(Linear Models) + Feature Engineering + Regularization
**Q**: I think you graded seven different questions incorrectly. Should I put them all in on Gradescope regrade request and submit that?

**A**: Please, no. I’d encourage you to watch this tutorial video from Gradescope so you know how to use this important tool.

Reminders

• Practice Problems: Exam 1

• Exam 1
  – Tue, Oct 4, 6:30pm – 8:30pm
  – see Piazza for details

• Homework 4: Logistic Regression
  – Out: Tue, Oct 4
  – Due: Thu, Oct 13 at 11:59pm
PERCEPTRON, LINEAR REGRESSION, AND LOGISTIC REGRESSION
Why is it not “Logistic Classification”?  

Whiteboard  
– Conceptual Change: 2D classification in 3D  
– Why is it called Logistic Regression and not Logistic Classification?
Matching Game

**Question:**
Match the Algorithm to its Update Rule

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Update Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. SGD for Logistic Regression</td>
<td>( h_\theta(x) = p(y</td>
</tr>
<tr>
<td>2. Least Mean Squares</td>
<td>( h_\theta(x) = \theta^T x )</td>
</tr>
<tr>
<td>3. Perceptron</td>
<td>( h_\theta(x) = \text{sign}(\theta^T x) )</td>
</tr>
<tr>
<td>4.</td>
<td>( \theta_k \leftarrow \theta_k + (h_\theta(x^{(i)}) - y^{(i)}) )</td>
</tr>
<tr>
<td>5.</td>
<td>( \theta_k \leftarrow \theta_k + \frac{1}{1 + \exp \lambda (h_\theta(x^{(i)}) - y^{(i)})} )</td>
</tr>
<tr>
<td>6.</td>
<td>( \theta_k \leftarrow \theta_k + \lambda (h_\theta(x^{(i)}) - y^{(i)})x_k^{(i)} )</td>
</tr>
</tbody>
</table>

**Answer:**
A. 1=5, 2=4, 3=6  
B. 1=5, 2=6, 3=4  
C. 1=6, 2=4, 3=4  
D. 1=5, 2=6, 3=6  
E. 1=6, 2=6, 3=6  
F. 1=6, 2=5, 3=5  
G. 1=5, 2=5, 3=5  
H. 1=4, 2=5, 3=6
Question:

Which of the following is a correct description of SGD for Logistic Regression?

Answer:

At each step (i.e. iteration) of SGD for Logistic Regression we...

A. (1) compute the gradient of the log-likelihood for all examples (2) update all the parameters using the gradient

B. (1) ask Matt for a description of SGD for Logistic Regression, (2) write it down, (3) report that answer

C. (1) compute the gradient of the log-likelihood for all examples (2) randomly pick an example (3) update only the parameters for that example

D. (1) randomly pick a parameter, (2) compute the partial derivative of the log-likelihood with respect to that parameter, (3) update that parameter for all examples

E. (1) randomly pick an example, (2) compute the gradient of the log-likelihood for that example, (3) update all the parameters using that gradient

F. (1) randomly pick a parameter and an example, (2) compute the gradient of the log-likelihood for that example with respect to that parameter, (3) update that parameter using that gradient
Gradient Descent

Algorithm 1 Gradient Descent

1: procedure GD(\(D, \theta^{(0)}\))
2: \(\theta \leftarrow \theta^{(0)}\)
3: while not converged do
4: \(\theta \leftarrow \theta - \gamma \nabla \theta J(\theta)\)
5: return \(\theta\)

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

\[
\nabla_\theta J(\theta) = \begin{bmatrix}
\frac{d}{d\theta_1} J(\theta) \\
\frac{d}{d\theta_2} J(\theta) \\
\vdots \\
\frac{d}{d\theta_M} J(\theta)
\end{bmatrix}
\]
Stochastic Gradient Descent (SGD)

**Algorithm 1** Stochastic Gradient Descent (SGD)

1: procedure SGD(\(D, \theta^{(0)}\))
2: \(\theta \leftarrow \theta^{(0)}\)
3: while not converged do
4:     for \(i \in \text{shuffle}\{1, 2, \ldots, N\}\) do
5:         \(\theta \leftarrow \theta - \gamma \nabla \theta J^{(i)}(\theta)\)
6: return \(\theta\)

We can also apply SGD to solve the MCLE problem for Logistic Regression.

