Outline of CSE 446
We’ve already covered stuff in blue!

- Problem formulations: classification, regression
- Supervised techniques: decision trees, nearest neighbors, perceptron, linear models, probabilistic models, neural networks, kernel methods
- Unsupervised techniques: clustering, linear dimensionality reduction
- “Meta-techniques”: ensembles, expectation-maximization
- Understanding ML: limits of learning, practical issues, bias & fairness
- Recurring themes: (stochastic) gradient descent, bullshit detection
Today: (More) Best Practices

You already know:

- Separating training and test data
- Hyperparameter tuning on development data

Understanding machine learning is partly about knowing algorithms and partly about the art of mapping abstract problems to learning tasks.
Features Matter
Features Matter
Features Matter
Features Matter
Irrelevant Features
Irrelevant Features

One irrelevant feature isn’t a big deal; what we’re worried about is when irrelevant features *outnumber* useful ones!
Irrelevant Features

One irrelevant feature isn’t a big deal; what we’re worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
Irrelevant Features

One irrelevant feature isn’t a big deal; what we’re worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!
Irrelevant Features

One irrelevant feature isn’t a big deal; what we’re worried about is when irrelevant features outnumber useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!
- \(K\)-nearest neighbors?
Irrelevant Features

One irrelevant feature isn’t a big deal; what we’re worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!

- $K$-nearest neighbors? 😞
Irrelevant Features

One irrelevant feature isn’t a big deal; what we’re worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!

- $K$-nearest neighbors? 😞

- Perceptron?
Irrelevant Features

One irrelevant feature isn’t a big deal; what we’re worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!

- $K$-nearest neighbors? 😞

- Perceptron? 😊
Irrelevant Features

One irrelevant feature isn’t a big deal; what we’re worried about is when irrelevant features *outnumber* useful ones!

- Decision trees (not too deep)?
  Somewhat protected, but beware spurious correlations!

- $K$-nearest neighbors? 😞
- Perceptron? 😊

What about *redundant* features $\phi_j$ and $\phi_{j'}$ such that $\phi_j \approx \phi_{j'}$?
Technique: Feature Pruning

If a binary feature is present in too small or too large a fraction of $D$, remove it.
Technique: Feature Pruning

If a binary feature is present in too small or too large a fraction of $D$, remove it.

Example: $\phi(x) = \llbracket \text{the word } the \text{ occurs in document } x \rrbracket$
Technique: Feature Pruning

If a binary feature is present in too small or too large a fraction of $D$, remove it.

Example: $\phi(x) = \left[ \text{the word } \textit{the} \text{ occurs in document } x \right]$

Generalization: if a feature has variance (in $D$) lower than some threshold value, remove it.

Note: in lecture, I mistakenly said to remove high-variance features. Mea culpa.

$$\text{sample mean}(\phi; D) = \frac{1}{N} \sum_{n=1}^{N} \phi(x_n) \quad \text{(call it “}\overline{\phi}\text{”)}$$

$$\text{sample variance}(\phi; D) = \frac{1}{N-1} \sum_{n=1}^{N} (\phi(x_n) - \overline{\phi})^2 \quad \text{(call it “}\text{Var}(\phi)\text{”)}$$
Technique: Feature Normalization

Center a feature:

\[ \phi(x) \rightarrow \phi(x) - \bar{\phi} \]

(This was a required step for principal components analysis!)

Scale a feature. Two choices:

\[ \phi(x) \rightarrow \frac{\phi(x)}{\sqrt{\text{Var}(\phi)}} \quad \text{“variance scaling”} \]
\[ \phi(x) \rightarrow \frac{\phi(x)}{\max_n |\phi(x_n)|} \quad \text{“absolute scaling”} \]
Technique: Feature Normalization

Center a feature:

\[ \phi(x) \rightarrow \phi(x) - \bar{\phi} \]

(This was a required step for principal components analysis!)

Scale a feature. Two choices:

\[ \phi(x) \rightarrow \frac{\phi(x)}{\sqrt{\text{Var}(\phi)}} \]  

“variance scaling”

\[ \phi(x) \rightarrow \frac{\phi(x)}{\max_n \left| \phi(x_n) \right|} \]  

“absolute scaling”

Remember that you’ll need to normalize test data before you test!
Techniques: Creating New Features

1. Consider two binary features, $\phi_j$ and $\phi_{j'}$. A new conjunction feature can be defined by:

$$\phi_{j\land j'}(x) = \phi_j(x) \land \phi_{j'}(x)$$
Techniques: Creating New Features

1. Consider two binary features, $\phi_j$ and $\phi_{j'}$. A new conjunction feature can be defined by:

$$\phi_{j \land j'}(x) = \phi_j(x) \land \phi_{j'}(x)$$

The classic “xor” problem: these points are not linearly separable.
Techniques: Creating New Features

1. Consider two binary features, $\phi_j$ and $\phi_{j'}$. A new conjunction feature can be defined by:

$$\phi_{j \land j'}(x) = \phi_j(x) \land \phi_{j'}(x)$$


![Diagram showing the conjunction of two features]
Techniques: Creating New Features

1. Consider two binary features, $\phi_j$ and $\phi_{j'}$. A new conjunction feature can be defined by:

$$\phi_{j \land j'}(x) = \phi_j(x) \land \phi_{j'}(x)$$

Rotating the view.

