CS 66: Machine Learning

Prof. Sara Mathieson

Spring 2019

Admin

- Office hours **TODAY**! 12:30-2pm (Sci Center 249)
- Lab 2 due **Tuesday night**
- (optional) **Lab 3** partner form by Tues night

Outline for February 11

- Lab 1 feedback + examples
- Linear regression analytic solution
- Polynomial regression
- Regularization
- Linear regression for classification

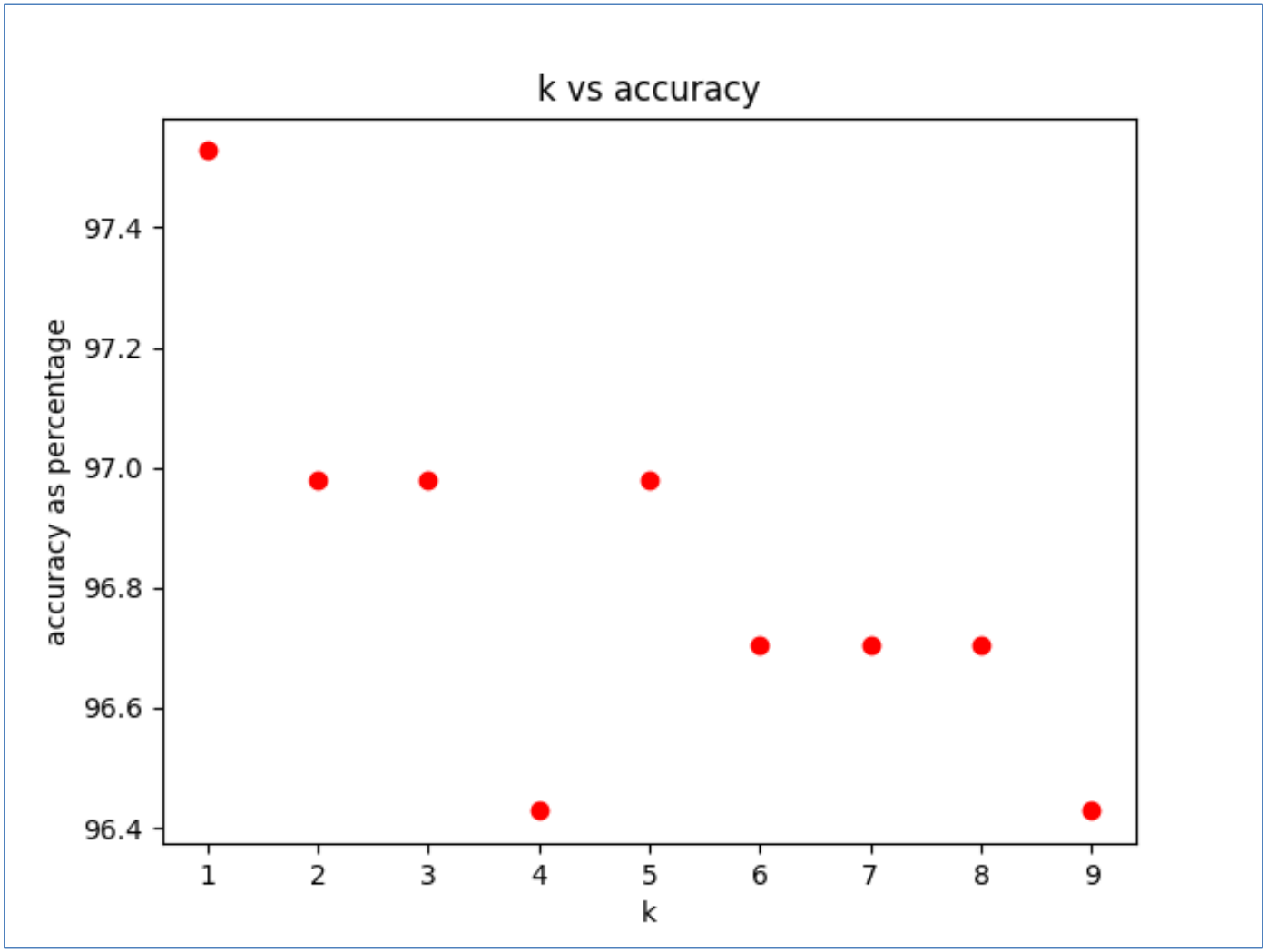
Outline for February 11

- Lab 1 feedback + examples
- Linear regression analytic solution
- Polynomial regression
- Regularization
- Linear regression for classification

