CS 613
Lecture 4
Aylin Caliskan-Islam
January 28, 2015
Reminders

• HW2 due next Wednesday before class
• Dr. Greenstadt will be watching the presentations
Outline

• Support Vector Machines
  – Optimization Objective
  – Large Margin Intuition
  – Kernels

• Random Forest
  – Ensemble Methods
  – Algorithm
  – Node split
  – OOB error
Outline

• Support Vector Machines
  – from Dr. Andrew Ng’s notes

• Random Forest
  – from Dr. Leo Breiman
  – from Dr. Markus Kalisch
Alternative view of logistic regression

\[ h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \]
Alternative view of logistic regression

\[ h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \]

If \( y = 1 \), we want \( h_\theta(x) \approx 1 \), \( \theta^T x \gg 0 \)
If \( y = 0 \), we want \( h_\theta(x) \approx 0 \), \( \theta^T x \ll 0 \)
Alternative view of logistic regression

Cost of example: \(- (y \log h_\theta(x) + (1 - y) \log (1 - h_\theta(x)))\)

\[= -y \log \frac{1}{1 + e^{-\theta^T x}} - (1 - y) \log \left(1 - \frac{1}{1 + e^{-\theta^T x}}\right)\]

If \(y = 1\) (want \(\theta^T x \gg 0\)):

\[cost_1(z) = -\log \frac{1}{1 + e^{-z}}\]

\[z = \Theta^T x\]

If \(y = 0\) (want \(\theta^T x \ll 0\)):

\[cost_0(z) = -\log(1 - \frac{1}{1 + e^{-z}})\]

\(-1\)
Support vector machine

Logistic regression:

\[
\min_{\theta} \frac{1}{m} \left[ \sum_{i=1}^{m} y^{(i)} \left( -\log h_\theta(x^{(i)}) \right) + (1 - y^{(i)}) \left( -\log(1 - h_\theta(x^{(i)})) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2
\]

Support vector machine:
Support vector machine
Logistic regression:

$$\min_{\theta} \frac{1}{\lambda} \left[ \sum_{i=1}^{m} y^{(i)} \left( -\log h_\theta(x^{(i)}) \right) + (1 - y^{(i)}) \left( -\log(1 - h_\theta(x^{(i)})) \right) \right] + \frac{\lambda}{2} \sum_{j=1}^{n} \theta_j^2$$

Support vector machine:

- $$\min_u (u - 5)^2 + 1 \Rightarrow u = 5$$
- $$\min_u 10((u - 5)^2 + 1) \Rightarrow u = 5$$
- $$\min_u 10(u - 5)^2 + 10 \Rightarrow u = 5$$

$$\min_{\theta} C \sum_{i=1}^{m} \left[ y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$
Support vector machine

Logistic regression:

\[ \min_\theta \frac{1}{m} \left[ \sum_{i=1}^{m} y^{(i)} \left( -\log h_\theta(x^{(i)}) \right) + (1 - y^{(i)}) \left( -\log(1 - h_\theta(x^{(i)})) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 \]

Support vector machine:

\[ \min_u (u - 5)^2 + 1 \Rightarrow u = 5 \]
\[ \min_u 10((u - 5)^2 + 1) \Rightarrow u = 5 \]
\[ \min_u 10(u - 5)^2 + 10 \Rightarrow u = 5 \]

Logistic regression: \( A + \lambda B \)
Support vector machine: \( CA + B \)
\( C \approx (1/\lambda) \)
SVM hypothesis

\[ \min_{\theta} C \sum_{i=1}^{m} \left[ y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2 \]

Hypothesis:

Unlike logistic regression, SVM does not output probability

\[ h_\theta(x) = \begin{cases} 
1 & \text{if } \theta^T x \geq 0 \\
0 & \text{otherwise}
\end{cases} \]
Outline

• Support Vector Machines
  ✔ – Optimization Objective
    – Large Margin Intuition
    – Kernels

• Random Forest
  – Ensemble Methods
  – Algorithm
  – Node split
  – OOB error
Large Margin Intuition

SVM hypothesis

$$\min_{\theta} \sum_{i=1}^{m} \left[ y^{(i)} \cos 1(\theta^T x^{(i)}) + (1 - y^{(i)}) \cos 0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

