10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Optimization for ML

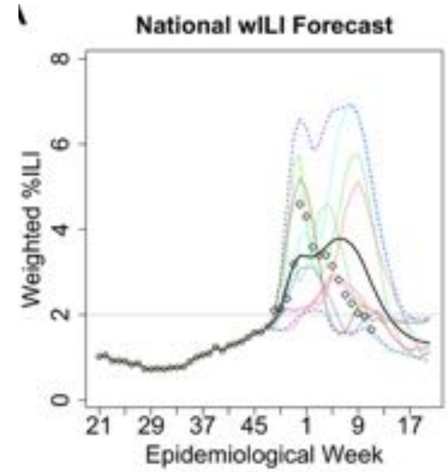
Matt Gormley
Lecture 7
Feb. 7, 2018

Reminders

- **Homework 3: KNN, Perceptron, Lin.Reg.**
  - **Out: Wed, Feb 7**
  - **Due: Wed, Feb 14 at 11:59pm**

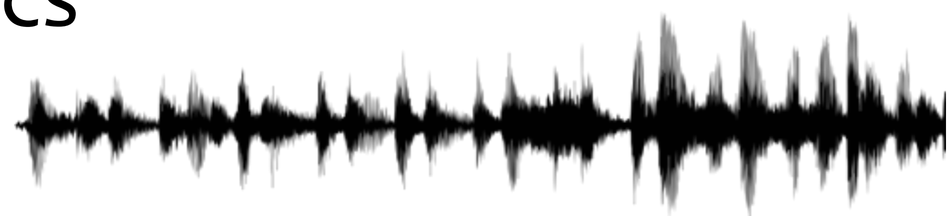
OPTIMIZATION FOR LINEAR REGRESSION

Regression



Example Applications:

- Stock price prediction
- Forecasting epidemics
- Speech synthesis



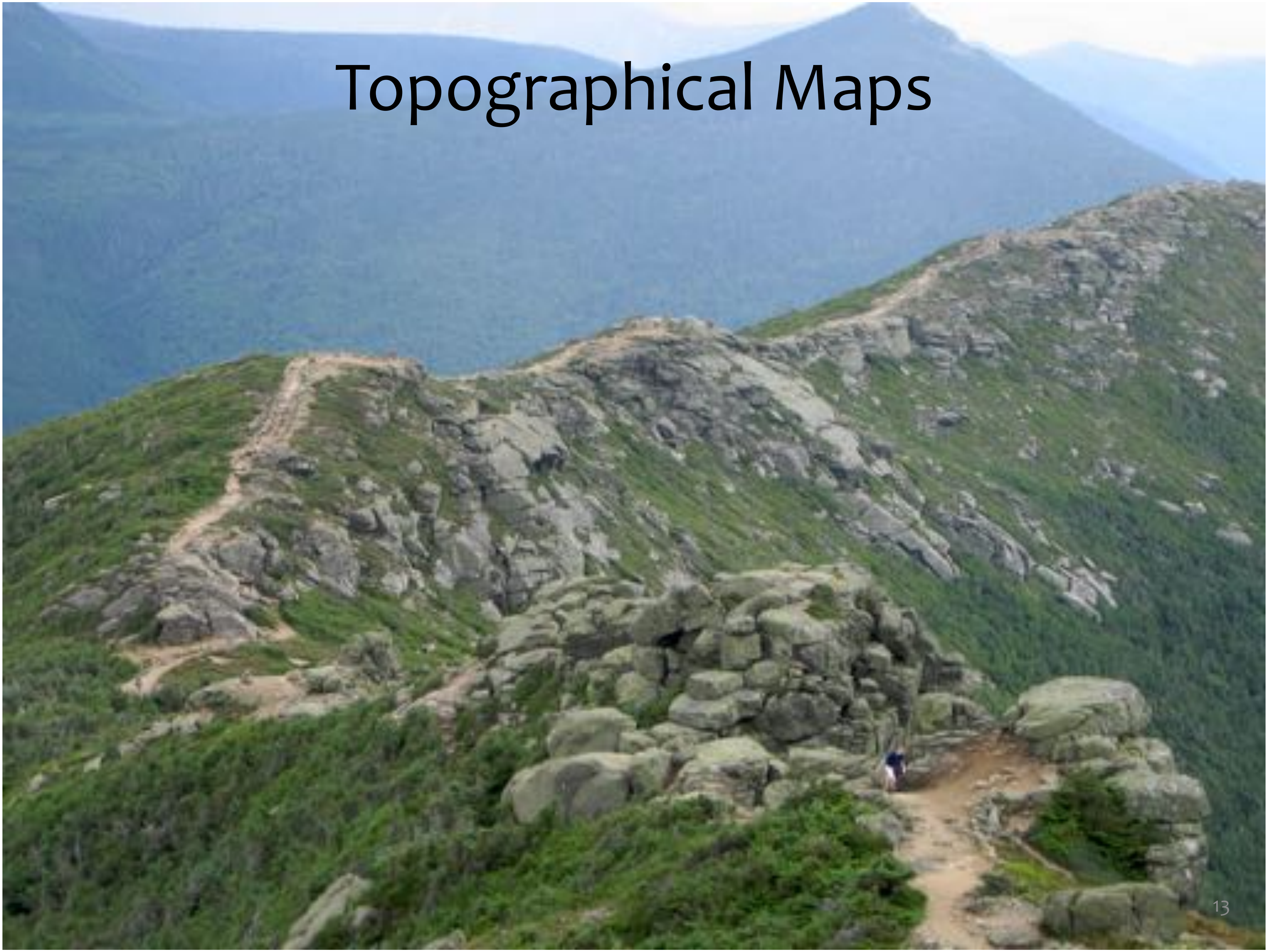
- Generation of images (e.g. *Deep Dream*)
- Predicting the number of tourists on Machu Picchu on a given day