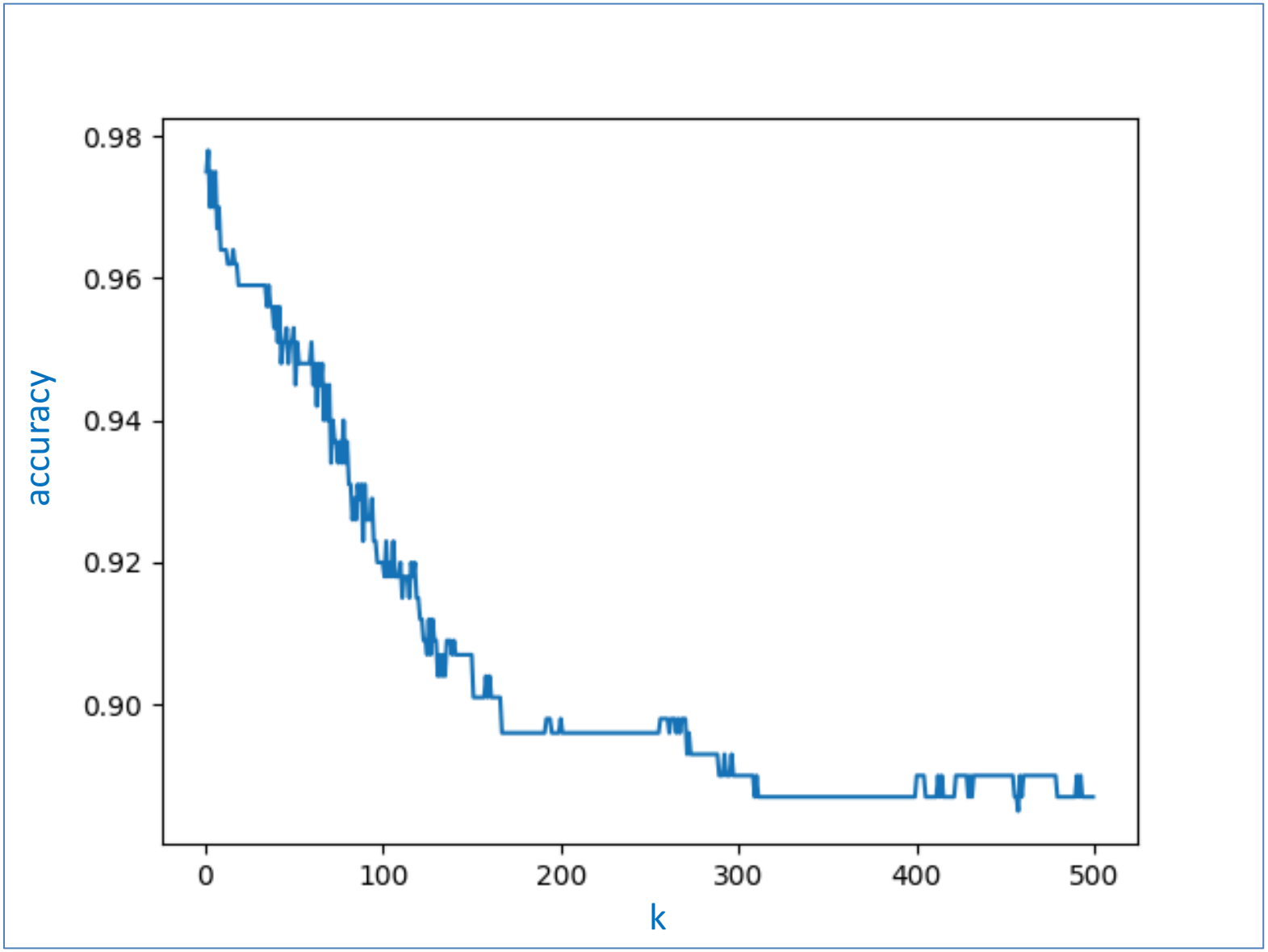
Lab 1 Extensions

- Gabriel & Haochen: when considering all 10 classes, k=3 was the best
- Rick & Keton: added command line arguments for which digits
- Hailie & Zach: added command line arguments for which digits