If \( y = 1 \) (want \( \theta^T x \gg 0 \)):  
If \( y = 0 \) (want \( \theta^T x \ll 0 \)):  

\[ z = \Theta^T x \]

\[ \cos 1(z) \]

\[ -1 \]
Large Margin Intuition

SVM hypothesis

$$\min_{\theta} C \sum_{i=1}^{m} \left[ y^{(i)} \text{cost}_1 (\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0 (\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

If $y = 1$ (want $\theta^T x \geq 1)$:

If $y = 0$ (want $\theta^T x \leq -1$):

$C = 100,000$

Wednesday - January 28, 2015
Large Margin Intuition

SVM Decision Boundary: \textit{Linearly separable} case

![Large margin classifier](image)
Large Margin Intuition

Large margin classifier in presence of outliers

$\rightarrow C$ very large

$\rightarrow C$ not too large

$(1/\lambda)$
Math Behind the Large Margin

**Vector Inner Product**

\[ u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \]
Math Behind the Large Margin

Vector Inner Product

\[ u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \]

\[ u^T v = ? \quad [u_1 \ u_2] [v_1 \ v_2]^T \]

\[ \|u\| = \text{length of vector } u \]
\[ = \sqrt{u_1^2 + u_2^2} \quad \in \mathbb{R} \]

\[ p = \text{length of projection of } v \text{ onto } u. \]

\[ u^T v = p \cdot \|u\| \leftarrow = v^T u \]

Signed
\[ = u_1 v_1 + u_2 v_2 \leftarrow \quad p \in \mathbb{R} \]

\[ u^T v = p \cdot \|u\| \]

\[ p < 0 \]
Math Behind the Large Margin

Vector Inner Product

\[ \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \]

\[ \mathbf{u}^T \mathbf{v} = \begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \]

\[ ||\mathbf{u}|| = \text{length of vector } \mathbf{u} \]
\[ = \sqrt{u_1^2 + u_2^2} \in \mathbb{R} \]

\[ p = \text{length of projection of } \mathbf{v} \text{ onto } \mathbf{u}. \]
\[ \mathbf{u}^T \mathbf{v} = p \cdot ||\mathbf{u}|| \]

Signed
\[ = u_1 v_1 + u_2 v_2 \leq p \in \mathbb{R} \]

\[ \mathbf{u}^T \mathbf{v} = p \cdot ||\mathbf{u}|| \]
\[ p < 0 \]
Math Behind the Large Margin

SVM Decision Boundary

\[
\min_{\theta} \frac{1}{2} \sum_{i=1}^{n} \theta_i^2 = \frac{1}{2} (\theta_1^2 + \theta_2^2) = \frac{1}{2} \left( \sqrt{\theta_1^2 + \theta_2^2} \right)^2 = \frac{1}{2} \|\theta\|^2
\]

s.t. \( \theta^T x^{(i)} \geq 1 \) if \( y^{(i)} = 1 \)

\[
\theta^T x^{(i)} \leq -1 \quad \text{if} \quad y^{(i)} = 0
\]

Simplification: \( \theta_0 = 0 \). \( n=2 \)

\[
\theta^T x^{(i)} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \cdot \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \end{bmatrix} = \theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)}
\]
Math Behind the Large Margin

**SVM Decision Boundary**

\[
\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2} \|\theta\|^2 \\
\text{s.t. } p^{(i)} \cdot \|\theta\| \geq 1 \quad \text{if } y^{(i)} = 1 \\
\text{or } p^{(i)} \cdot \|\theta\| \leq -1 \quad \text{if } y^{(i)} = 0
\]

where \( p^{(i)} \) is the projection of \( x^{(i)} \) onto the vector \( \theta \).

Simplification: \( \theta_0 = 0 \)
Math Behind the Large Margin

SVM Decision Boundary

\[
\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2} \|\theta\|^2
\]

s.t. \( p^{(i)} \cdot \|\theta\| \geq 1 \) if \( y^{(i)} = 1 \)

\( p^{(i)} \cdot \|\theta\| \leq -1 \) if \( y^{(i)} = 0 \)

where \( p^{(i)} \) is the projection of \( x^{(i)} \) onto the vector \( \theta \).