Optimization for Linear Regression

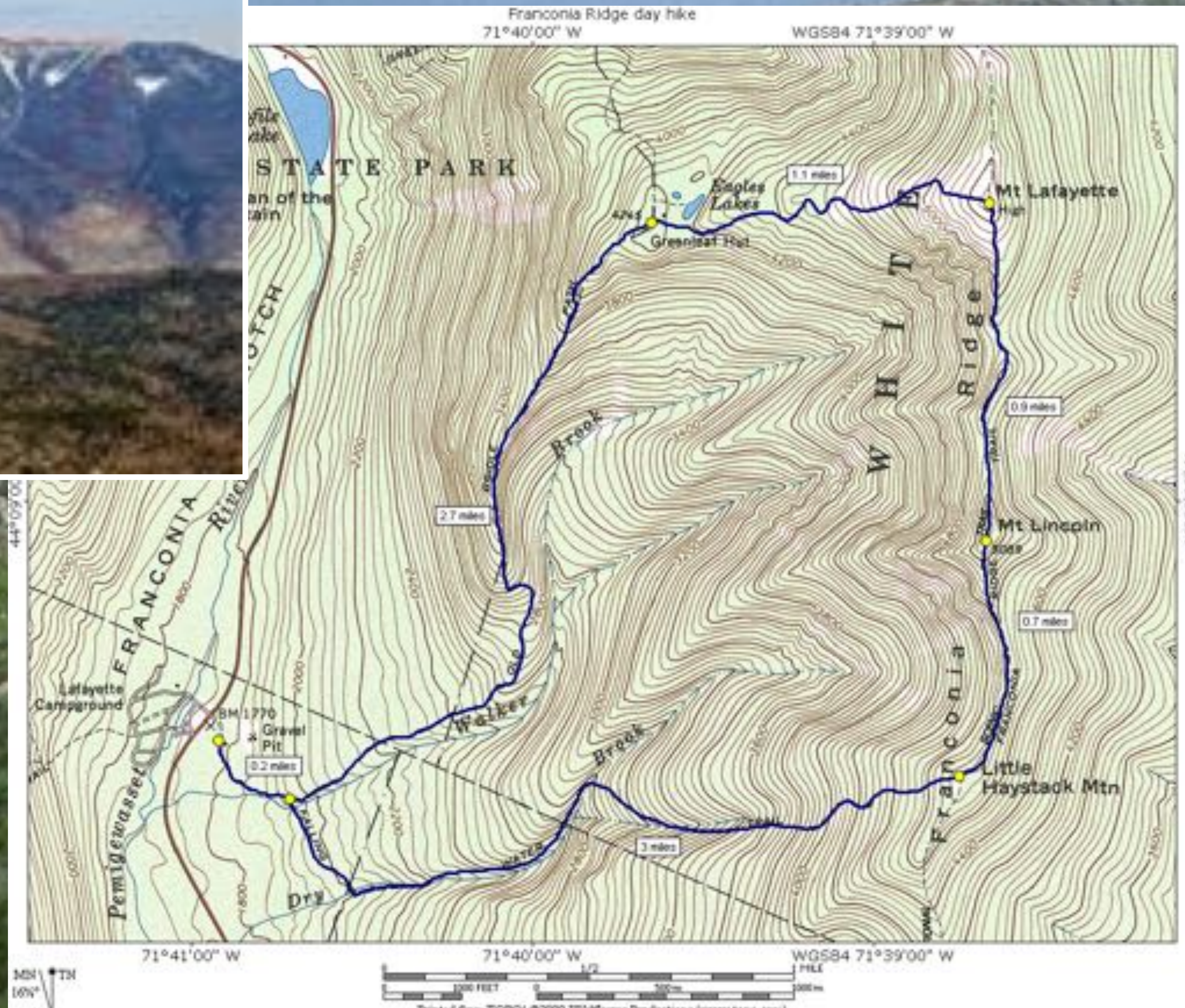
Whiteboard

- Closed-form (Normal Equations)
  - Computational complexity
  - Stability
- Gradient Descent for Linear Regression