We need a per-example objective:

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

where \(J^{(i)}(\theta) = -\log p_{\theta}(y^i|\mathbf{x}^i)\).
Logistic Regression vs. Perceptron

Question:
True or False: Just like Perceptron, one step (i.e. iteration) of SGD for Logistic Regression will result in a change to the parameters only if the current example is incorrectly classified.

Answer:
BAYES OPTIMAL CLASSIFIER
Bayes Optimal Classifier

Previously, we assumed that our output was generated using a deterministic target function:

\[ c^*(x) \]

Our goal was to learn a hypothesis \( h(x) \) that best approximates \( c^*(x) \).

Now, we assume that our output is sampled from a conditional probability distribution:

\[ p^*(y|x) \]

Our goal is to learn a probability distribution \( p(y|x) \) that best approximates \( p^*(y|x) \).

Question:
How would you design a function \( y = h(x) \) to predict a single label?

Answer:
You’d use the Bayes optimal classifier!
Bayes Optimal Classifier

Whiteboard

– Bayes Optimal Classifier
– Reducible / irreducible error
– Ex: Bayes Optimal Classifier for 0/1 Loss
OPTIMIZATION METHOD #4: MINI-BATCH SGD
Mini-Batch SGD

• **Gradient Descent:**
  Compute true gradient exactly from all N examples

• **Stochastic Gradient Descent (SGD):**
  Approximate true gradient by the gradient of one randomly chosen example

• **Mini-Batch SGD:**
  Approximate true gradient by the average gradient of K randomly chosen examples
Mini-Batch SGD

while not converged: $\theta \leftarrow \theta - \gamma g$

Three variants of first-order optimization:

Gradient Descent: $g = \nabla J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla J^{(i)}(\theta)$

SGD: $g = \nabla J^{(i)}(\theta)$ where $i$ sampled uniformly

Mini-batch SGD: $g = \frac{1}{S} \sum_{s=1}^{S} \nabla J^{(i_s)}(\theta)$ where $i_s$ sampled uniformly $\forall s$
Logistic Regression 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 or multiclass classification
• Prove that the decision boundary of binary logistic regression is linear
• For linear regression, show that the parameters which minimize squared error are equivalent to those that maximize conditional likelihood
FEATURE ENGINEERING
Handcrafted Features

\[
p(y|x) \propto \exp(\Theta_y \cdot f)
\]
Where do features come from?

- **hand-crafted features**
  - Sun et al., 2011
  - Zhou et al., 2005

- **Feature Engineering**
  - First word before M1
  - Second word before M1
  - Bag-of-words in M1
  - Head word of M1
  - Other word in between
  - First word after M2
  - Second word after M2
  - Bag-of-words in M2
  - Head word of M2
  - Bigrams in between
  - Words on dependency path
  - Country name list
  - Personal relative triggers
  - Personal title list
  - WordNet Tags
  - Heads of chunks in between
  - Path of phrase labels
  - Combination of entity types

- **Feature Learning**
Where do features come from?

- **Feature Engineering**
  - Hand-crafted features
    - Sun et al., 2011
    - Zhou et al., 2005
  - Word embeddings
    - Mikolov et al., 2013

- **Feature Learning**
  - CBOW model in Mikolov et al. (2013)
  - Unsupervised learning

Input (context words) → Embedding → Missing word

Similar words, similar embeddings

<table>
<thead>
<tr>
<th>Word</th>
<th>Embeddings</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>0.11 .23 ... -0.45</td>
</tr>
<tr>
<td>dog</td>
<td>0.13 .26 ... -0.52</td>
</tr>
</tbody>
</table>
Where do features come from?

