

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

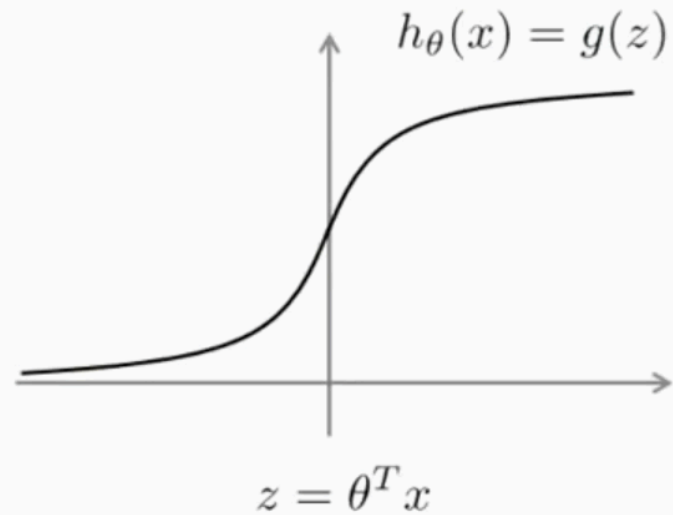
- Support Vector Machines
 - Optimization Objective
 - Large Margin Intuition
 - Kernels
- Random Forest
 - Ensemble Methods
 - Algorithm
 - Node split
 - OOB error

Outline

- Support Vector Machines
 - from Dr. Andrew Ng's notes
- Random Forest
 - from Dr. Leo Breiman
 - from Dr. Markus Kalisch

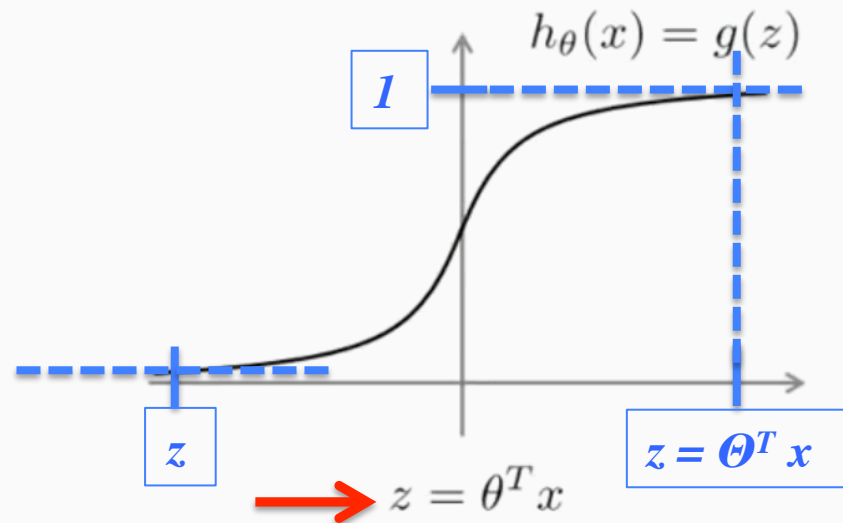
Alternative view of logistic regression

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



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 $h_{\theta}(x) \approx 0$, $\theta^T x \ll 0$

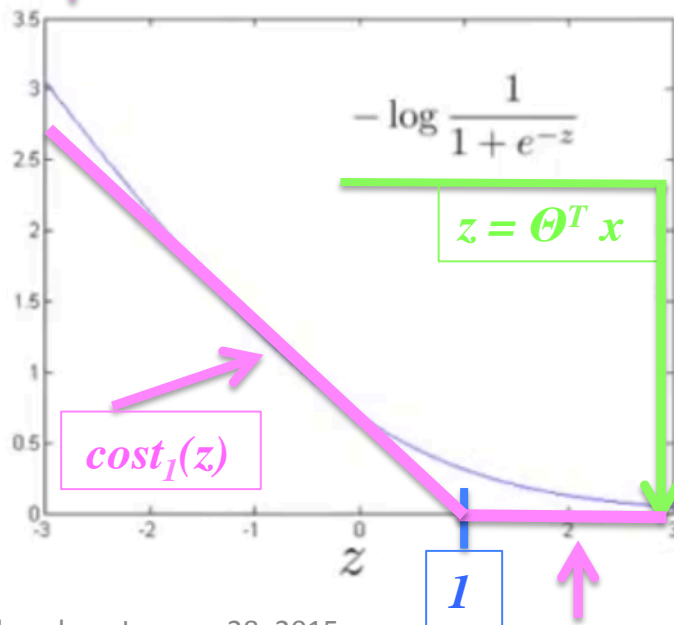
Alternative view of logistic regression

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

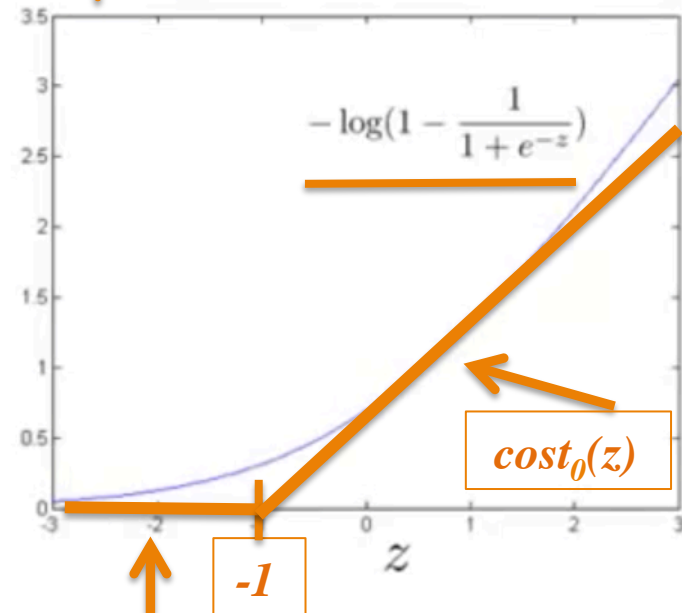
example (x, y) contributes

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

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



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



Support vector machine

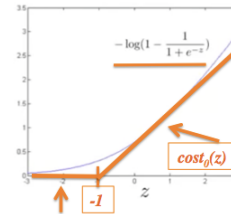
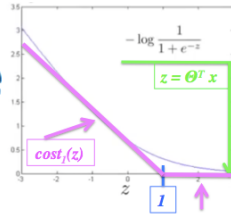
Logistic regression:

