Reminders

• Homework 4: Logistic Regression
  – Out: Sun, Sep 30
  – Due: Mon, Oct 8 at 11:59pm

• Reading on Probabilistic Learning is reused later in the course for MLE/MAP

• Schedule changes:
  – lecture on Friday (Oct. 5)
  – no lecture on Wednesday (Oct. 10)
MULTINOMIAL LOGISTIC REGRESSION
Multinomial Logistic Regression

Chalkboard

– Background: Multinomial distribution
– Definition: Multi-class classification
– Geometric intuitions
– Multinomial logistic regression model
– Generative story
– Reduction to binary logistic regression
– Partial derivatives and gradients
– Applying Gradient Descent and SGD
– Implementation w/ sparse features
Debug that Program!

In-Class Exercise: Think-Pair-Share
Debug the following program which is (incorrectly) attempting to run SGD for multinomial logistic regression

Buggy Program:

while not converged:
    for i in shuffle([1,...,N]):
        for k in [1,...,K]:
            theta[k] = theta[k] - lambda * grad(x[i], y[i], theta, k)

Assume: grad(x[i], y[i], theta, k) returns the gradient of the negative log-likelihood of the training example (x[i],y[i]) with respect to vector theta[k]. lambda is the learning rate. N = # of examples. K = # of output classes. M = # of features. theta is a K by M matrix.
Debug that Program!

**In-Class Exercise: Think-Pair-Share**
Debug the following program which is (incorrectly) attempting to run SGD for multinomial logistic regression

**Buggy Program:**

```python
while not converged:
    for i in shuffle([1,...,N]):
        for k in [1,...,K]:
            for m in [1,..., M]:
                theta[k,m] = theta[k,m] + lambda * grad(x[i], y[i], theta, k,m)
```

**Assume:** `grad(x[i], y[i], theta, k, m)` returns the partial derivative of the negative log-likelihood of the training example `(x[i],y[i])` with respect to `theta[k,m]`. `lambda` is the learning rate. `N` = # of examples. `K` = # of output classes. `M` = # of features. `theta` is a `K` by `M` matrix.
Handcrafted Features

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

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

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

**Feature Engineering**

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

**Feature Learning**

- **word embeddings**
  - Mikolov et al., 2013

**Diagram Description**

- **CBOB model in Mikolov et al. (2013)**
  - Similar words, similar embeddings
  - Input (context words)
  - Embedding
  - Missing word
  - Classifier

- **Look-up table**
  - cat: [0.11, 0.23, ..., -0.45]
  - dog: [0.13, 0.26, ..., -0.52]

- **Unsupervised learning**
Where do features come from?

Feature Engineering

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

2. **Word embeddings**
   - Mikolov et al., 2013

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

Feature Learning

Convolutional Neural Networks (*Collobert and Weston 2008*)

Recursive Auto Encoder (*Socher 2011*)

The [movie] showed [wars]

pooling

CNN

RAE
Where do features come from?

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

- **Word embeddings**
  - Mikolov et al., 2013
  - Socher, 2011
  - Collobert & Weston, 2008

- **String embeddings**
  - Socher, 2011

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

The [movie] showed [wars]
Where do features come from?

- **Feature Engineering**
  - hand-crafted features
    - Sun et al., 2011
    - Zhou et al., 2005
  - word embedding features
    - Turian et al., 2010
    - Koo et al., 2008
    - Mikolov et al., 2013

- **Feature Learning**
  - string embeddings
    - Socher, 2011
    - Collobert & Weston, 2008
  - tree embeddings
    - Socher et al., 2013
    - Hermann & Blunsom, 2013
  - word embeddings
    - Hermann et al., 2014

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?**
  - tree embeddings
    - Socher et al., 2013
    - Hermann & Blunsom, 2013
  - string embeddings
    - Socher, 2011
    - Mikolov et al., 2013
  - 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
## Feature Engineering for NLP

### 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, &quot;e&quot;)</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, &quot;d&quot;)</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, &quot;ed&quot;)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(w_i = ) &quot;aardvark&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(w_i = ) &quot;hope&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</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>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
</tr>
<tr>
<td>(w_i = \text{&quot;watched&quot;})</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} = \text{&quot;watched&quot;})</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} = \text{&quot;watched&quot;})</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} = \text{&quot;watched&quot;})</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} = \text{&quot;watched&quot;})</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
</tr>
</tbody>
</table>

**The movie I watched depicted hope**
Feature Engineering for NLP

Context Features:

<table>
<thead>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>wi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wi+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wi-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wi+2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wi-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The movie I watched depicted hope

deter. noun noun verb verb noun
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 3: Model images of planar objects are shown in the top row. Recognition results below show model outlines and image keys used for matching.