Lab 1: Kwate & Greg

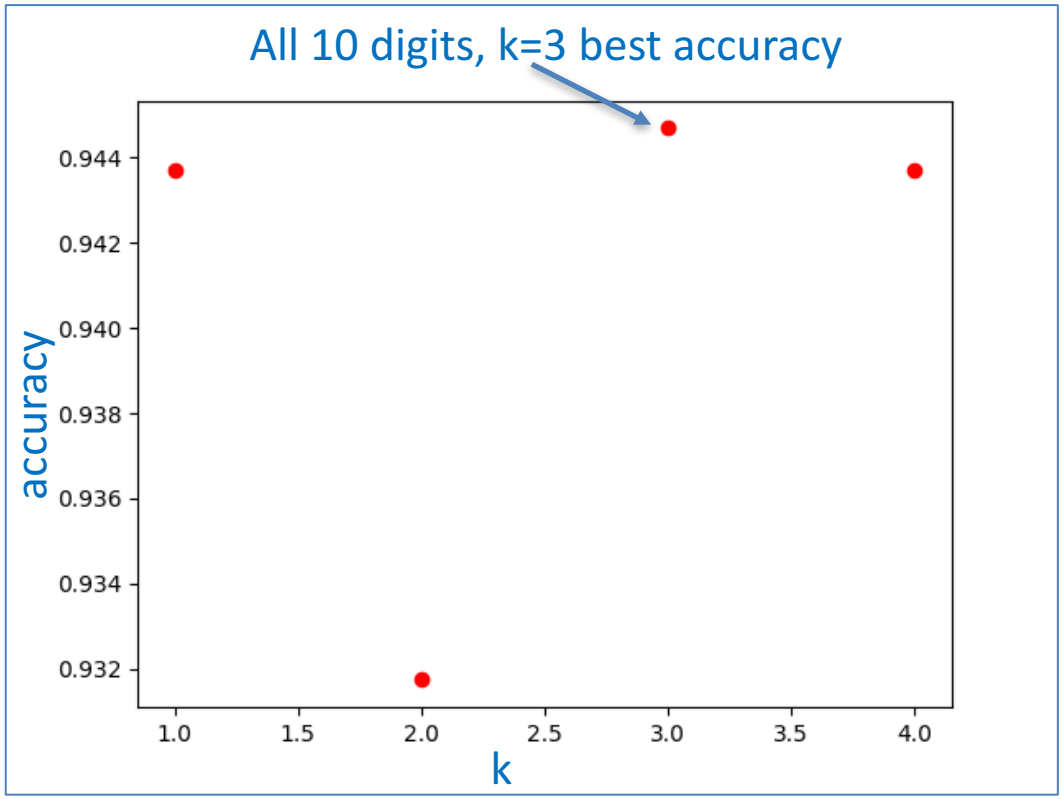


Lab 1: Nav



Lab 1: James & Daniel

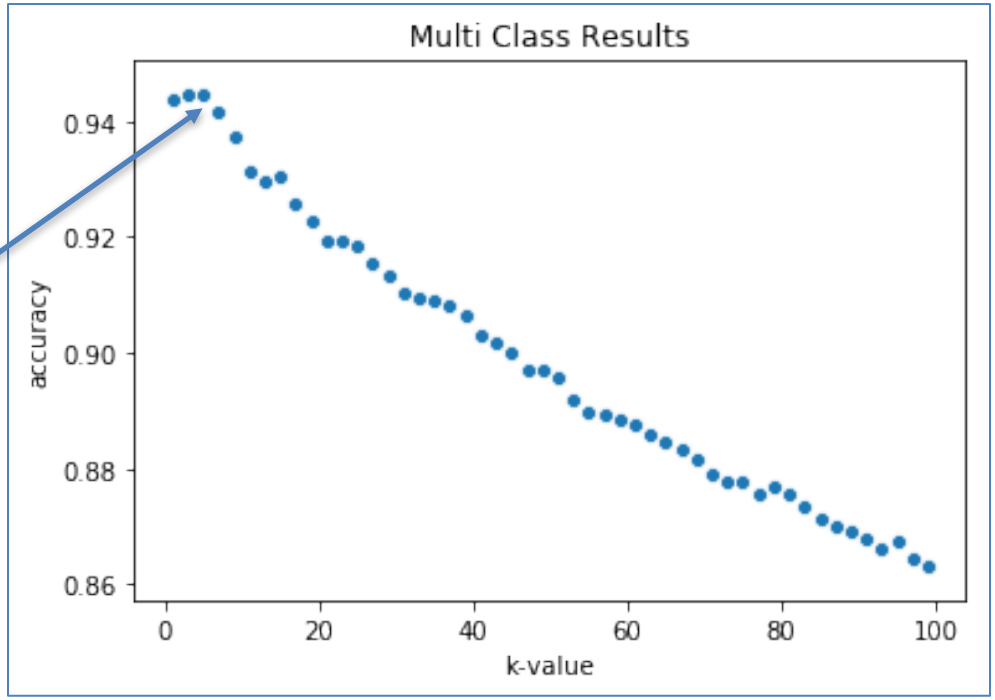
- Allow user to pick digits
- Allow user to pick range of k
- Shows accuracy per digit