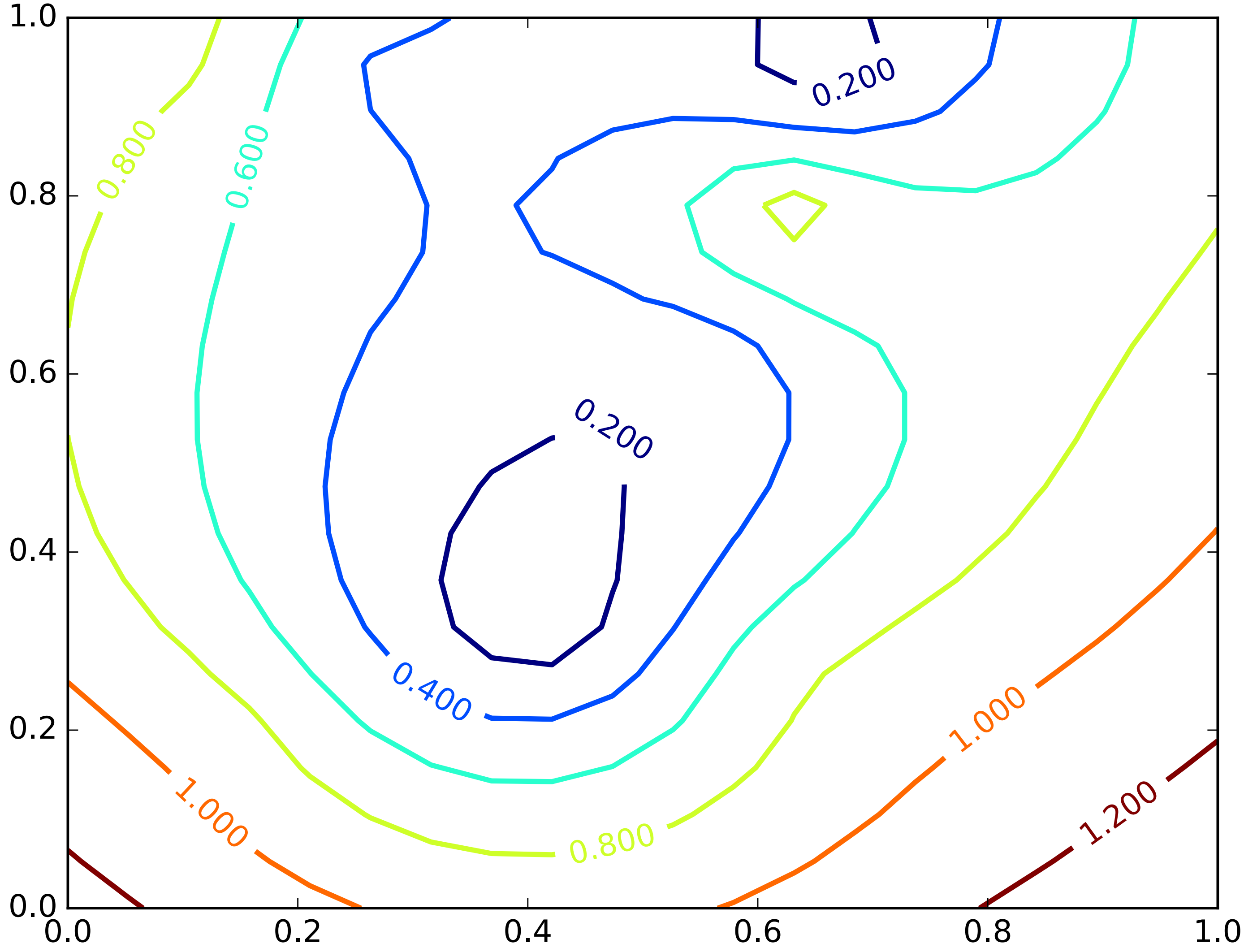
Topographical Maps



Topographical Maps

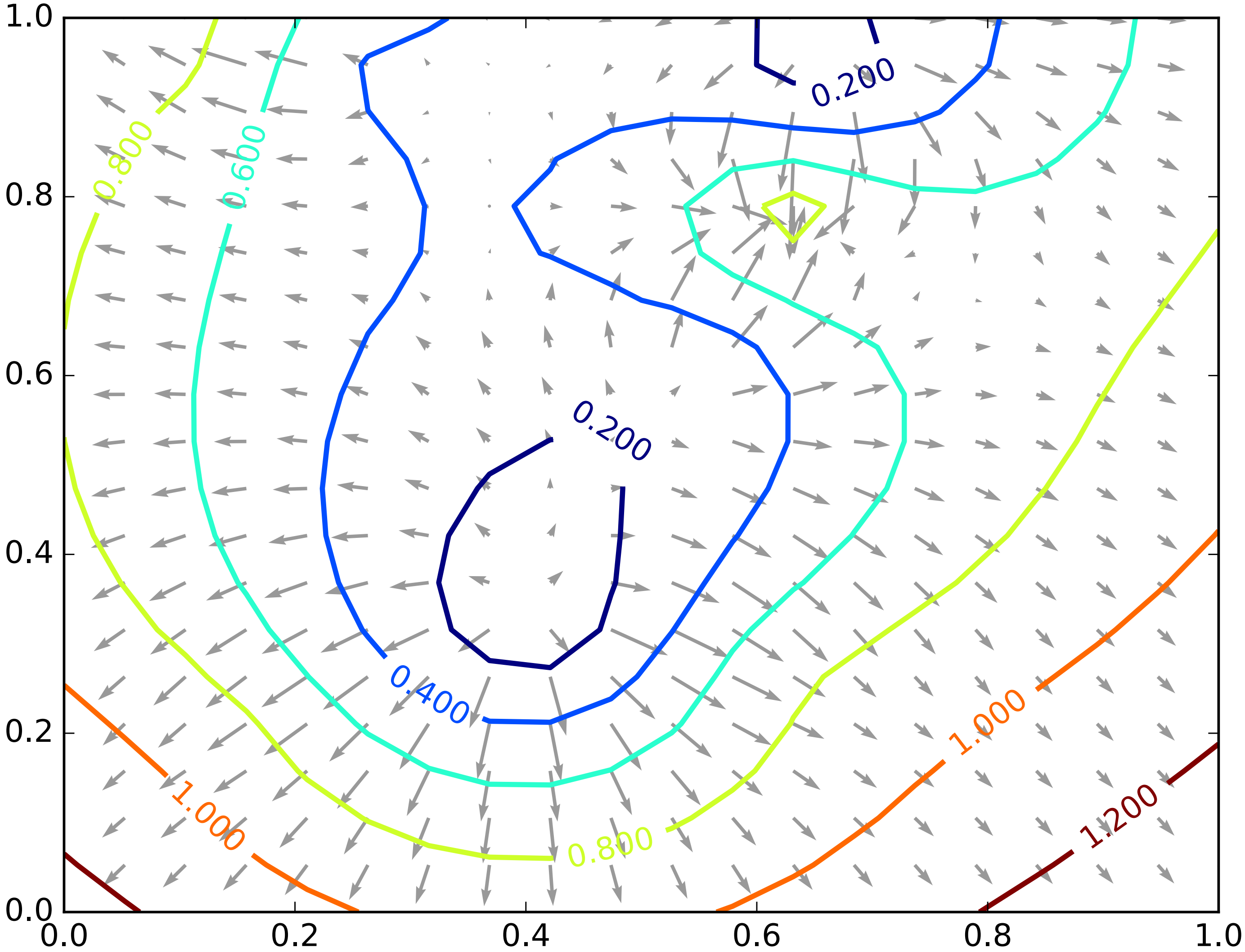


Gradients