**Feature Engineering**

- **Feature Learning**
  - **CNN**
    - Convolutional Neural Networks 
    - (Collobert and Weston 2008)
    - The [movie] showed [wars]
  - **RAE**
    - Recursive Auto Encoder 
    - (Socher 2011)
    - The [movie] showed [wars]

**Word Embeddings**
- Zhou et al., 2005
- Mikolov et al., 2013

**String Embeddings**
- Socher, 2011
- Collobert & Weston, 2008
Where do features come from?

- **Hand-crafted features**
  - Sun et al., 2011
  - Zhou et al., 2005

- **Word embeddings**
  - Mikolov et al., 2013
  - Socher, 2011

- **String embeddings**
  - Collobert & Weston, 2008

- **Tree embeddings**
  - Socher et al., 2013
  - Hermann & Blunsom, 2013

The diagram illustrates the process of feature creation, with arrows pointing from hand-crafted features to word embeddings, string embeddings, and tree embeddings, suggesting a progression from more manual to more automated methods.
Where do features come from?

- **Hand-crafted features**
  - Sun et al., 2011
  - Zhou et al., 2005

- **Word embeddings**
  - Turian et al., 2010
  - Koo et al., 2008
  - Mikolov et al., 2013

- **Tree embeddings**
  - Socher et al., 2013
  - Hermann & Blunsom, 2013

- **String embeddings**
  - Socher, 2011
  - Collobert & Weston, 2008

- **Word embedding features**
  - Sun et al., 2011
  - Koo et al., 2008

**Refine embedding features with semantic/syntactic info**
Where do features come from?

- **hand-crafted features**
  - Sun et al., 2011
  - Zhou et al., 2005

- **word embedding features**
  - Turian et al., 2010
  - Koo et al., 2008
  - Hermann et al., 2014

- **best of both worlds?**
  - Mikolov et al., 2013
  - Socher et al., 2013
  - Sun et al., 2011
  - Koo et al., 2008
  - Hermann & Blunsom, 2013

- **tree embeddings**
  - Socher et al., 2013

- **string embeddings**
  - Socher, 2011
  - Collobert & Weston, 2008
Feature Engineering for NLP

Suppose you build a logistic regression model to predict a part-of-speech (POS) tag for each word in a sentence.

What features should you use?

The movie I watched depicted hope
Per-word Features:

<table>
<thead>
<tr>
<th>Feature</th>
<th>$x^{(1)}$</th>
<th>$x^{(2)}$</th>
<th>$x^{(3)}$</th>
<th>$x^{(4)}$</th>
<th>$x^{(5)}$</th>
<th>$x^{(6)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>is-capital($w_i$)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>endswith($w_i$,”e”)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>endswith($w_i$,”d”)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>endswith($w_i$,”ed”)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$w_i$ == “aardvark”</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_i$ == “hope”</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The movie I watched depicted hope
Feature Engineering for NLP

Context Features:

<table>
<thead>
<tr>
<th></th>
<th>$x^{(1)}$</th>
<th>$x^{(2)}$</th>
<th>$x^{(3)}$</th>
<th>$x^{(4)}$</th>
<th>$x^{(5)}$</th>
<th>$x^{(6)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_i$ == “watched”</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_{i+1}$ == “watched”</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_{i-1}$ == “watched”</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$w_{i+2}$ == “watched”</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_{i-2}$ == “watched”</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The movie I watched depicted hope
Feature Engineering for NLP

Context Features:

\[
\begin{align*}
\ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots \\
\text{w}_i &= "I" \\
\text{w}_{i+1} &= "I" \\
\text{w}_{i-1} &= "I" \\
\text{w}_{i+2} &= "I" \\
\text{w}_{i-2} &= "I" \\
\ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots
\end{align*}
\]

\[
\begin{array}{cccccc}
  x^{(1)} & x^{(2)} & x^{(3)} & x^{(4)} & x^{(5)} & x^{(6)} \\
  \ldots & \ldots & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 1 & 0 & 0 \\
  1 & 0 & 0 & 0 & 0 & 0 \\
  \ldots & \ldots & \ldots & \ldots & \ldots & \ldots
\end{array}
\]

\[
\begin{array}{cccccc}
  \text{deter.} & \text{noun} & \text{noun} & \text{verb} & \text{verb} & \text{noun} \\
  \text{The} & \text{movie} & \text{I} & \text{watched} & \text{depicted} & \text{hope}
\end{array}
\]
## Feature Engineering for NLP