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

Support vector machine:

Support vector machine

Logistic regression:



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

Support vector machine:

$$\min_u (u - 5)^2 + 1 \quad \rightarrow u = 5$$

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

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

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

Support vector machine

Logistic regression:

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

A

Support vector machine:

$$\min_u (u - 5)^2 + 1 \quad \rightarrow u = 5$$

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

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

Logistic regression:

$$\text{A} + \lambda \text{B}$$

Support vector machine:

$$\text{C} \text{A} + \text{B}$$

$$\text{C} \approx (1/\lambda)$$

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

SVM hypothesis

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

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

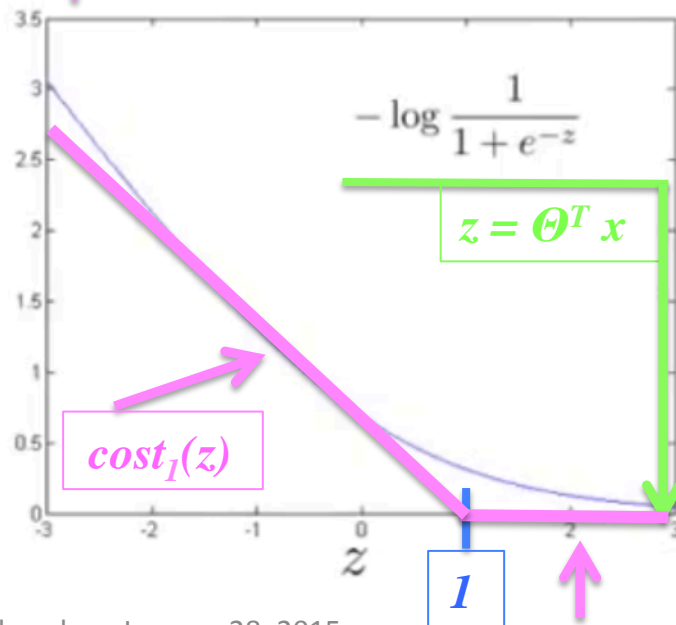
- Support Vector Machines
 - ✓ – Optimization Objective
 - Large Margin Intuition
 - Kernels
- Random Forest
 - Ensemble Methods
 - Algorithm
 - Node split
 - OOB error

Large Margin Intuition

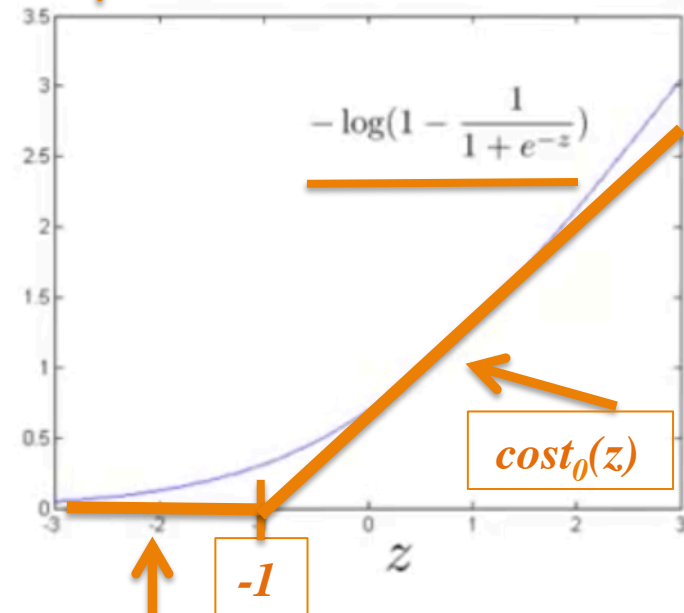
SVM hypothesis

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

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



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



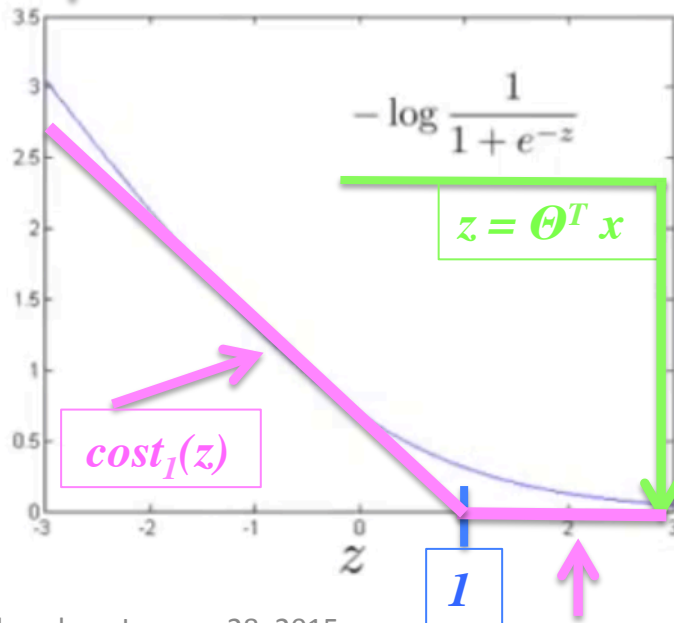
Large Margin Intuition

SVM hypothesis

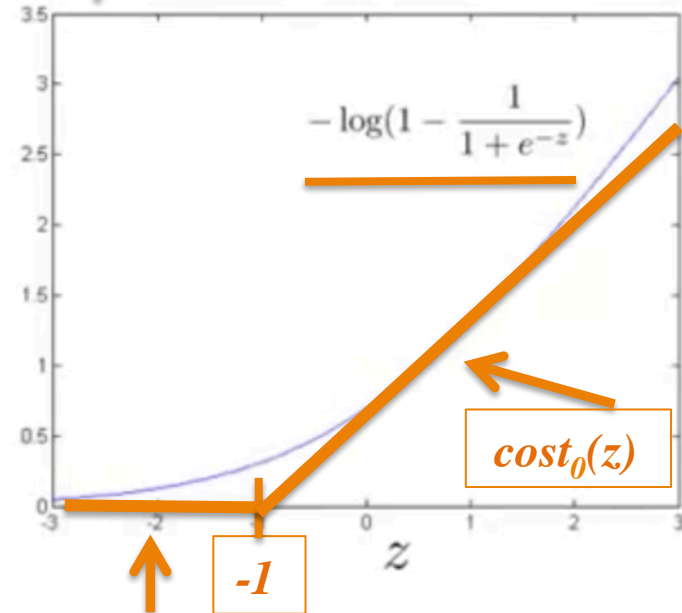
$$\min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