Simplification: \( \theta_0 = 0 \)
Outline

• Support Vector Machines
  ✔— Optimization Objective
  ✔— Large Margin Intuition
    – Kernels
• Random Forest
  – Ensemble Methods
  – Algorithm
  – Node split
  – OOB error
Kernels

Predict $y = 1$ if

$$
\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 \\
+ \theta_4 x_1^2 + \theta_5 x_2^2 + \cdots \geq 0
$$

$$
h_0(x) = \begin{cases} 
1 & \text{if } \theta_0 + \theta_1 x_1 + \cdots \geq 0 \\
0 & \text{otherwise}
\end{cases}
$$
Kernels

Non-linear Decision Boundary

\[ \text{Predict } y = 1 \text{ if } \]
\[ \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \ldots \geq 0 \]

\[ h_0(x) = \begin{cases} 
1 & \text{if } \theta_0 + \theta_1 x_1 + \ldots \geq 0 \\
0 & \text{otherwise}
\end{cases} \]

\[ \Rightarrow \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \ldots \]
\[ f_1 = x_1, \quad f_2 = x_2, \quad f_3 = x_1 x_2, \quad f_4 = x_1^2, \quad f_5 = x_2^2, \ldots \]

Is there a different / better choice of the features \( f_1, f_2, f_3, \ldots \)?
Kernels

Given $x$, compute new feature depending on proximity to landmarks $l^{(1)}, l^{(2)}, l^{(3)}$.
Kernels

Given $x$, compute new feature depending on proximity to landmarks $l^{(1)}, l^{(2)}, l^{(3)}$

Given $x$:

$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$

$f_2 = \text{similarity}(x, l^{(2)}) = \exp\left(-\frac{\|x - l^{(2)}\|^2}{2\sigma^2}\right)$

$f_3 = \text{similarity}(x, l^{(3)}) = \exp\left(\ldots\right)$
Given $x$, compute new feature depending on proximity to landmarks $l^{(1)}$, $l^{(2)}$, $l^{(3)}$

Given $x$:

1. $f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{||x - l^{(1)}||^2}{2 \sigma^2}\right)$
2. $f_2 = \text{similarity}_2(x, l^{(2)}) = \exp\left(-\frac{||x - l^{(2)}||^2}{2 \sigma^2}\right)$
3. $f_3 = \text{similarity}_3(x, l^{(3)}) = \exp\left(\ldots\right)$

Kernel (Gaussian kernels) $k(x, l^{(i)})$
Kernels

Kernels and Similarity

\[ f_1 = \text{similarity}(x, l^{(1)}) = \exp \left( -\frac{\|x - l^{(1)}\|^2}{2\sigma^2} \right) \]

If \( x \approx l^{(1)} \):

\[ f_1 \approx \exp \left( -\frac{b^2}{2\sigma^2} \right) \approx 1 \]

If \( x \) far from \( l^{(1)} \):

\[ f_1 = \exp \left( -\frac{(\text{large number})^2}{2\sigma^2} \right) \approx 0 \]
Kernels

Example:

\[ l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix} \quad f_1 = \exp \left( -\frac{\| x - l^{(1)} \|^2}{2\sigma^2} \right) \]

\[ \sigma^2 = 1 \]

\[ x = \begin{bmatrix} 3 \\ 5 \end{bmatrix} \quad \sigma^2 = 0.5 \]

\[ \sigma^2 = 3 \]
Kernels

Predict “1” when

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$$

$$\theta_0 = -0.5, \ \theta_1 = 1, \ \theta_2 = 1, \ \theta_3 = 0$$

$$f_1 \neq 0, \ f_2 \neq 0, \ f_3 \neq 0$$

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 = -0.5 + 1 = 0.5 > 0$$

$$f_1, f_2, f_3 \neq 0$$

$$\theta_0 + \theta_1 f_1 + \ldots \approx -0.5 < 0$$
Kernels

Choosing the landmarks

Given \( x \):

\[ f_i = \text{similarity}(x, l^{(i)}) \]
\[ = \exp \left( -\frac{||x - l^{(i)}||^2}{2\sigma^2} \right) \]

Predict \( y = 1 \) if \( \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0 \)

Where to get \( l^{(1)}, l^{(2)}, l^{(3)}, \ldots \)?
Kernels

**SVM with Kernels**

- Given \( (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)}) \),
- choose \( l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \ldots, l^{(m)} = x^{(m)} \).