**Table 3.** Tagging accuracies with different feature templates and other changes on the *WSJ 19-21* development set.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3GRAMMEMM</td>
<td>See text</td>
<td>248,798</td>
<td>52.07%</td>
<td>96.92%</td>
<td>88.99%</td>
</tr>
<tr>
<td>NAACL 2003</td>
<td>See text and [1]</td>
<td>460,552</td>
<td>55.31%</td>
<td>97.15%</td>
<td>88.61%</td>
</tr>
<tr>
<td>Replication</td>
<td>See text and [1]</td>
<td>460,551</td>
<td>55.62%</td>
<td>97.18%</td>
<td>88.92%</td>
</tr>
<tr>
<td>Replication'</td>
<td>+rareFeatureThresh = 5</td>
<td>482,364</td>
<td>55.67%</td>
<td>97.19%</td>
<td>88.96%</td>
</tr>
<tr>
<td>5W</td>
<td>+⟨t₀, w₋₂⟩, ⟨t₀, w₂⟩</td>
<td>730,178</td>
<td>56.23%</td>
<td>97.20%</td>
<td>89.03%</td>
</tr>
<tr>
<td>5WSHAPES</td>
<td>+⟨t₀, s₋₁⟩, ⟨t₀, s₀⟩, ⟨t₀, s₊₁⟩</td>
<td>731,661</td>
<td>56.52%</td>
<td>97.25%</td>
<td>89.81%</td>
</tr>
<tr>
<td>5WSHAPESDS</td>
<td>+ distributional similarity</td>
<td>737,955</td>
<td>56.79%</td>
<td>97.28%</td>
<td>90.46%</td>
</tr>
</tbody>
</table>
Feature Engineering for CV

Edge detection (Canny)

Corner Detection (Harris)

Figures from http://opencv.org
Feature Engineering for CV

Scale Invariant Feature Transform (SIFT)

Figure 1: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated.

Figure from Lowe (1999) and Lowe (2004)
NON-LINEAR FEATURES
Nonlinear Features

- aka. “nonlinear basis functions”
- So far, input was always \( \mathbf{x} = [x_1, \ldots, x_M] \)
- **Key Idea**: let input be some function of \( \mathbf{x} \)
  - original input: \( \mathbf{x} \in \mathbb{R}^M \) where \( M' > M \) (usually)
  - new input: \( \mathbf{x}' \in \mathbb{R}^{M'} \)
  - define \( \mathbf{x}' = b(\mathbf{x}) = [b_1(\mathbf{x}), b_2(\mathbf{x}), \ldots, b_{M'}(\mathbf{x})] \)
    where \( b_i : \mathbb{R}^M \rightarrow \mathbb{R} \) is any function

**Examples**: \( (M = 1) \)

- polynomial
  \[ b_j(x) = x^j \quad \forall j \in \{1, \ldots, J\} \]
- radial basis function
  \[ b_j(x) = \exp \left( \frac{-(x - \mu_j)^2}{2\sigma_j^2} \right) \]
- sigmoid
  \[ b_j(x) = \frac{1}{1 + \exp(-\omega_j x)} \]
- log
  \[ b_j(x) = \log(x) \]

**For a linear model**: still a linear function of \( b(\mathbf{x}) \) even though a nonlinear function of \( \mathbf{x} \)

**Examples**:
- Perceptron
- Linear regression
- Logistic regression
Example: Linear Regression

**Goal:** Learn $y = w^T f(x) + b$
where $f(.)$ is a polynomial basis function

<table>
<thead>
<tr>
<th>i</th>
<th>y</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>1.2</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
<td>1.7</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>1.1</td>
<td>1.9</td>
</tr>
</tbody>
</table>

true “unknown” target function is linear with negative slope and gaussian noise
Example: Linear Regression