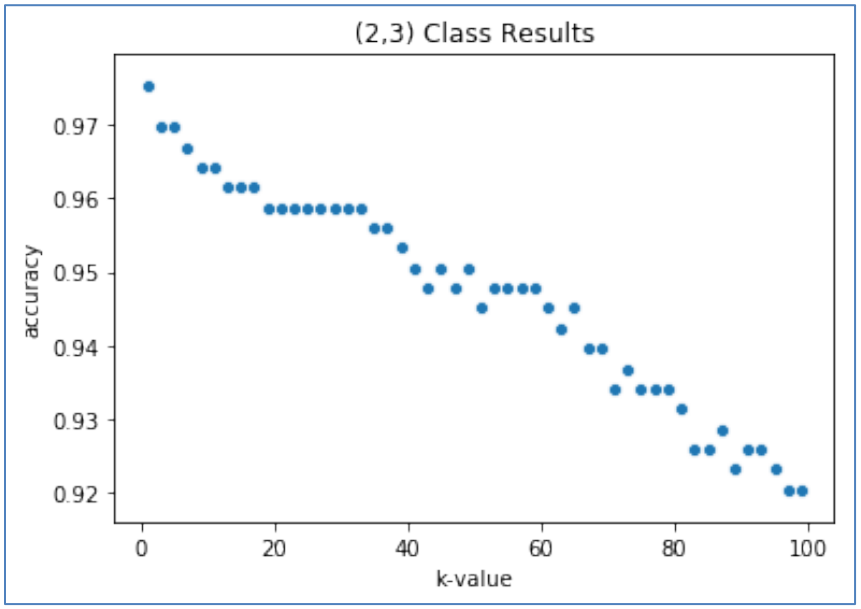



```
k = 1, accuracy = 0.943697, time = 636.49
   Numbers meant to be 0 had 0.988858 accuracy
   Numbers meant to be 1 had 0.965909 accuracy
   Numbers meant to be 2 had 0.924242 accuracy
   Numbers meant to be 3 had 0.927711 accuracy
   Numbers meant to be 4 had 0.910000 accuracy
   Numbers meant to be 5 had 0.906250 accuracy
   Numbers meant to be 6 had 0.964706 accuracy
   Numbers meant to be 7 had 0.945578 accuracy
   Numbers meant to be 8 had 0.891566 accuracy
   Numbers meant to be 9 had 0.954802 accuracy
k = 2, accuracy = 0.931739, time = 623.83
   Numbers meant to be 0 had 0.988858 accuracy
   Numbers meant to be 1 had 0.981061 accuracy
   Numbers meant to be 2 had 0.898990 accuracy
   Numbers meant to be 3 had 0.927711 accuracy
   Numbers meant to be 4 had 0.865000 accuracy
   Numbers meant to be 5 had 0.856250 accuracy
   Numbers meant to be 6 had 0.941176 accuracy
   Numbers meant to be 7 had 0.904762 accuracy
   Numbers meant to be 8 had 0.921687 accuracy
   Numbers meant to be 9 had 0.949153 accuracy
k = 3, accuracy = 0.944694, time = 628.53
   Numbers meant to be 0 had 0.988858 accuracy
   Numbers meant to be 1 had 0.977273 accuracy
   Numbers meant to be 2 had 0.924242 accuracy
   Numbers meant to be 3 had 0.921687 accuracy
   Numbers meant to be 4 had 0.910000 accuracy
   Numbers meant to be 5 had 0.900000 accuracy
   Numbers meant to be 6 had 0.958824 accuracy
   Numbers meant to be 7 had 0.938776 accuracy
   Numbers meant to be 8 had 0.915663 accuracy
   Numbers meant to be 9 had 0.949153 accuracy
k = 4, accuracy = 0.943697, time = 662.63
   Numbers meant to be 0 had 0.988858 accuracy
   Numbers meant to be 1 had 0.977273 accuracy
   Numbers meant to be 2 had 0.919192 accuracy
   Numbers meant to be 3 had 0.933735 accuracy
   Numbers meant to be 4 had 0.905000 accuracy
   Numbers meant to be 5 had 0.887500 accuracy
   Numbers meant to be 6 had 0.952941 accuracy
   Numbers meant to be 7 had 0.925170 accuracy
   Numbers meant to be 8 had 0.927711 accuracy
   Numbers meant to be 9 had 0.954802 accuracy
```