Given example \( x \):

- \( f_1 = \text{similarity}(x, l^{(1)}) \)
- \( f_2 = \text{similarity}(x, l^{(2)}) \)
- \( \ldots \)

For training example \( (x^{(i)}, y^{(i)}) \):

- \( f_1^{(i)} = \text{similarity}(x^{(i)}, l^{(1)}) \)
- \( f_2^{(i)} = \text{similarity}(x^{(i)}, l^{(2)}) \)
- \( \ldots \)
- \( f_m^{(i)} = \text{similarity}(x^{(i)}, l^{(m)}) \)

\( f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \)

\( f_0 = 1 \)
Kernels

**SVM with Kernels**

Hypothesis: Given $x$, compute features $f \in \mathbb{R}^{m+1}$

$\rightarrow$ Predict “y=1” if $\theta^T f \geq 0$

Training:

$$\min_{\theta} C \sum_{i=1}^{m} y^{(i)} \text{cost}_1(\theta^T f^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$$

$$\begin{bmatrix}
- \sum_{j=1}^{m} \theta_j^2 \\
\end{bmatrix} = \theta^T \theta \leftarrow \theta = \begin{bmatrix} \theta_1 \\
\vdots \\
\theta_m \\
\end{bmatrix}$$

(ignoring $\theta_0$)

$m = 10,000$
Kernels

**SVM parameters:**

C \left( = \frac{1}{\lambda} \right). \Rightarrow \text{Large C: Lower bias, high variance.}

\Rightarrow \text{Small C: Higher bias, low variance.}

\( \sigma^2 \)

\text{Large } \sigma^2: \text{ Features } f_i \text{ vary more smoothly.}

\Rightarrow \text{Higher bias, lower variance.}

\exp \left( - \frac{\|x - x_i\|^2}{2\sigma^2} \right)

\text{Small } \sigma^2: \text{ Features } f_i \text{ vary less smoothly.}

\text{Lower bias, higher variance.}
Using an SVM

Use SVM software package (e.g. liblinear, libsvm, ...) to solve for parameters $\theta$.

Need to specify:

→ Choice of parameter C.

Choice of kernel (similarity function):

E.g. No kernel ("linear kernel")

Predict "y = 1" if $\theta^T x \geq 0$

→ Gaussian kernel:

$$f_i = \exp \left( - \frac{||x - l(i)||^2}{2\sigma^2} \right)$$

where $l(i) = x(i)$.

Need to choose $\sigma^2$. 

Wednesday - January 28, 2015
Using an SVM

Kernel (similarity) functions:

\[
\text{function } f = \text{kernel}(x_1, x_2) \\
\quad f = \exp \left( -\frac{\|x_1 - x_2\|^2}{2\sigma^2} \right) \\
\text{return}
\]

→ Note: Do perform feature scaling before using the Gaussian kernel.

\[
\|x - \lambda\|^2 = v_1^2 + v_2^2 + \ldots + v_n^2 \\
= (x_1 - \lambda_1)^2 + (x_2 - \lambda_2)^2 + \ldots + (x_n - \lambda_n)^2 \\
\leq 1000 \text{ feet}^2 \quad 1-5 \text{ bedrooms}
\]
Using an SVM

Other choices of kernel

Note: Not all similarity functions \( \text{similarity}(x, l) \) make valid kernels.

\( \Rightarrow \) (Need to satisfy technical condition called “Mercer’s Theorem” to make sure SVM packages’ optimizations run correctly, and do not diverge).

Many off-the-shelf kernels available:

- Polynomial kernel: \( k(x, l) = (x^T l + \text{constant})^\text{degree} \)

- More esoteric: String kernel, chi-square kernel, histogram intersection kernel, ...
Multi-class classification

Many SVM packages already have built-in multi-class classification functionality.

Otherwise, use one-vs.-all method. (Train $K$ SVMs, one to distinguish $y=i$ from the rest, for $i = 1, 2, \ldots, K$), get $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(K)}$

Pick class $i$ with largest $(\theta^{(i)})^T x$
Logistic regression vs. SVMs

\[ n = \text{number of features (} x \in \mathbb{R}^{n+1}, \ m = \text{number of training examples} \]

- If \( n \) is large (relative to \( m \)): \( (n \geq m, \ n = 10,000, \ m = 10 \ldots 1000) \)
  - Use logistic regression, or SVM without a kernel (“linear kernel”)

- If \( n \) is small, \( m \) is intermediate:
  - Use SVM with Gaussian kernel
  \( (n = 1-1000, \ m = 10-10,000) \)

- If \( n \) is small, \( m \) is large:
  - Create/add more features, then use logistic regression or SVM without a kernel

- Neural network likely to work well for most of these settings, but may be slower to train.
Outline

• Support Vector Machines
  ✔ – Optimization Objective
  ✔ – Large Margin Intuition
  ✔ – Kernels

