the Logistic Function?



$\sigma : \mathbb{R} \mapsto (0, 1)$

smooth → differentiable everywhere

⇒ linear decision boundary!

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

Logistic Regression Decision Boundary

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

$$P(Y=1|x,\theta) = \sigma(\theta^T x) = \frac{1}{2}$$

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

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

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

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

bias term prepended

Logistic Regression Decision Boundary

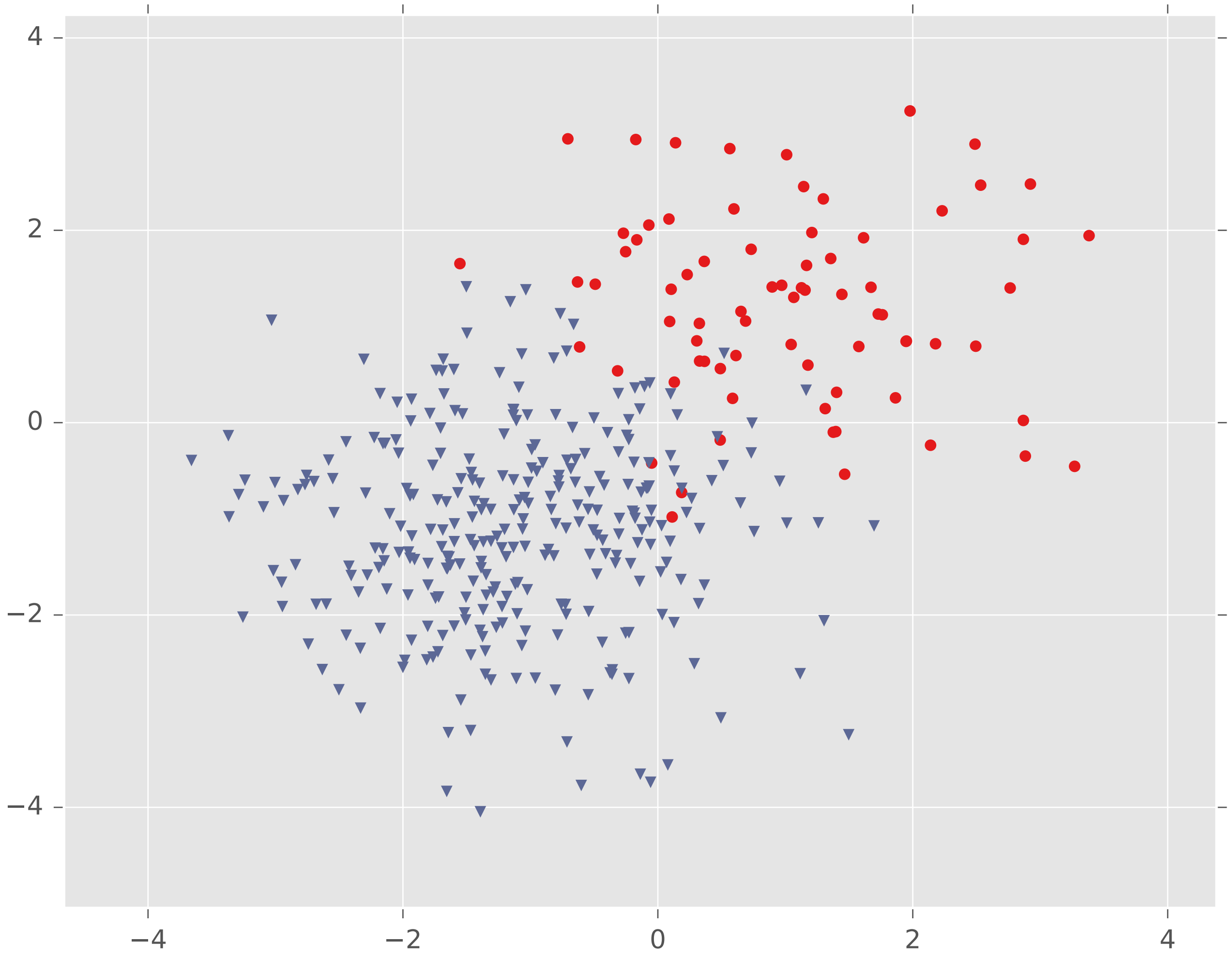


Figure courtesy of Matt Gormley

Logistic Regression Decision Boundary

Figure courtesy of Matt Gormley

Logistic Regression Decision Boundary

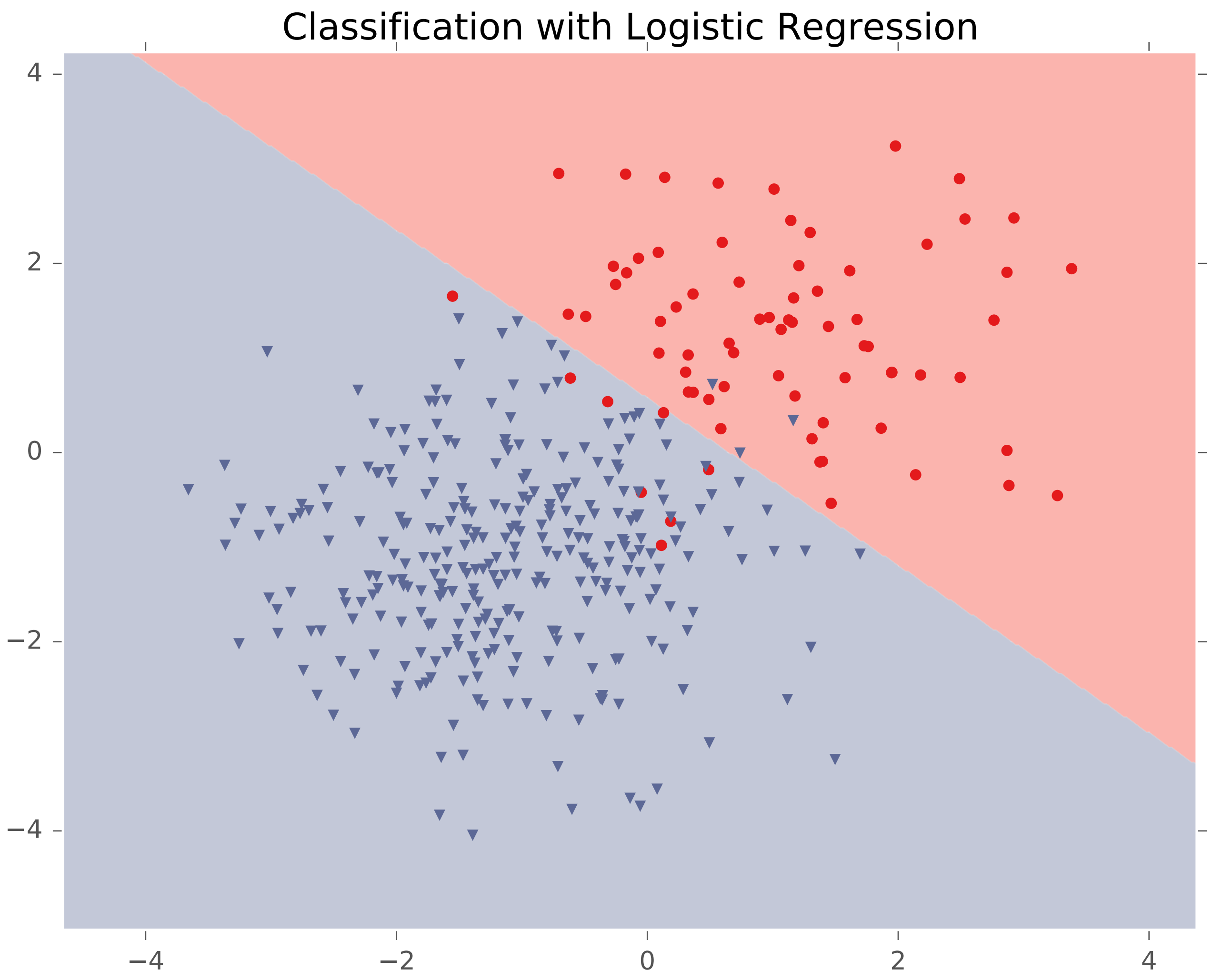


Figure courtesy of Matt Gormley

$\log(a^b c^d)$

$= b \log a + d \log c$

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

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

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