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 $b_i : \mathbb{R}^M \to \mathbb{R}$ 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

true “unknown” target function is \( y = \tanh(x) + \text{noise} \)
Example: Linear Regression

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

true “unknown” target function is $y = \tanh(x) + \text{noise}$
Example: Linear Regression

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

true “unknown” target function is $y = \tanh(x) + \text{noise}$
Example: Linear Regression

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

true “unknown” target function is \( y = \tanh(x) + \text{noise} \)
**Example: Linear Regression**

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

true “unknown” target function is \( y = \tanh(x) + \text{noise} \)
Example: Linear Regression

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

true “unknown” target function is \( y = \tanh(x) + \text{noise} \)
Example: Linear Regression

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

true “unknown” target function is $y = \tanh(x) + \text{noise}$
**Example: Linear Regression**

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

true “unknown” target function is $y = \tanh(x) + \text{noise}$
Example: Linear Regression

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

true “unknown” target function is $y = \tanh(x) + \text{noise}$
Example: Linear Regression

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

true “unknown” target function is $y = \tanh(x) + \text{noise}$
Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + b$ where $f(.)$ is a polynomial basis function.

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
Example: Linear Regression

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

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.
**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
**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.
Example: Linear Regression

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

ture “unknown” target function is linear with negative slope and gaussian noise
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></td>
<td>-1.27</td>
<td>7.99</td>
<td>232.37</td>
</tr>
<tr>
<td>$\theta_2$</td>
<td></td>
<td></td>
<td>-25.43</td>
<td>-5321.83</td>
</tr>
<tr>
<td>$\theta_3$</td>
<td></td>
<td></td>
<td>17.37</td>
<td>48568.31</td>
</tr>
<tr>
<td>$\theta_4$</td>
<td></td>
<td></td>
<td></td>
<td>-231639.30</td>
</tr>
<tr>
<td>$\theta_5$</td>
<td></td>
<td></td>
<td></td>
<td>640042.26</td>
</tr>
<tr>
<td>$\theta_6$</td>
<td></td>
<td></td>
<td></td>
<td>-1061800.52</td>
</tr>
<tr>
<td>$\theta_7$</td>
<td></td>
<td></td>
<td></td>
<td>1042400.18</td>
</tr>
<tr>
<td>$\theta_8$</td>
<td></td>
<td></td>
<td></td>
<td>-557682.99</td>
</tr>
<tr>
<td>$\theta_9$</td>
<td></td>
<td></td>
<td></td>
<td>125201.43</td>
</tr>
</tbody>
</table>
**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.
Example: Linear Regression

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

Same as before, but now with \( N = 100 \) points.

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

– KNN (e.g. when k is small)
– Naïve Bayes (e.g. without a prior)
– Linear Regression (e.g. with basis function)
– Logistic Regression (e.g. with many rare features)
Motivation: Regularization

Example: Stock Prices

• Suppose we wish to predict Google’s stock price at time t+1

• What features should we use? (putting all computational concerns aside)
  – Stock prices of all other stocks at times t, t-1, t-2, …, t-k
  – Mentions of Google with positive / negative sentiment words in all newspapers and social media outlets

• Do we believe that all of these features are going to be useful?
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

Chalkboard

– L2, L1, L0 Regularization
– Example: Linear Regression
Regularization

Don’t Regularize the Bias (Intercept) Parameter!
• In our models so far, the bias / intercept parameter is usually denoted by $\theta_0$ -- that is, the parameter for which we fixed $x_0 = 1$
• Regularizers always avoid penalizing this bias / intercept parameter
• Why? Because otherwise the learning algorithms wouldn’t be invariant to a shift in the y-values