(not just > 0) **C = 100,000** (not just < 0)

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

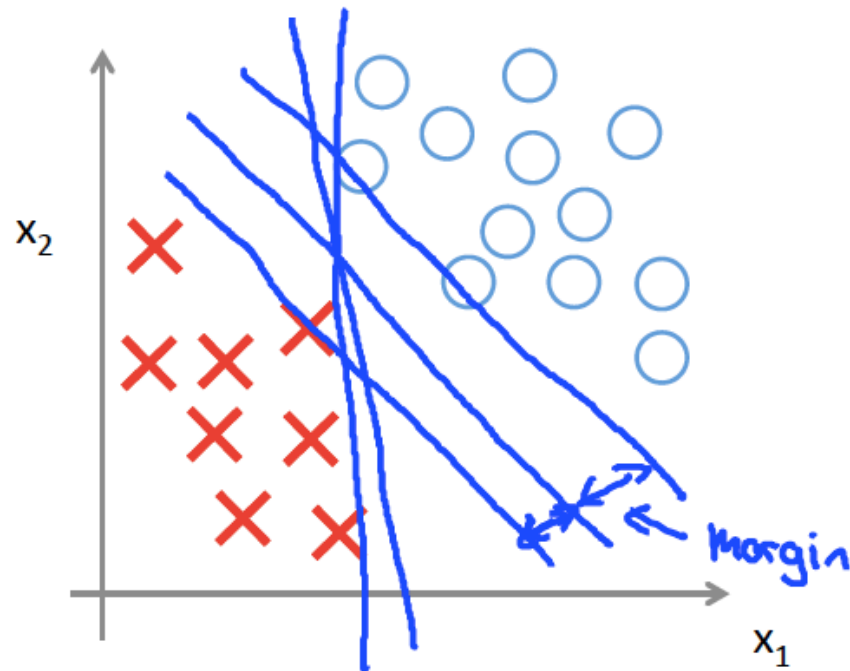


If $y = 0$ (want $\theta^T x \leq -1$):



Large Margin Intuition

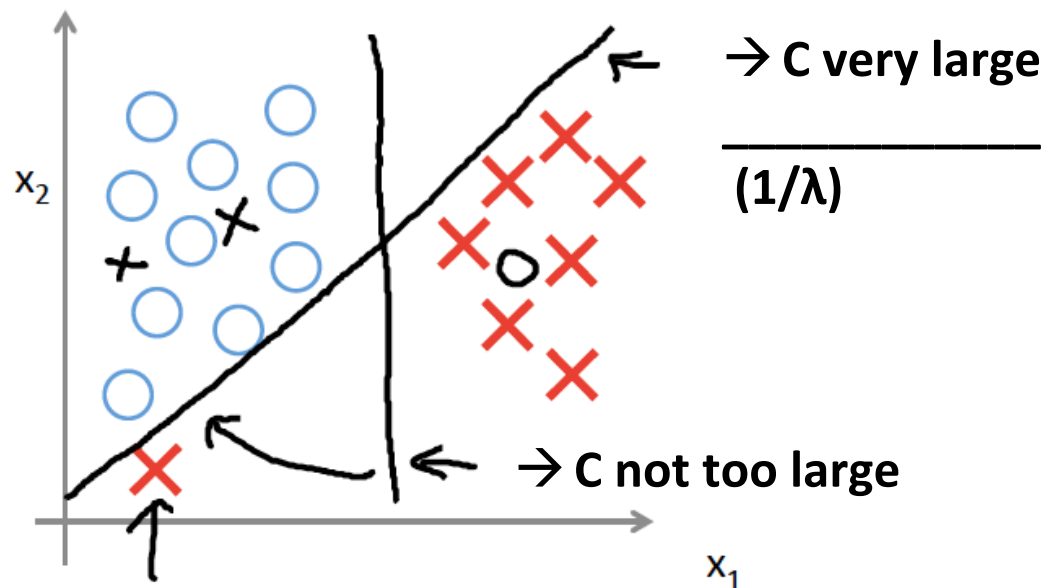
SVM Decision Boundary: Linearly separable case



Large margin classifier

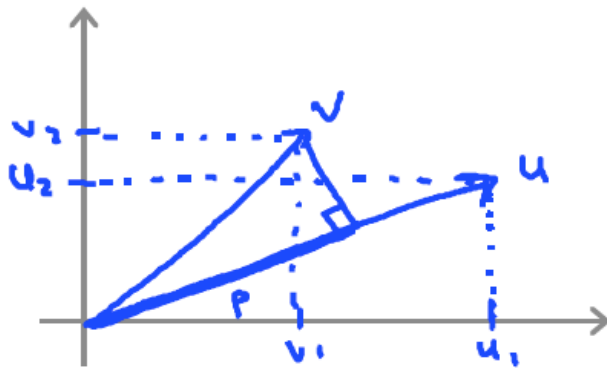
Large Margin Intuition

Large margin classifier in presence of outliers



Math Behind the Large Margin

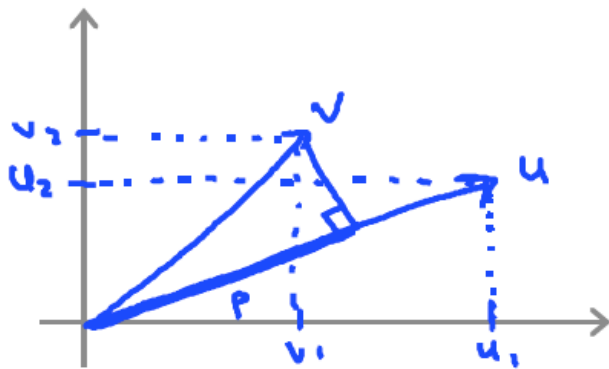
Vector Inner Product



$$\rightarrow u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \rightarrow v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Math Behind the Large Margin