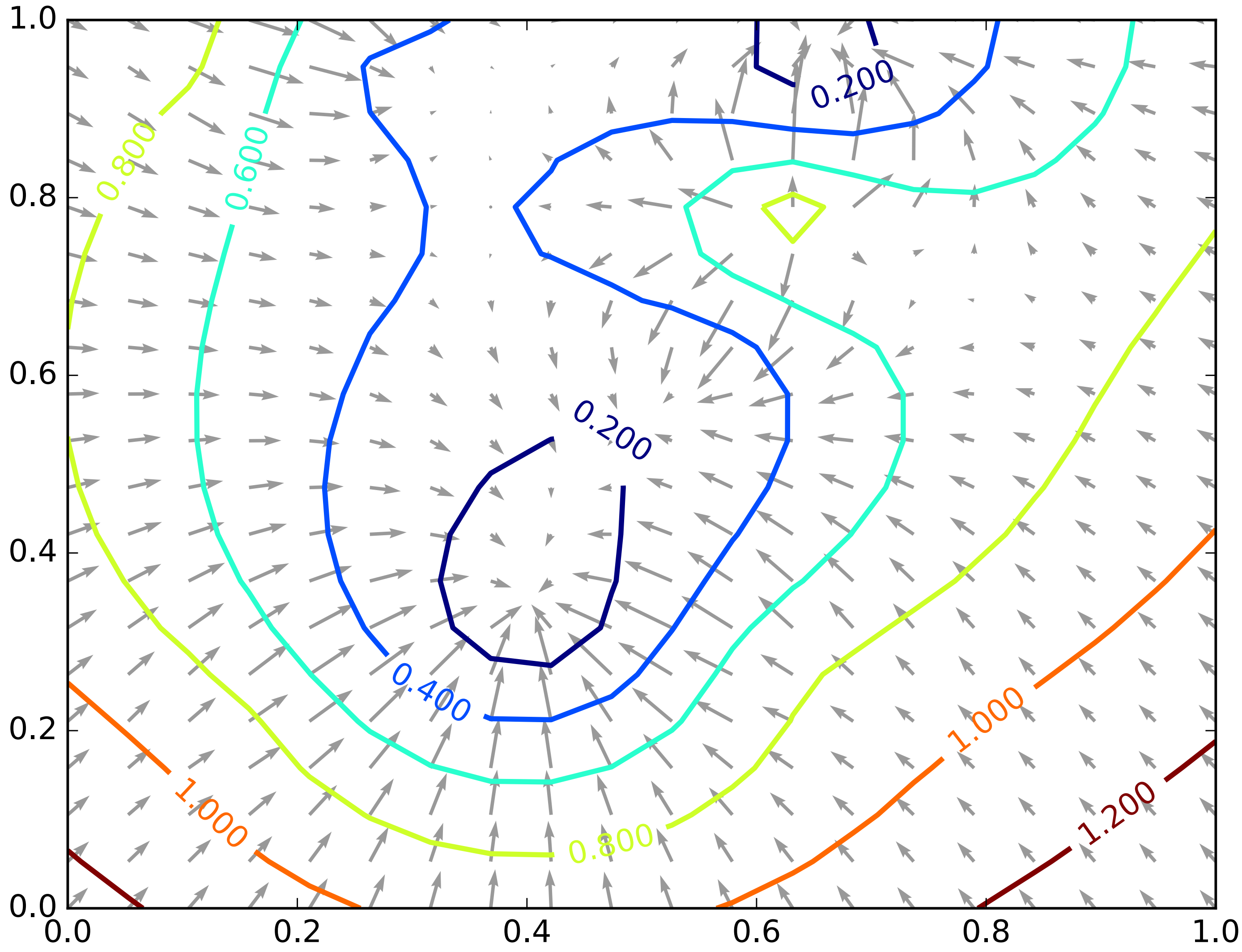
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
0.200
0.200
0.400
0.600
0.800
0.800
1.000
1.000
1.200