• Random Forest
  – Ensemble Methods
  – Algorithm
  – Node split
  – OOB error
RANDOM FOREST
RANDOM FOREST

• **Advantages**: Accurate, easy to use, fast, robust
• **Disadvantages**: Difficult to interpret
• **In general**: Combines results of different predictors (decision trees)
• **Ensemble methods** combine predictions of *weak classifiers*. 
Ensemble methods

• **Simple (a.k.a. weak) learners are good**
  – e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)
  – Low variance, don’t usually overfit

• **Simple (a.k.a. weak) learners are bad**
  – High bias, can’t solve hard learning problems

• Can we combine weak classifiers to form a strong classifier?
Ensemble methods: boosting

- Idea: given a weak learner, run it multiple times on (rewighted) training data, then let the learned classifiers vote
  - On each iteration $t$:
    - weight each training example by how incorrectly it was classified
    - Learn a hypothesis – $h_t$
    - A strength for this hypothesis – $\alpha_t$
  - Final classifier:
    - A linear combination of the votes of the different classifiers weighted by their strength
Ensemble methods: bagging

• **Bagging or bootstrap aggregation** a technique for reducing the variance of an estimated prediction function.
• Random forest is a bagging classifier with a committee of trees.
• For classification, a *committee* of trees each cast a vote for the predicted class.
Bagging reduces variance

Single tree decision boundary

100 bagged trees
Outline

• Support Vector Machines
  ✔ – Optimization Objective
  ✔ – Large Margin Intuition
  ✔ – Kernels

• Random Forest
  ✔ – Ensemble Methods
    – Algorithm
    – Node split
    – OOB error
Random Forest Algorithm

(a) Draw a **bootstrap sample** $Z^*$ of size $N$ from the training data.

(b) Grow a random-forest tree $T_b$ to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{min}$ is reached.

   i. Select **$m$ variables at random** from the $p$ variables.
   ii. Pick the best variable/split-point among the $m$.
   iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_B^1$.

To make a prediction at a new point $x$:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the $b$th random-forest
Overview
Intuition of Random Forest

The Random Forest Algorithm

De-correlation gives better accuracy

Out-of-bag error (OOB error)

Variable importance
New sample: old, retired, male, short

Tree predictions: diseased, healthy, diseased

Majority rule: diseased
Differences to standard tree

• Train each tree on bootstrap resample of data
  (Bootstrap resample of data set with N samples:
   Make new data set by drawing with replacement N samples; i.e., some samples will probably occur multiple times in new data set)

• For each split,
  consider only m randomly selected variables

• Don’t prune

• Fit B trees in such a way and use average or majority voting to aggregate results
Why Random Forests works:

• Mean Squared Error = Variance + $\text{Bias}^2$
  – If trees are sufficiently deep, they have very small bias

• How could we improve the variance over that of a single tree?
Why Random Forests works:

\[
\text{Var} \left( \frac{1}{B} \sum_{i=1}^{B} T_i(c) \right) = \frac{1}{B^2} \sum_{i=1}^{B} \sum_{j=1}^{B} \text{Cov}(T_i(x), T_j(x))
\]

\[
= \frac{1}{B^2} \sum_{i=1}^{B} \left( \sum_{j \neq i}^{B} \text{Cov}(T_i(x), T_j(x)) + \text{Var}(T_i(x)) \right)
\]

\[
= \frac{1}{B^2} \sum_{i=1}^{B} \left( (B-1)\sigma^2 \cdot \rho + \sigma^2 \right)
\]

\[
= \frac{B(B-1)\rho\sigma^2 + B\sigma^2}{B^2}
\]

\[
= \frac{(B-1)\rho\sigma^2}{B} + \frac{\sigma^2}{B}
\]

\[
= \rho\sigma^2 - \frac{\rho\sigma^2}{B} + \frac{\sigma^2}{B}
\]

\[
= \rho\sigma^2 + \frac{\sigma^2(1-\rho)}{B}
\]

Decreases, if number of trees B increases (irrespective of \( \rho \)).

Decreases, if \( \rho \) decreases, i.e., if \( m \) decreases.

De-correlation gives better accuracy.
Outline

• Support Vector Machines
  ✔ – Optimization Objective
  ✔ – Large Margin Intuition
  ✔ – Kernels

• Random Forest
  ✔ – Ensemble Methods
  ✔ – Algorithm
    – Node split
    – OOB error
Splitting the nodes

• At each node:

• $m$ predictor variables are selected at random from all the predictor variables $p$.