Vector Inner Product



$$\rightarrow u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \rightarrow v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

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

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

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

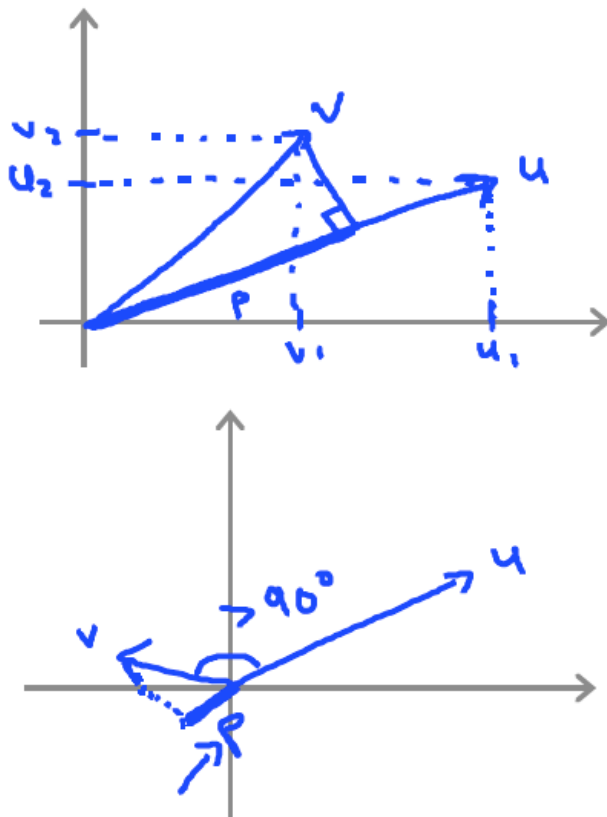
$$\begin{aligned} u^T v &= \underline{p} \cdot \|u\| \leftarrow = v^T u \\ \text{Signed} \quad &= u_1 v_1 + u_2 v_2 \leftarrow p \in \mathbb{R} \end{aligned}$$

$$u^T v = p \cdot \|u\|$$

$$p < 0$$

Math Behind the Large Margin

Vector Inner Product



$$\rightarrow u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \rightarrow v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

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

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

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

$$\begin{aligned} u^T v &= \underline{p} \cdot \|u\| \leftarrow = v^T u \\ \text{Signed} \quad &= u_1 v_1 + u_2 v_2 \leftarrow p \in \mathbb{R} \end{aligned}$$

$$u^T v = p \cdot \|u\|$$

$$p < 0$$

Math Behind the Large Margin

SVM Decision Boundary

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} (\theta_1^2 + \theta_2^2) = \frac{1}{2} \left(\sqrt{\theta_1^2 + \theta_2^2} \right)^2 = \frac{1}{2} \|\theta\|^2$$

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

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

Simplification: $\theta_0 = 0$. $n=2$

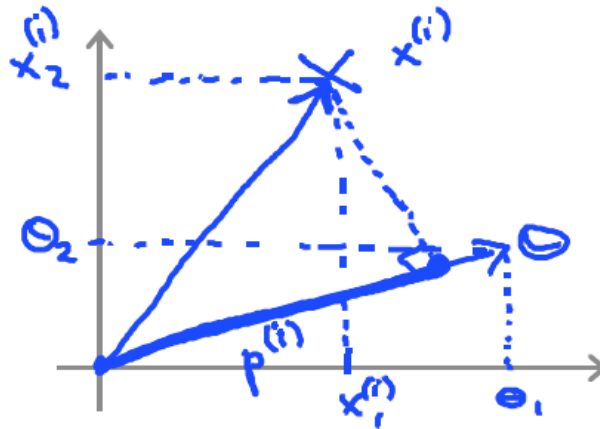
$$\omega = (\sqrt{\omega'})^2$$

$$= \|\theta\|$$

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \theta_0 = 0$$

$$\theta^T x^{(i)} = ?$$

↑ ↑
u^T v



$$\theta^T x^{(i)} = \boxed{p^{(i)} \cdot \|\theta\|} \leftarrow$$

$$= \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} \leftarrow$$

Math Behind the Large Margin

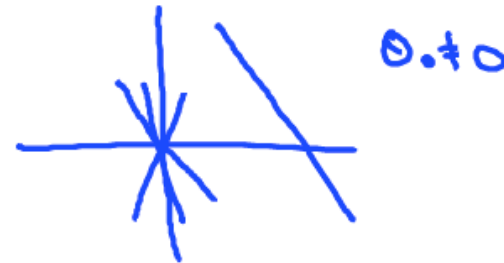
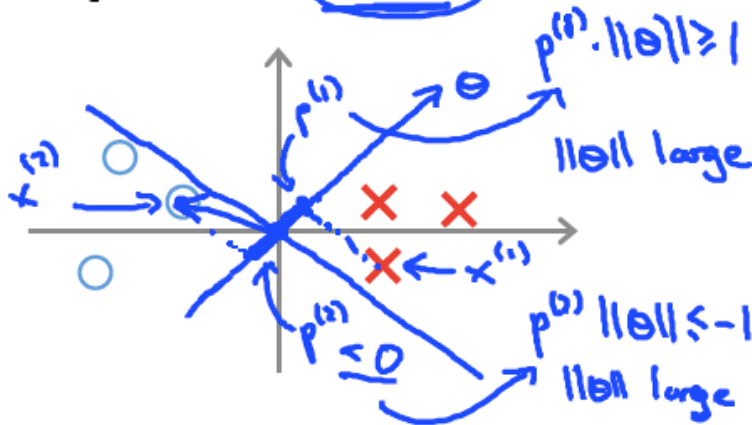
SVM Decision Boundary

$$\Rightarrow \min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} \|\theta\|^2 \leftarrow$$

$$\text{s.t. } \left. \begin{array}{ll} p^{(i)} \cdot \|\theta\| \geq 1 & \text{if } y^{(i)} = 1 \\ p^{(i)} \cdot \|\theta\| \leq -1 & \text{if } y^{(i)} = 0 \end{array} \right\} C \text{ very large}$$