Gradients



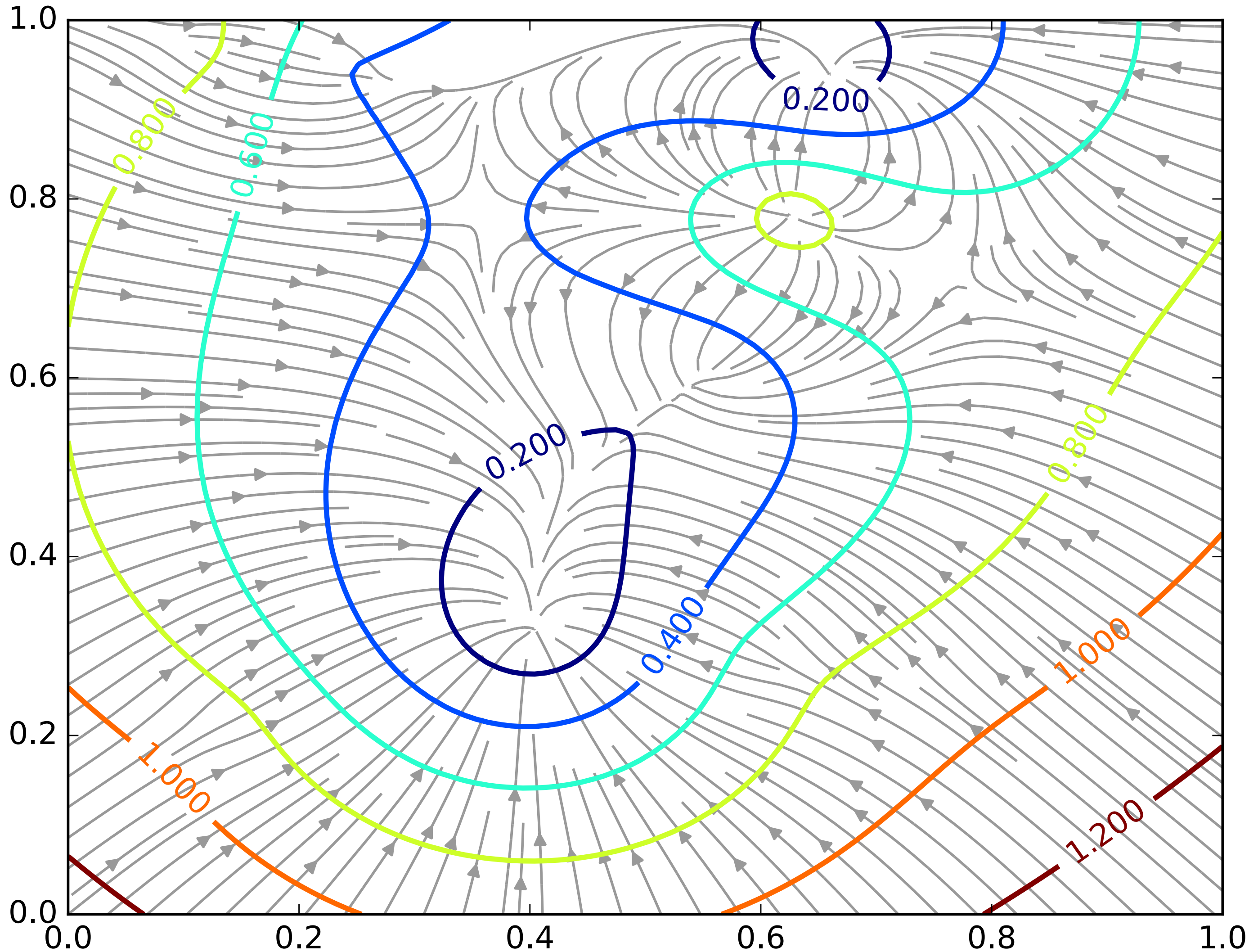
These are the **gradients** that
Gradient **Ascent** would follow.