Lab 1: Dylan

All 10 digits, k=3 best accuracy





	Predicted No	Predicted Yes
Acutal No	192	6
Actual Yes	3	163

Lab 1: how to make k-NN faster?

- If we have n training examples and m test examples, k-NN is $O[m(n + n\log n)]$, i.e. not great (haven't considered p)

- Don't need to sort all distances – for small k, we can find the top k neighbors in linear time

- Save matrix of pair-wise distances across k
- Use less of the training data

- Put each training example in a "zone" or "cluster". For each test example, identify cluster and only consider neighbors within that cluster

Outline for February 11

- Lab 1 feedback + examples
- Linear regression analytic solution
- Polynomial regression
- Regularization
- Linear regression for classification

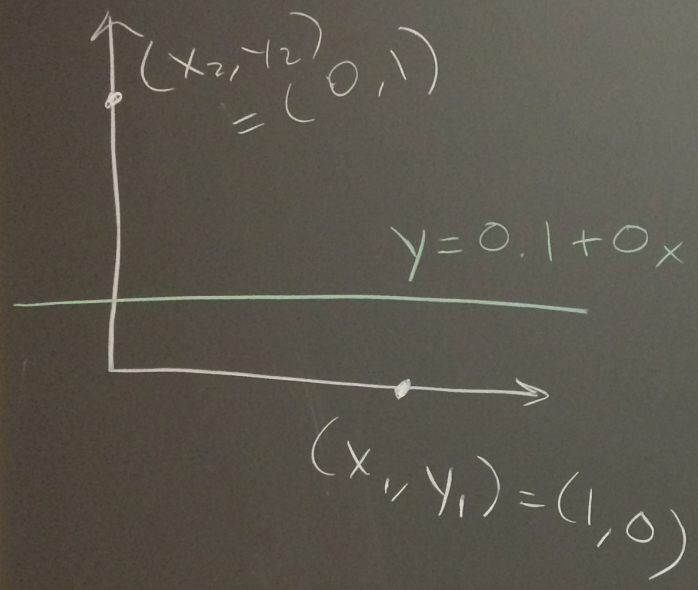
Stochastic Gradient Descent

- SGD pseudocode

```
while J(b) not changing and max iter not reached:
    # shuffle data points
    for i = 1, 2, ··· n:
        for j = 0, 1, ··· , p:
            b_j ← b_j − α(b^T x_i − y_i) x_ij
```

All at once!

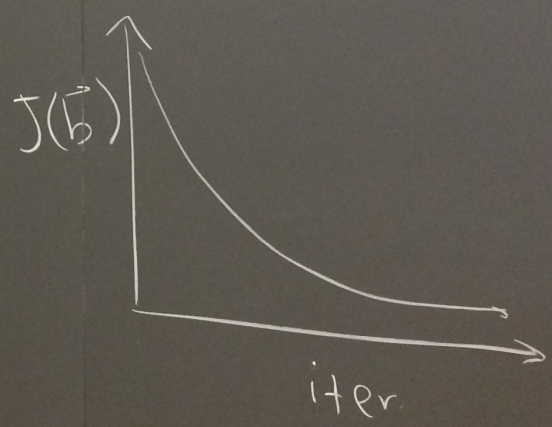
$$J(\vec{b}) = \frac{1}{2}\sum_{i=1}^{n}\left(\vec{b}^T\vec{x}_i - y_i\right)^2$$