Example: $\phi_j(x)$ is the count of the word cool in document $x$. 
Techniques: Creating New Features

1. Consider two binary features, $\phi_j$ and $\phi_{j'}$. A new conjunction feature can be defined by:

$$\phi_{j \land j'}(x) = \phi_j(x) \land \phi_{j'}(x)$$
Techniques: Creating New Features

1. Consider two binary features, $\phi_j$ and $\phi_{j'}$. A new conjunction feature can be defined by:

$$\phi_{j \land j'}(x) = \phi_j(x) \land \phi_{j'}(x)$$

2. Even more generally, we can create conjunctions (or products) using as many features as we’d like.

3. Transformations on features can be useful. For example, $\phi_j(x) \rightarrow \text{sign}(\phi_j(x)) \cdot \log(1 + |\phi_j(x)|)$

Example: $\phi_j(x)$ is the count of the word cool in document $x$. 

$$2 \cdot x[1] + 2 \cdot x[2] - 4 \cdot x[3] - 1 = 0$$
Techniques: Creating New Features

1. Consider two binary features, $\phi_j$ and $\phi_{j'}$. A new conjunction feature can be defined by:

$$
\phi_{j \land j'}(x) = \phi_j(x) \land \phi_{j'}(x)
$$

Generalization: take the product of two features.
Techniques: Creating New Features

1. Consider two binary features, $\phi_j$ and $\phi_{j'}$. A new conjunction feature can be defined by:

$$\phi_{j \land j'}(x) = \phi_j(x) \land \phi_{j'}(x)$$

Generalization: take the product of two features.

2. Even more generally, we can create conjunctions (or products) using as many features as we’d like.
Techniques: Creating New Features

1. Consider two binary features, $\phi_j$ and $\phi_{j'}$. A new conjunction feature can be defined by:

$$\phi_{j \land j'}(x) = \phi_j(x) \land \phi_{j'}(x)$$

Generalization: take the product of two features.

2. Even more generally, we can create conjunctions (or products) using as many features as we’d like. 
   This is one view of what decision trees are doing!
   ▶ Every leaf’s path (from root) is a conjunction feature.
   ▶ Why not build decision trees, extract the features and toss them into the perceptron?

Example: $\phi_j(x)$ is the count of the word cool in document $x$. 
Techniques: Creating New Features

1. Consider two binary features, $\phi_j$ and $\phi_{j'}$. A new conjunction feature can be defined by:

$$\phi_{j \land j'}(x) = \phi_j(x) \land \phi_{j'}(x)$$

Generalization: take the product of two features.

2. Even more generally, we can create conjunctions (or products) using as many features as we'd like. This is one view of what decision trees are doing!
   - Every leaf’s path (from root) is a conjunction feature.
   - Why not build decision trees, extract the features and toss them into the perceptron?

3. Transformations on features can be useful. For example,

$$\phi(x) \rightarrow \text{sign}(\phi(x)) \cdot \log (1 + |\phi(x)|)$$

Example: $\phi(x)$ is the count of the word cool in document $x$. 
Evaluation

Accuracy:

\[ A(f) = \sum_x D(x, f(x)) \]

\[ = \sum_{x,y} D(x, y) \cdot \begin{cases} 1 & \text{if } f(x) = y \\ 0 & \text{otherwise} \end{cases} \]

\[ = \sum_{x,y} D(x, y) \cdot [f(x) = y] \]

where \( D \) is the true distribution over data. Error is \( 1 - A \); earlier we denoted error \( \epsilon(f) \).

This is estimated using a test dataset \( \langle x_1, y_2 \rangle, \ldots, \langle x_{N'}, y_{N'} \rangle \):

\[ \hat{A}(f) = \frac{1}{N'} \sum_{i=1}^{N'} [f(x_i) = y_i] \]
Issues with Test-Set Accuracy

Class imbalance: if $D(\ast, \text{not spam}) = 0.99$, then you can get $\hat{A} \approx 0.99$ by always guessing "not spam."

Relative importance of classes or cost of error types.

Variance due to the test data.
Issues with Test-Set Accuracy

▶ Class imbalance: if $D(*, \text{not spam}) = 0.99$, then you can get $\hat{A} \approx 0.99$ by always guessing “not spam.”
Issues with Test-Set Accuracy

- Class imbalance: if \( D(\ast, \text{not spam}) = 0.99 \), then you can get \( \hat{A} \approx 0.99 \) by always guessing “not spam.”
- Relative importance of classes or cost of error types.
Issues with Test-Set Accuracy

▶ Class imbalance: if $D(\ast, \text{not spam}) = 0.99$, then you can get $\hat{A} \approx 0.99$ by always guessing “not spam.”
▶ Relative importance of classes or cost of error types.
▶ Variance due to the test data.
Evaluation in the Two-Class Case

Suppose we have two classes, and one of them, $t$, is a “target.”

- E.g., given a query, find relevant documents.

**Precision** and **recall** encode the goals of returning a “pure” set of targeted instances and capturing *all* of them.

$$\hat{P}(f) = \frac{|C|}{|B|} = \frac{|A \cap B|}{|B|}$$

$$\hat{R}(f) = \frac{|C|}{|A|} = \frac{|A \cap B|}{|A|}$$

$$\hat{F}_1(f) = 2 \cdot \frac{\hat{P} \cdot \hat{R}}{\hat{P} + \hat{R}}$$
Another View: Contingency Table

<table>
<thead>
<tr>
<th></th>
<th>$y = t$</th>
<th>$y \neq t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x) = t$</td>
<td>$C$ (true positives)</td>
<td>$B \setminus C$ (false positives)</td>
</tr>
<tr>
<td>$f(x) \neq t$</td>
<td>$A \setminus C$ (false negatives)</td>
<td>(true negatives)</td>
</tr>
</tbody>
</table>