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

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

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

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

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

Minimizing the Negative Conditional (log-)Likelihood

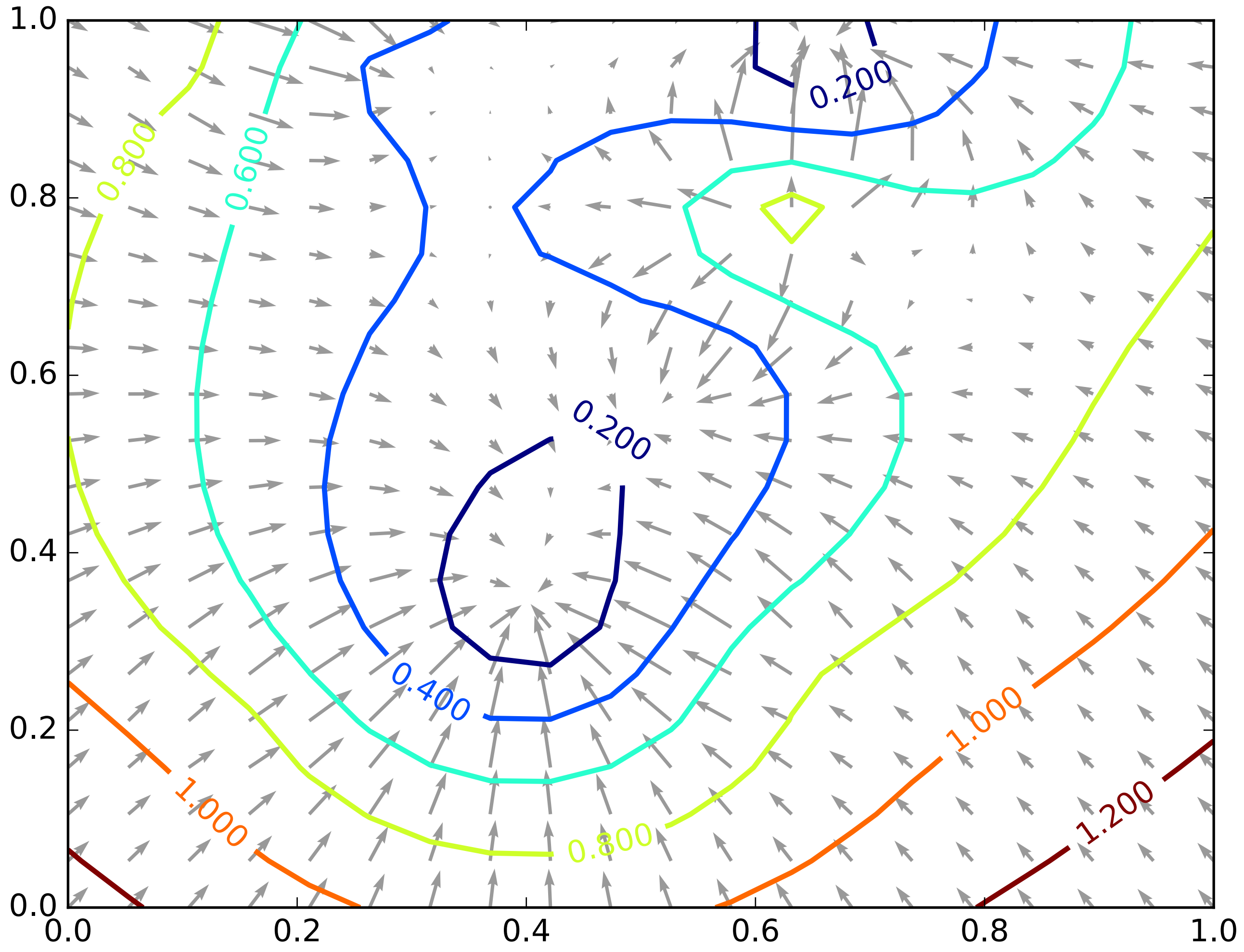
$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{n=1}^{N} y^{(n)}\boldsymbol{\theta}^T\boldsymbol{x}^{(n)} - \log\left(1 + \exp(\boldsymbol{\theta}^T\boldsymbol{x}^{(n)})\right)$$

