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

accuracy

$\text{accuracy}$

$k$

$k$
Lab 1: James & Daniel

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

All 10 digits, $k=3$ best accuracy

$k = 1$, 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 0.988850 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

<table>
<thead>
<tr>
<th>Predicted No</th>
<th>Predicted Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acutal No</td>
<td>192</td>
</tr>
<tr>
<td>Actual Yes</td>
<td>3</td>
</tr>
</tbody>
</table>

![Graph showing multi-class results](image)

![Graph showing (2,3) class results](image)

![Images of handwritten digits](image)
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

\[
\text{while } J(b) \text{ not changing and max iter not reached:} \\
\quad \# \text{ shuffle data points} \\
\quad \text{for } i = 1, 2, \cdots n: \\
\quad \quad \text{for } j = 0, 1, \cdots, p: \\
\quad \quad \quad b_j \leftarrow b_j - \alpha (b^T x_i - y_i) x_{ij} \quad \quad \text{All at once!}
\]
\[ J(\hat{b}) = \frac{1}{2} \sum_{i=1}^{n} (\hat{b}^T x_i - y_i)^2 \]

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

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

\[ J(\hat{b}) = 0.41 \]

Handout 2, question 2(b)
Choosing step size $\alpha$

$\alpha$ too small
- slow convergence

$\alpha$ too large
- increasing value for $J(b)$
- may overshoot minimum
- may fail to converge (may even diverge)
Analytic Solution

\[ b = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_p \end{bmatrix} \]

\[ X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \]

\[ \hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} \]

\[ (AB)^T = B^T A^T \]

\[ J(\hat{b}) = \frac{1}{2} \sum_{i=1}^{n} (b^T x_i - \hat{y}_i)^2 \]

\[ = \frac{1}{2} (X\hat{b} - \hat{y})^T (X\hat{b} - \hat{y}) \]

\[ = \frac{1}{2} \begin{bmatrix} \hat{y}^T \hat{y} - \hat{b}^T X^T \hat{y} + \hat{y}^T \hat{b}^T X \hat{y} - \hat{y}^T \hat{y} \end{bmatrix} \]
\[ \nabla_b J(\hat{b}) = X^T X \hat{b} - X^T y = 0 \]

\[ (X^T X)^{-1} X^T \tilde{y} \]

\[ (p+1) \times n \]

\[ n \times (p+1) \]

\[ \hat{b} = (X^T X)^{-1} X^T \tilde{y} \]

\[ \not\exists \text{ invertible} \]

\[ \Rightarrow \text{ no solution} \]
Handout 2, question 2(c)

\[
X = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad Y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

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

Find \( \vec{b} \)

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

\[
\vec{b} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

\[
= \frac{1}{z^2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\]

\[
= \begin{bmatrix} 1 \\ -1 \end{bmatrix}
\]

\[
Y = 1 - 1 \cdot x
\]
Pros and Cons

Gradient Descent

• requires multiple iterations
• need to choose $\alpha$
• works well when $p$ is large
• can support online learning

Normal Equations

• non-iterative
• no need for $\alpha$
• slow if $p$ is large
  – matrix inversion is $O(p^3)$
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
Polynomial Regression

\[ \vec{X} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ x_1 \\ x_1^2 \\ \vdots \\ x_1^{2p+1} \\ x_2 \\ x_2^2 \\ \vdots \\ x_2^{2p+1} \\ \vdots \\ x_p \end{bmatrix} \]

Degree = \( d \) \Rightarrow p \cdot d + 1 features