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

Simplification: $\theta_0 = 0$



Math Behind the Large Margin

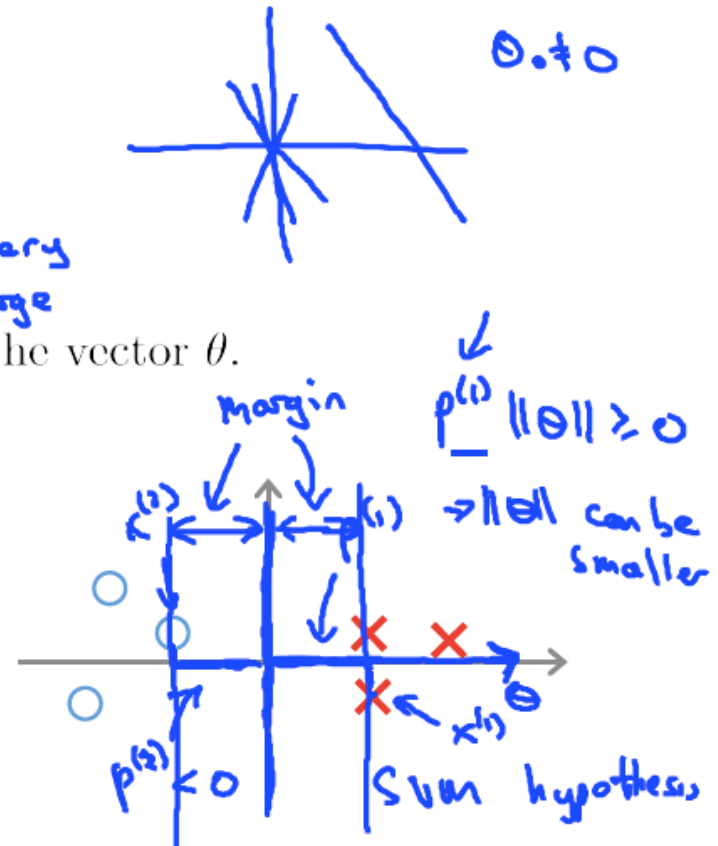
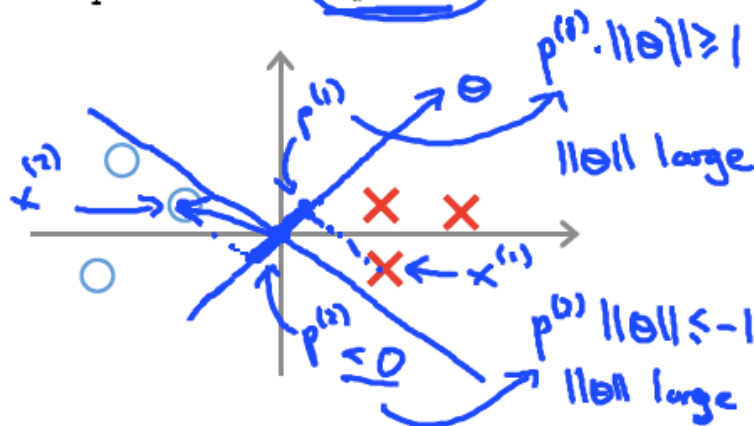
SVM Decision Boundary

$$\Rightarrow \min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} \|\theta\|^2 \leftarrow$$

$$\text{s.t. } \left. \begin{array}{ll} p^{(i)} \cdot \|\theta\| \geq 1 & \text{if } y^{(i)} = 1 \\ p^{(i)} \cdot \|\theta\| \leq -1 & \text{if } y^{(i)} = 0 \end{array} \right\} C \text{ very large}$$

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

Simplification: $\theta_0 = 0$

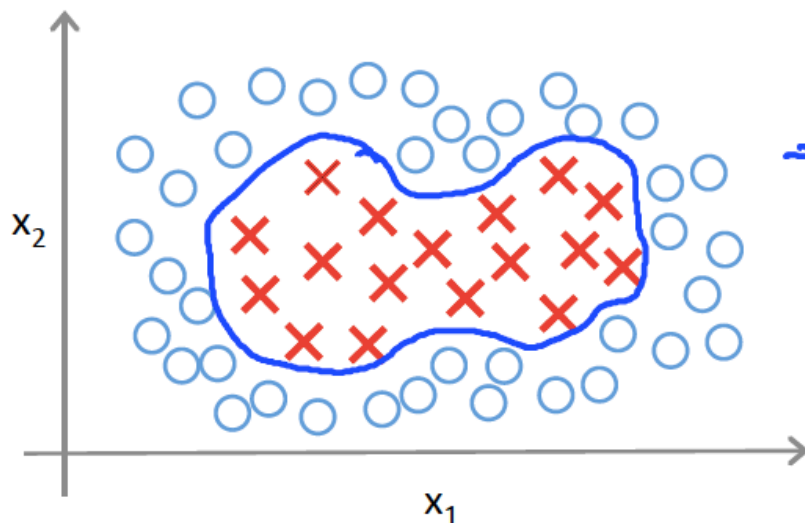


Outline

- Support Vector Machines
 - ✓ – Optimization Objective
 - ✓ – Large Margin Intuition
 - Kernels
- Random Forest
 - Ensemble Methods
 - Algorithm
 - Node split
 - OOB error