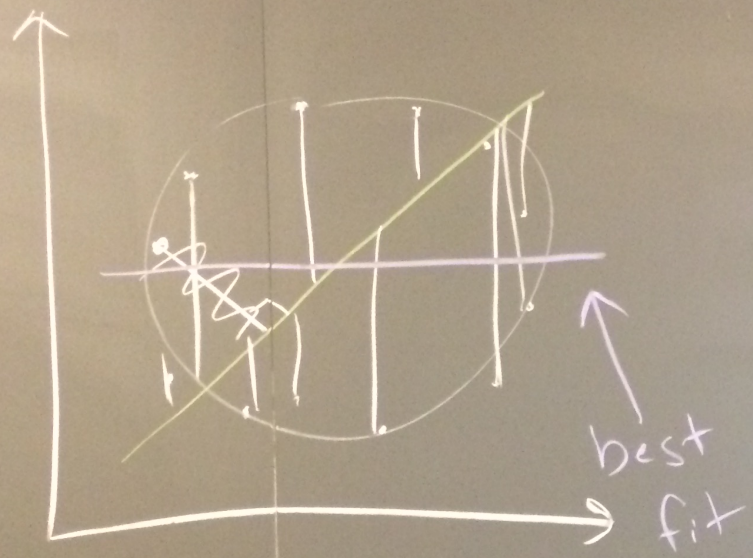
$$J(\vec{b}) = \frac{1}{2}\left[(0.1-0)^2 + (0.1-1)^2\right]$$

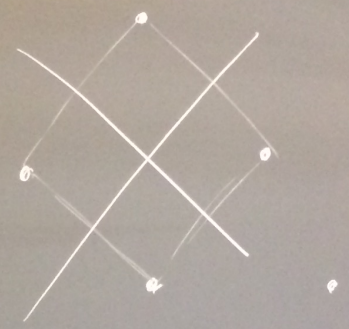
$$= \frac{1}{2}(0.01 + 0.81)$$

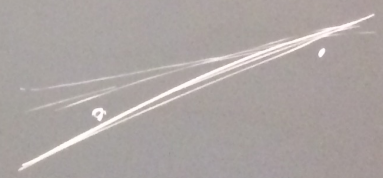
$$\boxed{J(\vec{b}) = 0.41}$$

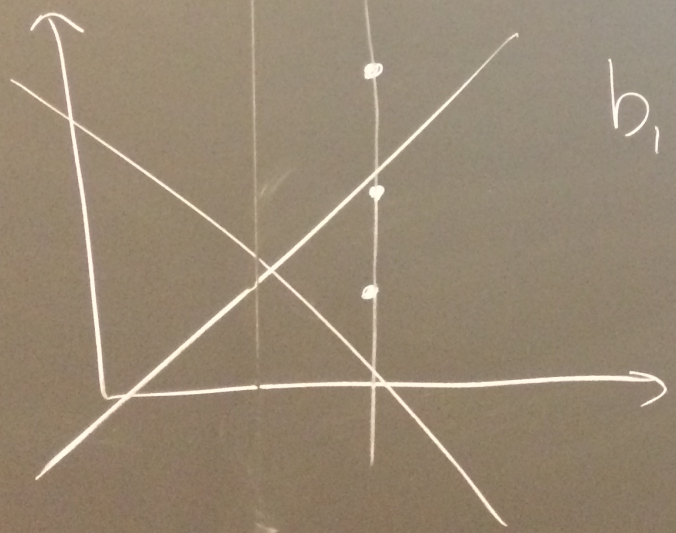


Handout 2, question 2(b)