**Goal:** Learn $y = \mathbf{w}^\top f(x) + b$
where $f(.)$ is a polynomial basis function

<table>
<thead>
<tr>
<th>i</th>
<th>y</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>1.2</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
<td>1.7</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>1.1</td>
<td>1.9</td>
</tr>
</tbody>
</table>

true “unknown” target function is linear with negative slope and gaussian noise
Example: Linear Regression

**Goal:** Learn \( y = w^T f(x) + b \)
where \( f(.) \) is a polynomial basis function

<table>
<thead>
<tr>
<th>i</th>
<th>y</th>
<th>x</th>
<th>( x^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>1.2</td>
<td>(1.2)^2</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
<td>1.7</td>
<td>(1.7)^2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>1.1</td>
<td>1.9</td>
<td>(1.9)^2</td>
</tr>
</tbody>
</table>

true “unknown” target function is linear with negative slope and gaussian noise
**Example: Linear Regression**

**Goal:** Learn \( y = \mathbf{w}^\top f(\mathbf{x}) + b \)
where \( f(.) \) is a polynomial basis function

<table>
<thead>
<tr>
<th>i</th>
<th>y</th>
<th>x</th>
<th>x^2</th>
<th>x^3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>1.2</td>
<td>(1.2)^2</td>
<td>(1.2)^3</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
<td>1.7</td>
<td>(1.7)^2</td>
<td>(1.7)^3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>1.1</td>
<td>1.9</td>
<td>(1.9)^2</td>
<td>(1.9)^3</td>
</tr>
</tbody>
</table>

true “unknown” target function is linear with negative slope and gaussian noise
**Example: Linear Regression**

**Goal:** Learn $y = w^T f(x) + b$ where $f(.)$ is a polynomial basis function

<table>
<thead>
<tr>
<th>$i$</th>
<th>$y$</th>
<th>$x$</th>
<th>...</th>
<th>$x^5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>1.2</td>
<td>...</td>
<td>(1.2)$^5$</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
<td>1.7</td>
<td>...</td>
<td>(1.7)$^5$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>1.1</td>
<td>1.9</td>
<td>...</td>
<td>(1.9)$^5$</td>
</tr>
</tbody>
</table>

true “unknown” target function is linear with negative slope and gaussian noise
Example: Linear Regression

**Goal:** Learn \( y = w^T f(x) + b \) where \( f(.) \) is a polynomial basis function.

<table>
<thead>
<tr>
<th>i</th>
<th>y</th>
<th>x</th>
<th>...</th>
<th>( x^8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>1.2</td>
<td>...</td>
<td>(1.2)^8</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
<td>1.7</td>
<td>...</td>
<td>(1.7)^8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>1.1</td>
<td>1.9</td>
<td>...</td>
<td>(1.9)^8</td>
</tr>
</tbody>
</table>

true “unknown” target function is linear with negative slope and gaussian noise.
Example: Linear Regression

**Goal:** Learn $y = w^T f(x) + b$ where $f(.)$ is a polynomial basis function.

- True "unknown" target function is linear with negative slope and gaussian noise.

<table>
<thead>
<tr>
<th>i</th>
<th>y</th>
<th>x</th>
<th>...</th>
<th>$x^9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>1.2</td>
<td>...</td>
<td>$(1.2)^9$</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
<td>1.7</td>
<td>...</td>
<td>$(1.7)^9$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>1.1</td>
<td>1.9</td>
<td>...</td>
<td>$(1.9)^9$</td>
</tr>
</tbody>
</table>
Over-fitting