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

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

$$= \frac{1}{N}\sum_{n=1}^{N} x^{(n)}\left(P(Y=1 \mid x^{(n)}, \theta) - y^{(n)}\right)$$

Recall: Gradient Descent



Gradient Descent

- 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. Compute the gradient:
   $$\nabla_{\boldsymbol{\theta}} J\left(\boldsymbol{\theta}^{(t)}\right) = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}^{(i)}\left(P\left(Y = 1 \middle| \boldsymbol{x}^{(i)}, \boldsymbol{\theta}^{(t)}\right) - y^{(i)}\right)$$
   b. Update $\boldsymbol{\theta}$: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \gamma \nabla_{\boldsymbol{\theta}} J\left(\boldsymbol{\theta}^{(t)}\right)$
   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:

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

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

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

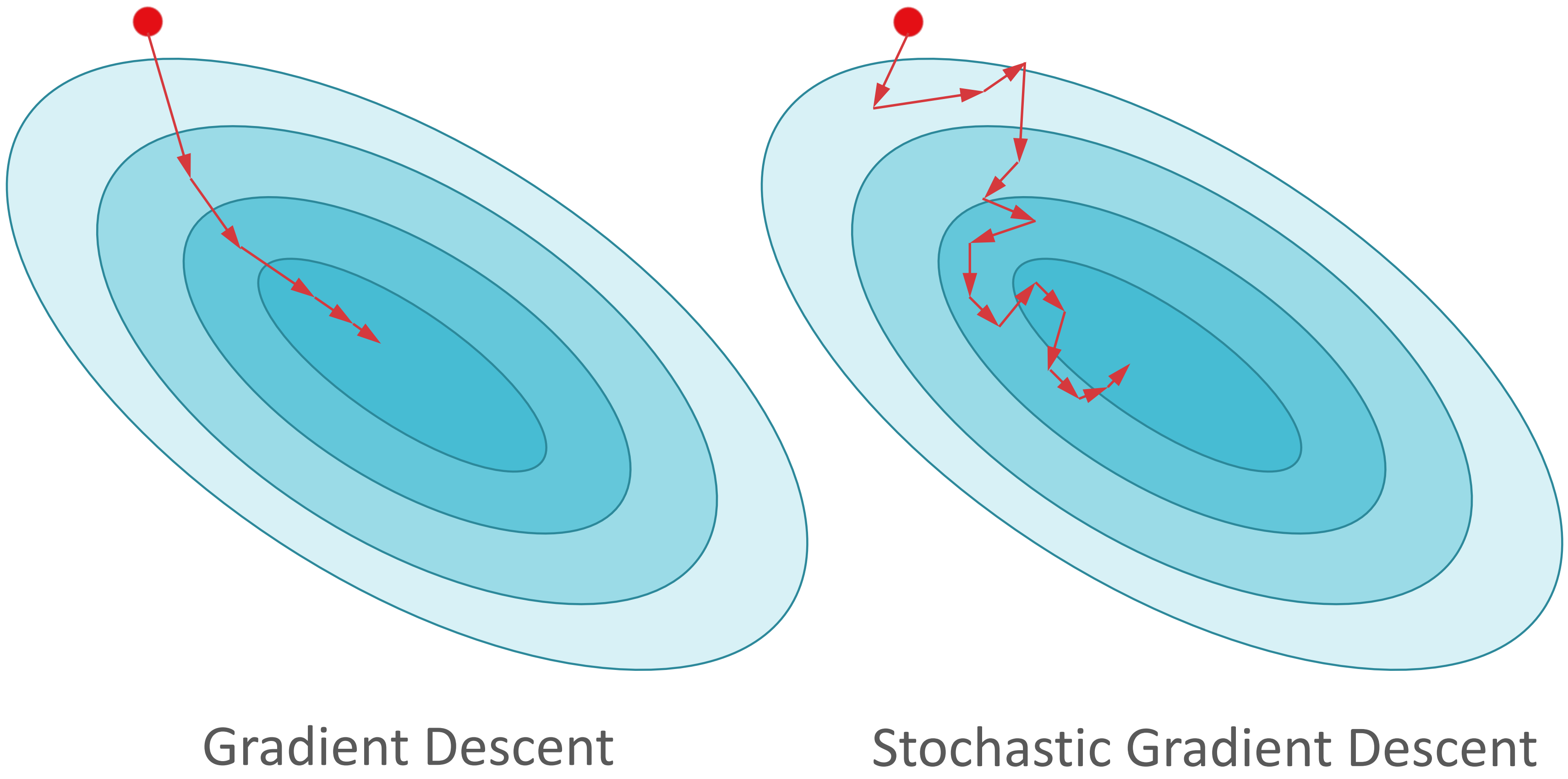
Stochastic Gradient Descent (SGD)