Whitening Data
• It’s common to *whiten* each feature by subtracting its mean and dividing by its variance
• For regularization, this helps all the features be penalized in the same units (e.g. convert both centimeters and kilometers to z-scores)
Regularization:

\[ \ln \lambda = +18 \]
## Polynomial Coefficients

<table>
<thead>
<tr>
<th>$w^*_0$</th>
<th>none</th>
<th>exp(18)</th>
<th>huge</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.35</td>
<td>0.35</td>
<td>0.13</td>
<td></td>
</tr>
<tr>
<td>$w^*_1$</td>
<td>232.37</td>
<td>4.74</td>
<td>-0.05</td>
</tr>
<tr>
<td>$w^*_2$</td>
<td>-5321.83</td>
<td>-0.77</td>
<td>-0.06</td>
</tr>
<tr>
<td>$w^*_3$</td>
<td>48568.31</td>
<td>-31.97</td>
<td>-0.05</td>
</tr>
<tr>
<td>$w^*_4$</td>
<td>-231639.30</td>
<td>-3.89</td>
<td>-0.03</td>
</tr>
<tr>
<td>$w^*_5$</td>
<td>640042.26</td>
<td>55.28</td>
<td>-0.02</td>
</tr>
<tr>
<td>$w^*_6$</td>
<td>-1061800.52</td>
<td>41.32</td>
<td>-0.01</td>
</tr>
<tr>
<td>$w^*_7$</td>
<td>1042400.18</td>
<td>-45.95</td>
<td>-0.00</td>
</tr>
<tr>
<td>$w^*_8$</td>
<td>-557682.99</td>
<td>-91.53</td>
<td>0.00</td>
</tr>
<tr>
<td>$w^*_9$</td>
<td>125201.43</td>
<td>72.68</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Over Regularization:
Regularization Exercise

In-class Exercise
1. Plot train error vs. # features (cartoon)
2. Plot test error vs. # features (cartoon)
Example: Logistic Regression
Example: Logistic Regression

Test Data
Example: Logistic Regression

![Graph showing error vs. 1/λ for train and test data.](image)
Example: Logistic Regression

Classification with Logistic Regression (\lambda=1e-05)
Example: Logistic Regression

Classification with Logistic Regression (\(\lambda=0.0001\))
Example: Logistic Regression

Classification with Logistic Regression (|lambda|=0.001)
Example: Logistic Regression
Example: Logistic Regression

Classification with Logistic Regression (\(\text{lambda}=0.1\))
Example: Logistic Regression

Classification with Logistic Regression (λ = 1)
Example: Logistic Regression

Classification with Logistic Regression (\(\lambda = 10\))
Example: Logistic Regression

Classification with Logistic Regression (\(\text{lambda}=100\))
Example: Logistic Regression

Classification with Logistic Regression (\(\lambda = 1000\))
Example: Logistic Regression

Classification with Logistic Regression (\(\text{lambda}=10000\))
Example: Logistic Regression
Example: Logistic Regression

Classification with Logistic Regression (\(\lambda = 1e+06\))
Example: Logistic Regression

Classification with Logistic Regression (\(\lambda = 1e+07\))
Example: Logistic Regression
Regularization as MAP

• L1 and L2 regularization can be interpreted as \textit{maximum a-posteriori (MAP) estimation} of the parameters

• To be discussed later in the course...
Takeaways

1. **Nonlinear basis functions** allow **linear models** (e.g. Linear Regression, Logistic Regression) to capture **nonlinear** aspects of the original input

2. Nonlinear features are **require no changes to the model** (i.e. just preprocessing)

3. **Regularization** helps to avoid **overfitting**

4. **Regularization** and **MAP estimation** are equivalent for appropriately chosen priors
Feature Engineering / Regularization

Objectives

You should be able to...

• Engineer appropriate features for a new task
• Use feature selection techniques to identify and remove irrelevant features
• Identify when a model is overfitting
• Add a regularizer to an existing objective in order to combat overfitting
• Explain why we should not regularize the bias term
• Convert linearly inseparable dataset to a linearly separable dataset in higher dimensions
• Describe feature engineering in common application areas