Root-Mean-Square (RMS) Error: $E_{\text{RMS}} = \sqrt{2E(w^*)/N}$

Slide courtesy of William Cohen
Polynomial Coefficients

<table>
<thead>
<tr>
<th></th>
<th>$M = 0$</th>
<th>$M = 1$</th>
<th>$M = 3$</th>
<th>$M = 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_0$</td>
<td>0.19</td>
<td>0.82</td>
<td>0.31</td>
<td>0.35</td>
</tr>
<tr>
<td>$\theta_1$</td>
<td>-1.27</td>
<td>7.99</td>
<td></td>
<td>232.37</td>
</tr>
<tr>
<td>$\theta_2$</td>
<td></td>
<td>-25.43</td>
<td>-5321.83</td>
<td></td>
</tr>
<tr>
<td>$\theta_3$</td>
<td></td>
<td>17.37</td>
<td></td>
<td>48568.31</td>
</tr>
<tr>
<td>$\theta_4$</td>
<td></td>
<td></td>
<td>-231639.30</td>
<td></td>
</tr>
<tr>
<td>$\theta_5$</td>
<td></td>
<td></td>
<td>640042.26</td>
<td></td>
</tr>
<tr>
<td>$\theta_6$</td>
<td></td>
<td></td>
<td>-1061800.52</td>
<td></td>
</tr>
<tr>
<td>$\theta_7$</td>
<td></td>
<td></td>
<td>1042400.18</td>
<td></td>
</tr>
<tr>
<td>$\theta_8$</td>
<td></td>
<td></td>
<td>-557682.99</td>
<td></td>
</tr>
<tr>
<td>$\theta_9$</td>
<td></td>
<td></td>
<td>125201.43</td>
<td></td>
</tr>
</tbody>
</table>
Example: Linear Regression

**Goal:** Learn \( y = w^T f(x) + b \) where \( f(.) \) is a polynomial basis function

<table>
<thead>
<tr>
<th>i</th>
<th>y</th>
<th>x</th>
<th>...</th>
<th>( x^9 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>1.2</td>
<td>...</td>
<td>(1.2)^9</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
<td>1.7</td>
<td>...</td>
<td>(1.7)^9</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>1.1</td>
<td>1.9</td>
<td>...</td>
<td>(1.9)^9</td>
</tr>
</tbody>
</table>

- With just \( N = 10 \) points we overfit!
- But with \( N = 100 \) points, the overfitting (mostly) disappears
- **Takeaway:** more data helps prevent overfitting
Example: Linear Regression

Goal: Learn $y = \mathbf{w}^T f(x) + b$ where $f(.)$ is a polynomial basis function

- With just $N = 10$ points we overfit!
- But with $N = 100$ points, the overfitting (mostly) disappears
- **Takeaway:** more data helps prevent overfitting
REGULARIZATION
Overfitting

**Definition:** The problem of **overfitting** is when the model captures the noise in the training data instead of the underlying structure.

Overfitting can occur in all the models we’ve seen so far:

- Decision Trees (e.g. when tree is too deep)
- KNN (e.g. when k is small)
- Perceptron (e.g. when sample isn’t representative)
- Linear Regression (e.g. with nonlinear features)
- Logistic Regression (e.g. with many rare features)
Motivation: Regularization

• **Occam’s Razor**: prefer the simplest hypothesis

• What does it mean for a hypothesis (or model) to be **simple**?
  1. small number of features (model selection)
  2. small number of “important” features (shrinkage)
Regularization

• **Given** objective function: \( J(\theta) \)
• **Goal** is to find: \( \hat{\theta} = \arg\min_{\theta} J(\theta) + \lambda r(\theta) \)

• **Key idea**: Define regularizer \( r(\theta) \) s.t. we tradeoff between fitting the data and keeping the model simple

• **Choose form of** \( r(\theta) \):
  – Example: q-norm (usually p-norm): \( \|\theta\|_q = \left( \sum_{m=1}^{M} |\theta_m|^q \right)^{\frac{1}{q}} \)

<table>
<thead>
<tr>
<th>( q )</th>
<th>( r(\theta) )</th>
<th>yields parameters that are...</th>
<th>name</th>
<th>optimization notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( |\theta|_0 = \sum 1(\theta_m \neq 0) )</td>
<td>zero values</td>
<td>L0 reg.</td>
<td>no good computational solutions</td>
</tr>
<tr>
<td>1</td>
<td>( |\theta|_1 = \sum</td>
<td>\theta_m</td>
<td>)</td>
<td>zero values</td>
</tr>
<tr>
<td>2</td>
<td>( \left( |\theta|_2 \right)^2 = \sum \theta_m^2 )</td>
<td>small values</td>
<td>L2 reg.</td>
<td>differentiable</td>
</tr>
</tbody>
</table>