Kernels

Non-linear Decision Boundary



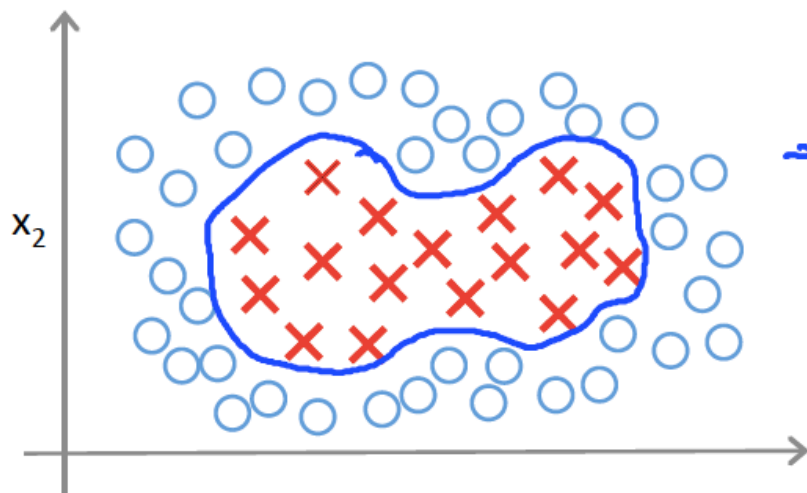
Predict $y = 1$ if

$$\rightarrow \theta_0 + \theta_1 \underline{x_1} + \theta_2 \underline{x_2} + \theta_3 \underline{x_1 x_2} \\ + \theta_4 \underline{x_1^2} + \theta_5 \underline{x_2^2} + \dots \geq 0$$

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

Kernels

Non-linear Decision Boundary



Predict $y = 1$ if

$$\rightarrow \theta_0 + \theta_1 \underline{x_1} + \theta_2 \underline{x_2} + \theta_3 \underline{x_1 x_2} + \theta_4 \underline{x_1^2} + \theta_5 \underline{x_2^2} + \dots \geq 0$$

$$h_{\theta}(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \dots \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

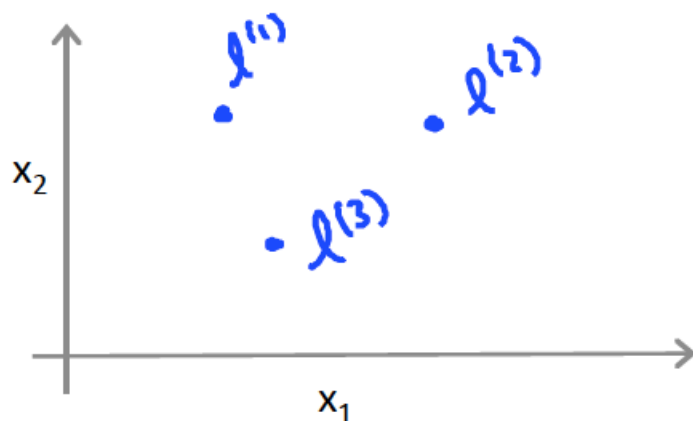
$$\rightarrow \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \dots$$

$$f_1 = x_1, \quad f_2 = x_2, \quad f_3 = x_1 x_2, \quad f_4 = x_1^2, \quad f_5 = x_2^2, \dots$$

Is there a different / better choice of the features f_1, f_2, f_3, \dots ?

Kernels

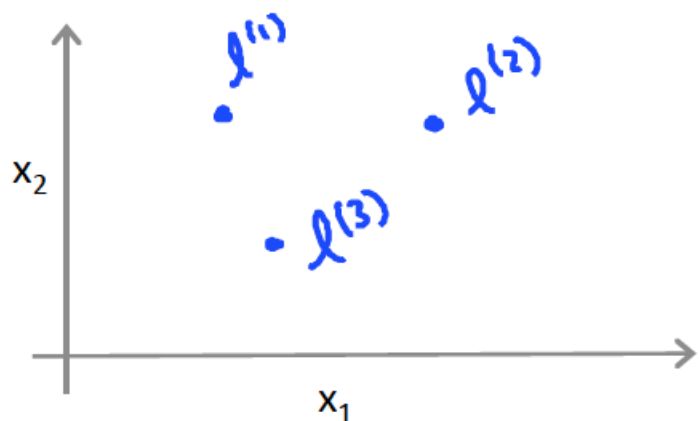
Kernel



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

Kernels

Kernel



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

Given x :

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

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

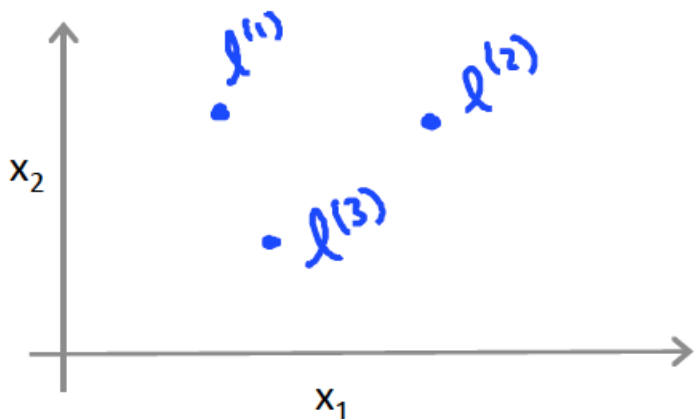
$$f_3 = \text{similarity}(x, l^{(3)}) = \exp(\dots)$$

Handwritten annotations for the first equation:

- $\|w\|$ with an arrow pointing to the denominator $2\sigma^2$.
- $\|x - l^{(1)}\|^2$ with an arrow pointing to the numerator.
- A large arrow pointing from the right towards the equation.

Kernels

Kernel



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

x_2
 $\bullet l^{(3)}$
 x_1
 Given x :
 $f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$
 $f_2 = \text{similarity}(x, l^{(2)}) = \exp\left(-\frac{\|x - l^{(2)}\|^2}{2\sigma^2}\right)$
 $f_3 = \text{similarity}(x, l^{(3)}) = \exp(\dots)$
kernel (Gaussian kernels) $k(x, l^{(i)})$

Kernels

Kernels and Similarity

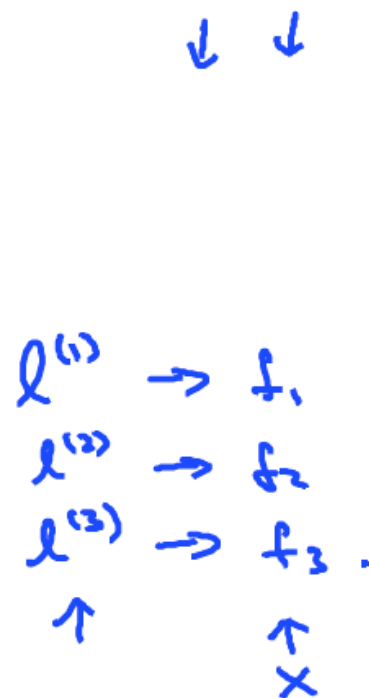
$$f_1 = \text{similarity}(x, \underline{l^{(1)}}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$$