• The predictor variable that provides the best split, according to some objective function (e.g., information gain), is used to do a binary split on that node.

• At the next node, choose another $m$ variables at random from all predictor variables and do the same. (Breiman suggests $m=\frac{1}{2}\sqrt{p}$, $\sqrt{p}$, and $2\sqrt{p}$)
Use a subset of variables

• A randomly selected subset of variables is used to split each node
• The number of variables used is decided by the user (default=$sqrt(p)$)
• Smaller subset of variables produces less correlation but lower predictive power √

Optimum range of values is often quite wide
Outline

• Support Vector Machines
  ✔ – Optimization Objective
  ✔ – Large Margin Intuition
  ✔ – Kernels

• Random Forest
  ✔ – Ensemble Methods
  ✔ – Algorithm
  ✔ – Node split
    – OOB error
Generalization error \approx \text{Out-of-bag error}

- Similar to leave-one-out cross-validation, but almost without any additional computational burden
- OOB error is a random number, since based on random resamples of the data

Data:
- old, tall – healthy
- old, short – diseased
- young, tall – healthy
- young, short – diseased
- young, short – healthy
- young, tall – healthy
- old, short – diseased

Resampled Data:
- old, tall – healthy
- old, short – diseased
- young, tall – healthy
- young, tall – healthy

Out of bag samples:
- young, short – diseased
- young, short – healthy
- young, tall – healthy
- old, short – diseased

Out of bag (OOB) error rate: \( \frac{1}{4} = 0.25 \)
Advantages of Random Forest

- No need for pruning trees
- Accuracy and variable importance generated automatically
- Overfitting is not a problem
- Not very sensitive to outliers in training data
- Easy to set parameters
- Good performance
Trees vs Random Forest

+ Trees yield insight into decision rules
+ Rather fast
+ Easy to tune parameters

- Prediction of trees tend to have a high variance

+ RF has smaller prediction variance and therefore usually a better general performance
+ Easy to tune parameters

- Slower (can be parallelized)
- “Black Box”: Rather hard to get insights into decision rules
Random Forest vs LDA

+ Can model nonlinear class boundaries
+ OOB error “for free” (no CV needed)
+ Works on continuous and categorical responses (regression / classification)
+ Gives variable importance
+ Very good performance
- “Black box”
- Slower but fast enough

+ Very fast
+ Discriminants for visualizing group separation
+ Can read off decision rule
- Can model only linear class boundaries
- Mediocre performance
- No variable selection
- Only on categorical response
- Needs CV for estimating prediction error
Practical example

Dataset in CPP ~100,000 users

code jam

System.out.println("hello, world!");

fuzzy AST parser

preprocessing

Extract features

A

B

C

D

Random Forest

classification

majority vote

Wednesday - January 28, 2015
Results

<table>
<thead>
<tr>
<th>Application</th>
<th>Classes</th>
<th>Instances</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stylometric plagiarism detection</td>
<td>250 class</td>
<td>2250</td>
<td>95.3%</td>
</tr>
<tr>
<td>Copyright investigation</td>
<td>Two-class</td>
<td>360</td>
<td>98.9%</td>
</tr>
<tr>
<td>Authorship verification</td>
<td>Two-class/One-class</td>
<td>960</td>
<td>93.2%</td>
</tr>
</tbody>
</table>

- A new principled method with a robust syntactic feature set for performing source code stylometry.
- Our authorship attribution technique is impervious to common off-the-shelf source code obfuscators.
- Insights about programmers and coding style.
  - Implementing harder functionality makes programming style more unique
  - Better programmers have more distinct coding style
Common Applications of Random Forests

• Classification
  – Land cover classification
  – Cloud/shadow screening

• Regression
  – Biomass mapping
  – Continuous fields (percent cover) mapping
Outline

• Support Vector Machines
  ✔ – Optimization Objective
  ✔ – Large Margin Intuition
  ✔ – Kernels

• Random Forest
  ✔ – Ensemble Methods
  ✔ – Algorithm
  ✔ – Node split
  ✔ – OOB error
Qiong Feng

Avichal Chum
Sebastian Thrun, Mike Montemerlo, Andrei Aron Stanford Artificial Intelligence Lab (SAIL), Probabilistic Terrain Analysis For High-Speed Desert Driving.

Amelia Solon