(Negative) Gradients



These are the **negative** gradients that Gradient **Descent** would follow.

(Negative) Gradient *Paths*

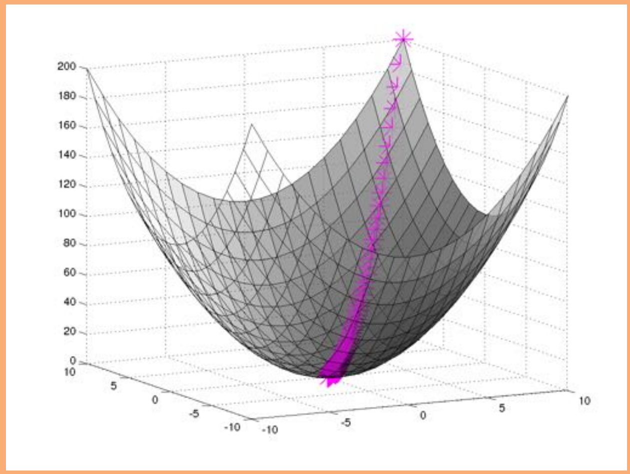


Shown are the **paths** that Gradient Descent would follow if it were making **infinitesimally small steps.**

Gradient Descent

Algorithm 1 Gradient Descent

$$
\begin{array}{l}
\textbf{procedure } \mathrm{GD}(\mathcal{D}, \boldsymbol{\theta}^{(0)}) \\
\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)} \\
\quad \textbf{while } \text{not converged } \textbf{do} \\
\quad\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \\
\quad \textbf{return } \boldsymbol{\theta}
\end{array}
$$



In order to apply GD to Linear Regression all we need is the **gradient** of the objective function (i.e. vector of partial derivatives).

$$
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_N} J(\boldsymbol{\theta}) \end{bmatrix}
$$

Gradient Descent

Algorithm 1 Gradient Descent

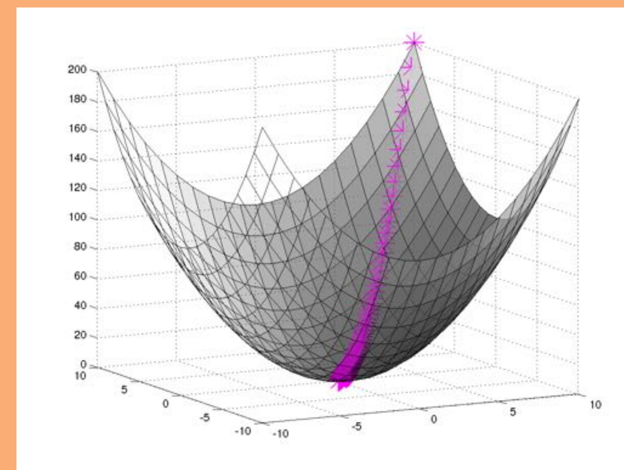
procedure GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)