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

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

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

Stochastic Gradient Descent vs. Gradient Descent



Stochastic Gradient Descent vs. Gradient Descent

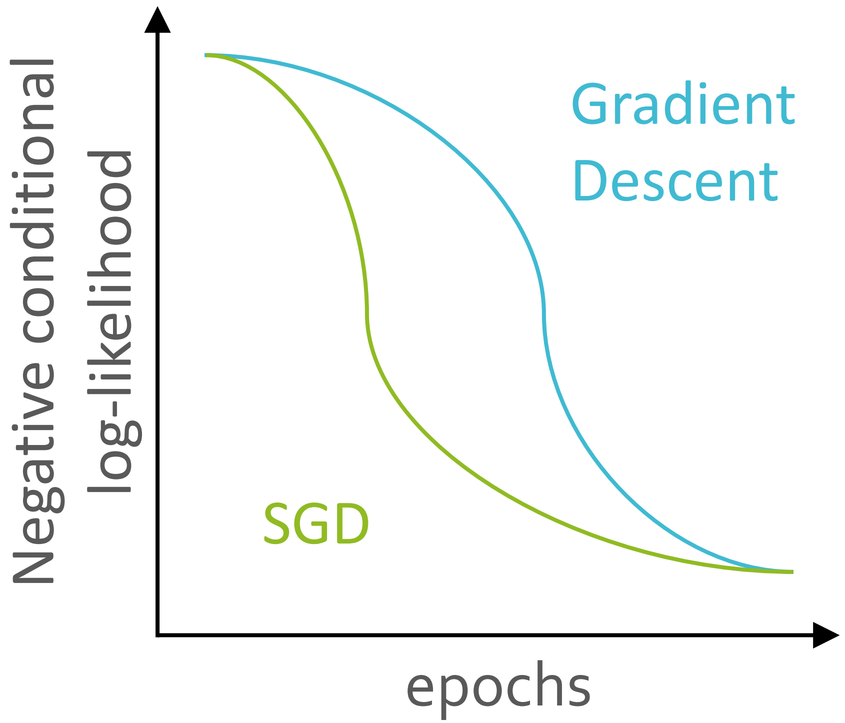
- An *epoch* is a single pass through the entire training dataset
  - Gradient descent updates the parameters once per epoch
  - SGD updates the parameters N times per epoch
- Theoretical comparison:
  - Define convergence to be when $J\left(\boldsymbol{\theta}^{(t)}\right) - J(\boldsymbol{\theta}^{*})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

Method	Steps to Convergence	Computation per Step
Gradient descent	$O\left(\log {}^{1}/_{\epsilon}\right)$	$O(ND)$
SGD	$O\left({}^{1}/_{\epsilon}\right)$	$O(D)$

(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