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

$$f_1 \approx \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx 1$$

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

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



Kernels

Example:

$$\rightarrow l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$$

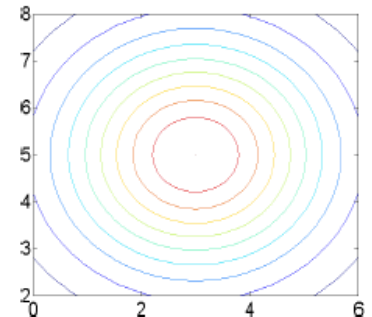
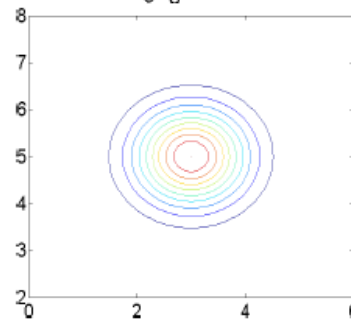
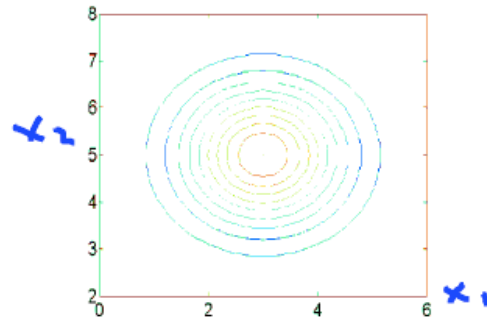
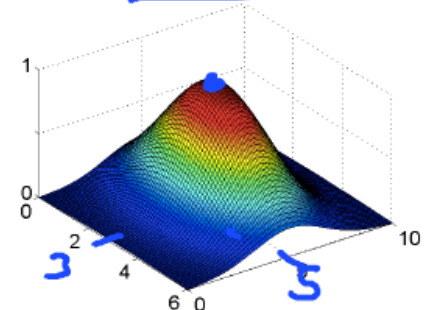
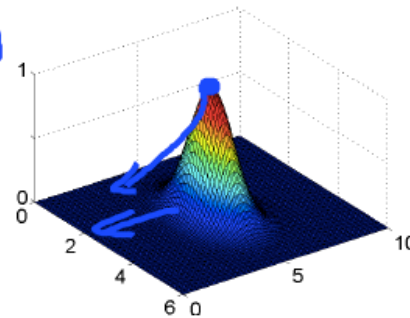
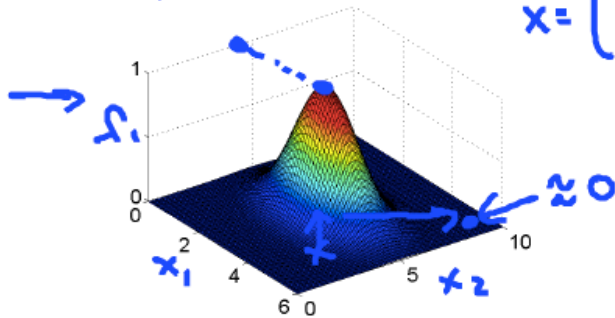
$$\rightarrow \sigma^2 = 1$$

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

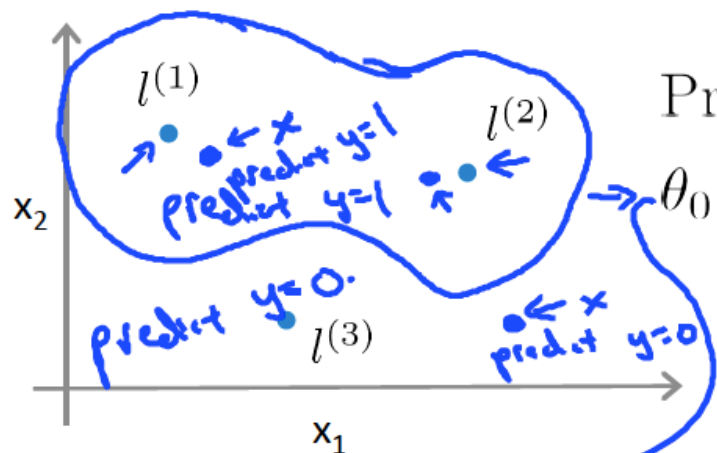
$$\sigma^2 = 0.5$$

$$\sigma^2 = 3$$

$$x = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$$



Kernels



Predict "1" when

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

↑
x

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

$$f_1 \approx 1, f_2 \approx 0, f_3 \approx 0.$$

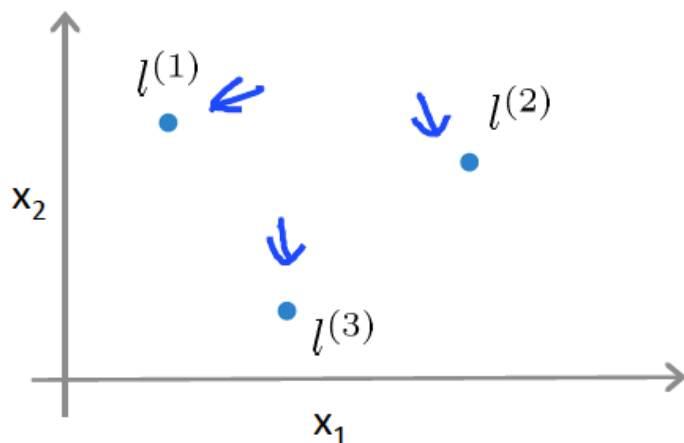
$$\begin{aligned} \rightarrow \theta_0 + \theta_1 \times 1 + \theta_2 \times 0 + \theta_3 \times 0 \\ = -0.5 + 1 = 0.5 \geq 0 \end{aligned}$$

$$f_1, f_2, f_3 \approx 0$$

$$\rightarrow \underline{\theta_0} + \theta_1 \underline{f_1} + \dots \approx -0.5 < 0$$