  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$

  while not converged **do**

    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

  return $\boldsymbol{\theta}$



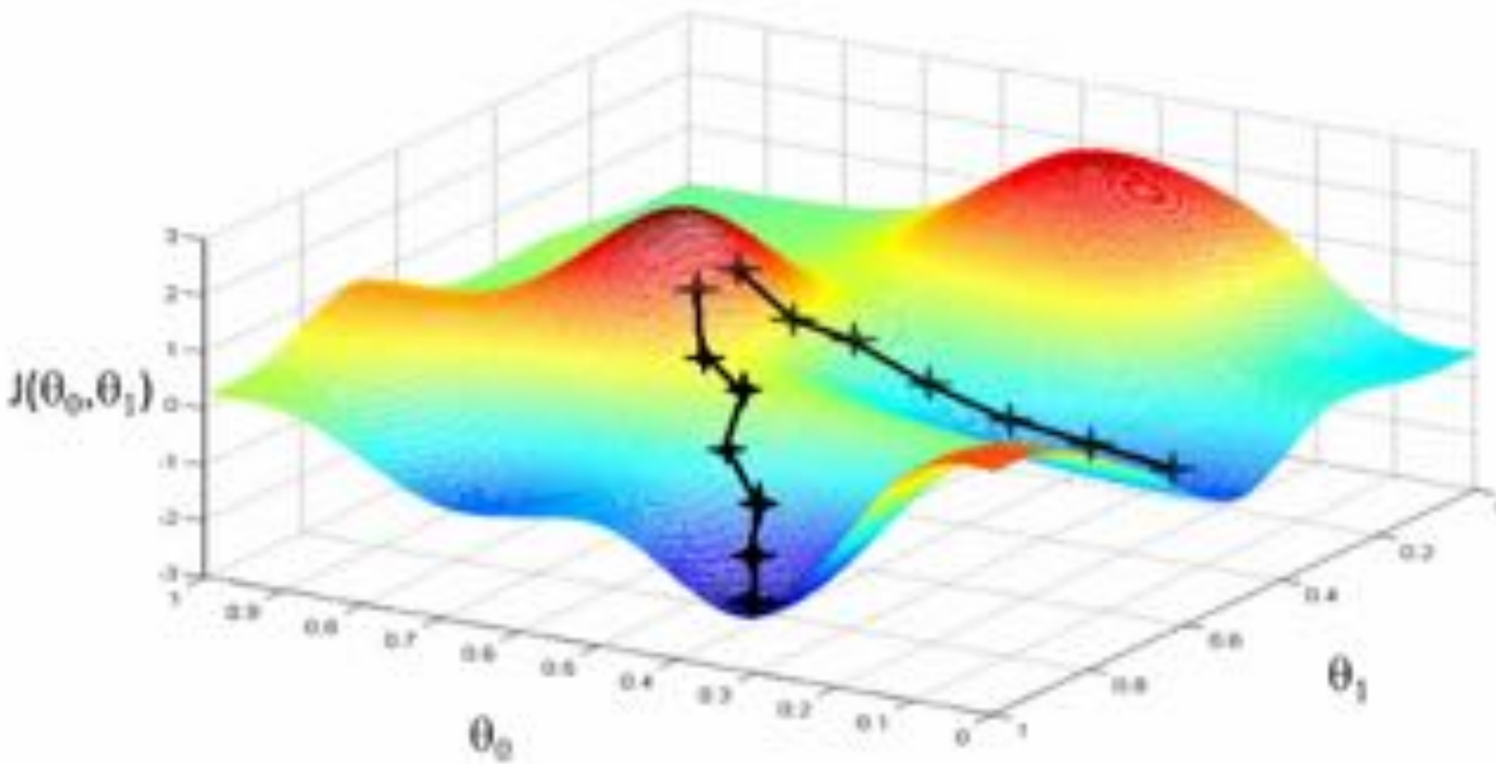
There are many possible ways to detect **convergence**. For example, we could check whether the L2 norm of the gradient is below some small tolerance.

$$||\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})||_2 \leq \epsilon$$

Alternatively we could check that the reduction in the objective function from one iteration to the next is small.

Pros and cons of gradient descent

- Simple and often quite effective on ML tasks
- Often very scalable
- Only applies to smooth functions (differentiable)
- Might find a local minimum, rather than a global one

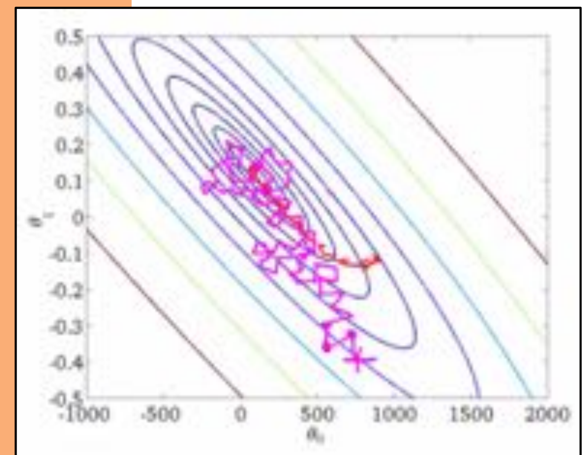


Slide courtesy of William Cohen

Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

procedure SGD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
    while not converged **do**
        for $i \in \text{shuffle}(\{1, 2, \ldots, N\})$ **do**
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
    return $\boldsymbol{\theta}$



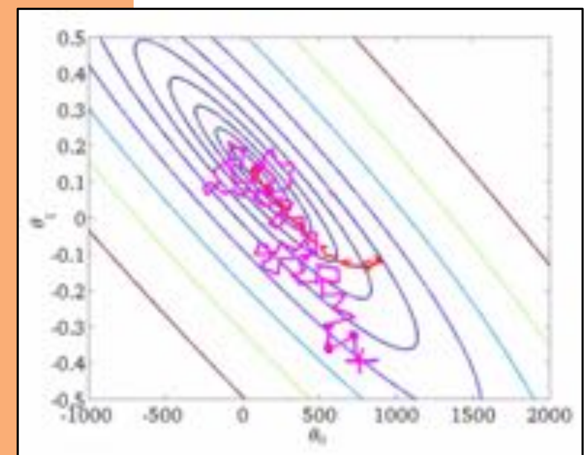
We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

procedure SGD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
    while not converged **do**
        for $i \in \text{shuffle}(\{1, 2, \ldots, N\})$ **do**
            for $k \in \{1, 2, \ldots, K\}$ **do**
                $\theta_k \leftarrow \theta_k - \lambda \frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta})$
    return $\boldsymbol{\theta}$



We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \textstyle\sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

Stochastic Gradient Descent (SGD)