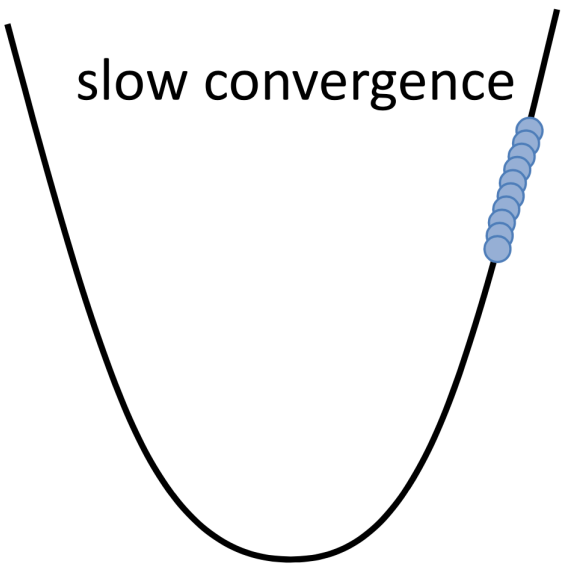


$$b_1 = \frac{Cov(x,y)}{Var(x)}$$

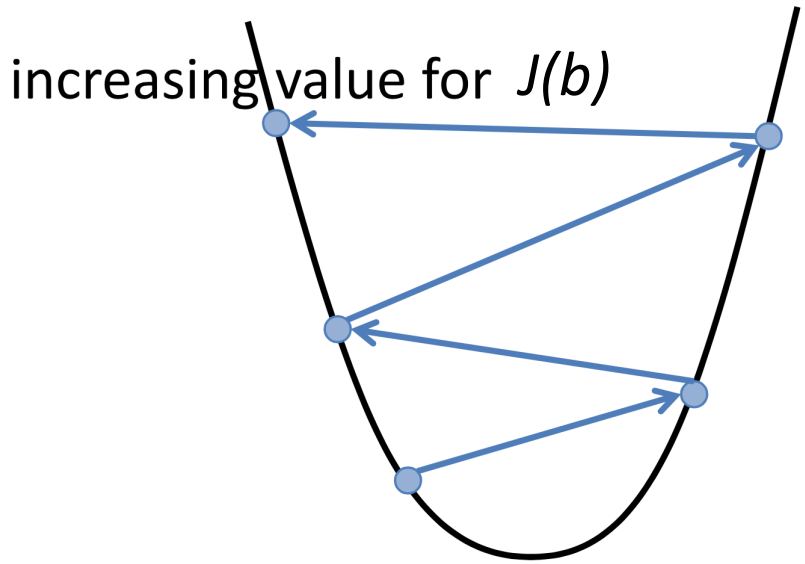
AHHH!

Choosing step size α

α too small



α too large



- may overshoot minimum
- may fail to converge (may even diverge)

Slide: modified from Jessica Wu
Based on slide by Eric Eaton
[Originally by Andrew Ng]

Analytic Solution

solve!

$$\vec{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_p \end{bmatrix}_{(p+1)\times 1} \qquad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & & & & \\ 1 & & & & \\ \vdots & & & & \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}_{n\times(p+1)}$$

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n\times 1}$$

$$(AB)^T = B^T A^T$$

$$J(\vec{b}) = \frac{1}{2}\sum_{i=1}^{n}\left(\vec{b}^T\vec{x}_i - y_i\right)^2$$

$$= \frac{1}{2}\underbrace{\left(\underbrace{X\vec{b}}_{n\times 1} - \underbrace{\vec{y}}_{n\times 1}\right)^T}_{1\times n}\underbrace{\left(X\vec{b} - \vec{y}\right)}_{n\times 1}$$

$$= \frac{1}{2}\left[\vec{b}^T X^T X\vec{b} - \vec{b}^T X^T\vec{y} - \vec{y}^T X\vec{b} + \vec{y}^T\vec{y}\right]$$

$$= \frac{1}{2}\left[\vec{b}^T X^T X\vec{b} - 2\vec{b}^T X^T\vec{y} + \vec{y}^T y\right]$$

$$\nabla_{\vec{b}} J(\vec{b}) = \underbrace{X^T X \vec{b} - X^T y = 0}_{(X^T X)^{-1}}$$

$$\cancel{(X^T X)^{-1}} \underbrace{\cancel{X^T X}} \vec{b} = X^T \vec{y}$$

$(p+1) \times n$

$n \times (p+1)$

$$\boxed{\hat{\vec{b}} = \underbrace{(X^T X)^{-1}}_{\text{like } \frac{1}{Var(x)}} \underbrace{X^T \vec{y}}_{\text{like } Cov(x,y)}} \quad \star$$

not invertible

$\Rightarrow$ no solution

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$X^T = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

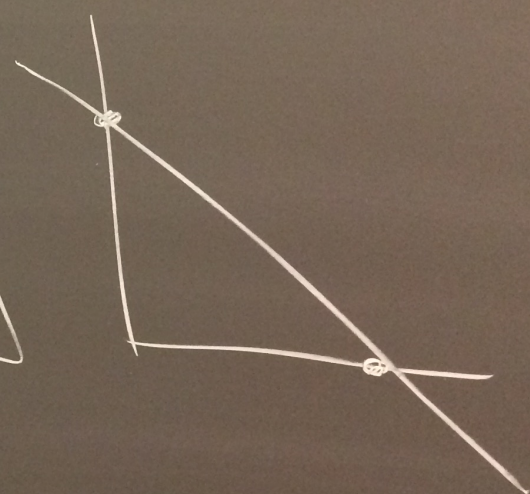
$$\boxed{\text{find } \vec{b}}$$

$$\vec{b} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$= \frac{1}{2-1} \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\boxed{y = 1 - 1 \cdot x}$$



Handout 2, question 2(c)

Pros and Cons

Gradient Descent

- requires multiple iterations
- need to choose α
- works well when p is large
- can support online learning

Normal Equations

- non-iterative
- no need for α
- slow if p is large
  - matrix inversion is $O(p^3)$

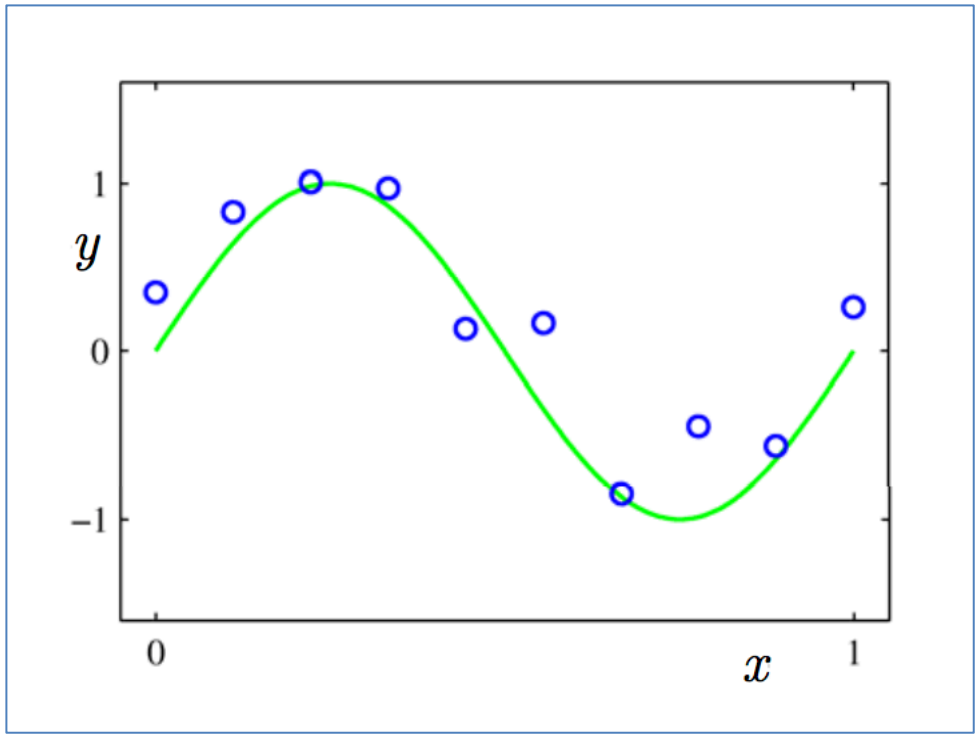
Slide: modified from Jessica Wu
Based on slide by Eric Eaton

Outline for February 11

- Lab 1 feedback + examples
- Linear regression analytic solution
- Polynomial regression
- Regularization
- Linear regression for classification

Polynomial Regression

- Can be thought of as regular linear regression with a change of basis



Plot: Jessica Wu

Polynomial Regression

$$\vec{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_p \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ x_1 \\ x_1^2 \\ x_2 \\ x_2^2 \\ x_3 \\ x_3^2 \\ \vdots \\ x_p \\ x_p^2 \end{bmatrix} \Bigg\} \; 2p+1 \text{ coefficients}$$

degree = d

$\Rightarrow$ $p \cdot d + 1$ features

high?

bad!