Whiteboard

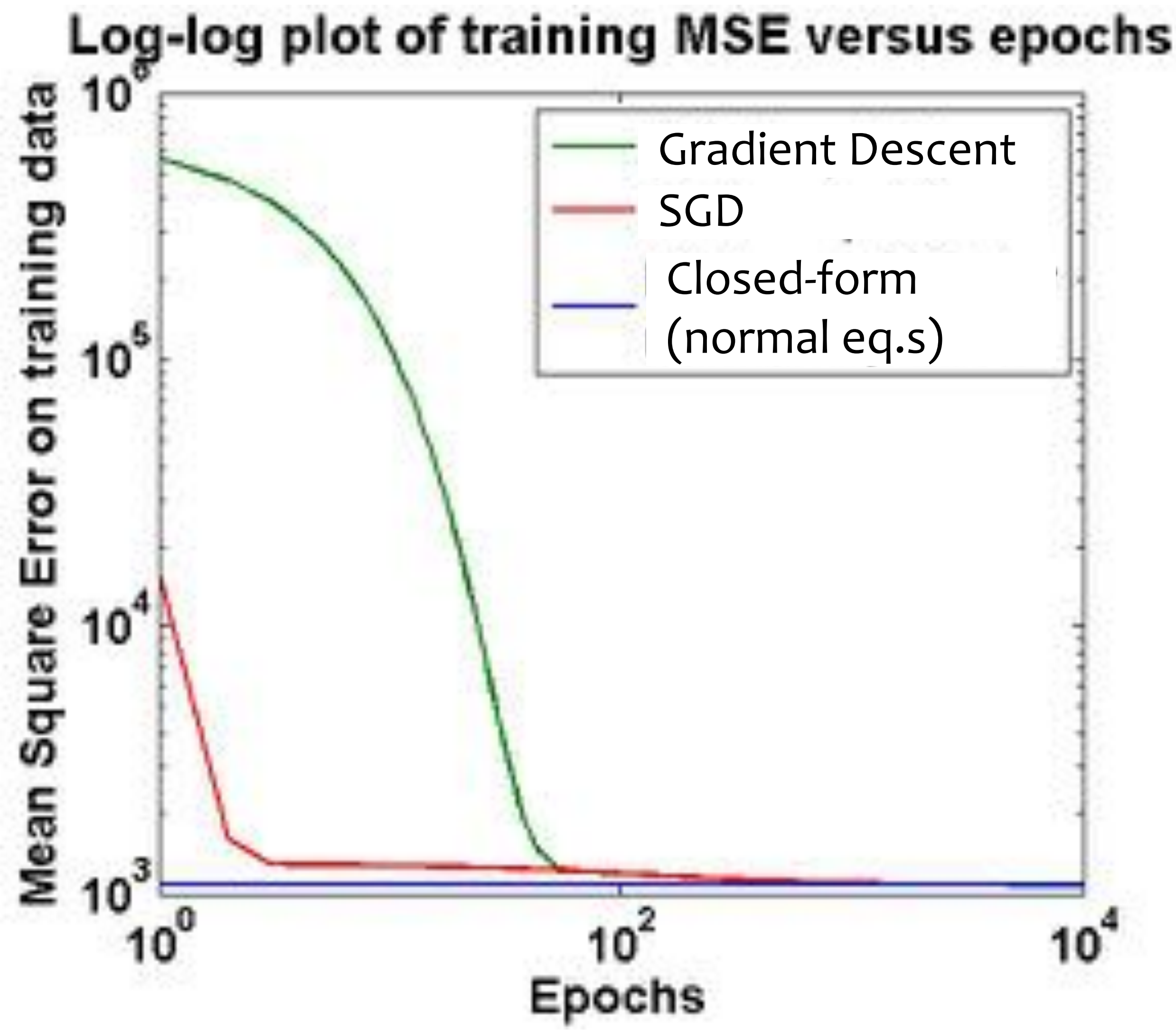
– Expectations of gradients
– Algorithm
– Mini-batches
– Details: mini-batches, step size, stopping criterion
– Problematic cases for SGD

Convergence

Whiteboard

- Comparison of Newton's method, Gradient Descent, SGD
- Asymptotic convergence
- Convergence in practice

Convergence Curves



- *Def*: an **epoch** is a single pass through the training data

1. For GD, only **one update** per epoch
2. For SGD, ***N* updates** per epoch
   N = (# train examples)

- SGD reduces MSE much more rapidly than GD
- For GD / SGD, training MSE is initially large due to uninformed initialization

Figure adapted from Eric P. Xing

Optimization 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

Linear Regression Objectives

You should be able to…

- Design k-NN Regression and Decision Tree Regression
- Implement learning for Linear Regression using three optimization techniques: (1) closed form, (2) gradient descent, (3) stochastic gradient descent
- Choose a Linear Regression optimization technique that is appropriate for a particular dataset by analyzing the tradeoff of computational complexity vs. convergence speed
- Distinguish the three sources of error identified by the bias-variance decomposition: bias, variance, and irreducible error.