Kernels

Choosing the landmarks

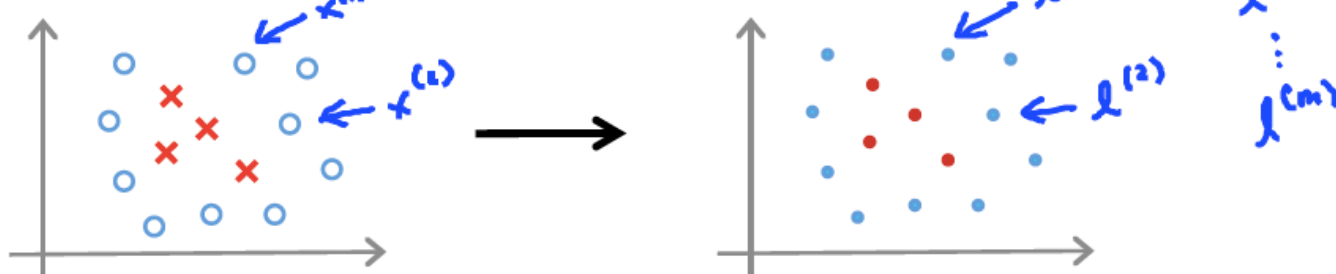


Given x :

$$\begin{aligned} \rightarrow f_i &= \text{similarity}(x, l^{(i)}) \\ &= \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right) \end{aligned}$$

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

Where to get $l^{(1)}, l^{(2)}, l^{(3)}, \dots$?



Kernels

SVM with Kernels

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

Given example x :

$$\begin{aligned} \rightarrow f_1 &= \text{similarity}(x, l^{(1)}) \\ \rightarrow f_2 &= \text{similarity}(x, l^{(2)}) \\ &\vdots \end{aligned}$$

$$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \quad f_0 = 1$$

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

$$\begin{aligned} x^{(i)} \rightarrow \begin{bmatrix} f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} &= \begin{aligned} f_1^{(i)} &= \text{sim}(x^{(i)}, l^{(1)}) \\ f_2^{(i)} &= \text{sim}(x^{(i)}, l^{(2)}) \\ &\vdots \\ f_i^{(i)} &= \text{sim}(x^{(i)}, l^{(i)}) = \exp(-\frac{0}{2\sigma_i^2}) = 1 \\ &\vdots \\ f_m^{(i)} &= \text{sim}(x^{(i)}, l^{(m)}) \end{aligned} \end{aligned}$$

$$x^{(i)} \in \mathbb{R}^{n+1} \quad \text{(or } \mathbb{R}^n \text{)}$$

$$f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} \quad f_0^{(i)} = 1$$

Kernels

SVM with Kernels

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

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

$$\Theta \in \mathbb{R}^{n+1}$$

$$\Theta_0 f_0 + \Theta_1 f_1 + \dots + \Theta_m f_m$$

Training:

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

Handwritten notes: $\theta^T f^{(i)}$, $\theta^T f^{(i)}$, θ_j , $n=m$

$$\sum_{j=1}^m \theta_j^2 = \Theta^T \Theta$$

$$\Theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix}$$

Handwritten notes: $\|\theta\|^2$, $\Theta^T M \Theta$, (ignore Θ_0), $M = 10,000$

Kernels

SVM parameters:

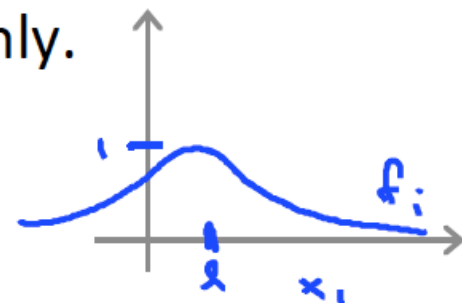
$C (= \frac{1}{\lambda})$. \rightarrow Large C : Lower bias, high variance.
 \rightarrow Small C : Higher bias, low variance.

(small λ)

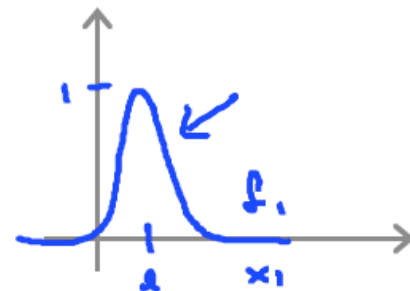
(large λ)

σ^2 Large σ^2 : Features f_i vary more smoothly.
 \rightarrow Higher bias, lower variance.

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



Small σ^2 : Features f_i vary less smoothly.
 Lower bias, higher variance.



Using an SVM

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

Need to specify:

→ Choice of parameter C .

Choice of kernel (similarity function):

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

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


$\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n \geq 0$
 → n large, m small $x \in \mathbb{R}^{n+1}$

→ Gaussian kernel:

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

Need to choose σ^2 .

$x \in \mathbb{R}^n$, n small
 and/or n large



Using an SVM

Kernel (similarity) functions:

function $f = \text{kernel}(\underline{x1}, \underline{x2})$

$$f = \exp\left(-\frac{\|\underline{x1} - \underline{x2}\|^2}{2\sigma^2}\right)$$

return

$x^{(i)}$
 $l^{(i)} = x^{(i)}$

$x \rightarrow \begin{matrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{matrix}$

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

$$\|x - l\|^2$$

$$v = x - l$$

$$\|v\|^2 = v_1^2 + v_2^2 + \dots + v_n^2$$

$$= \underbrace{(x_1 - l_1)^2}_{1000 \text{ feet}^2} + \underbrace{(x_2 - l_2)^2}_{1-5 \text{ bedrooms}} + \dots + (x_n - l_n)^2$$

$x \in \mathbb{R}^n$

Using an SVM

Other choices of kernel

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

→ (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:

$$k(x, l) = (x^T l)^3, \quad (x^T l)^2 + 0, \quad (x^T l + 1)^3, \quad (x^T l + 5)^4$$

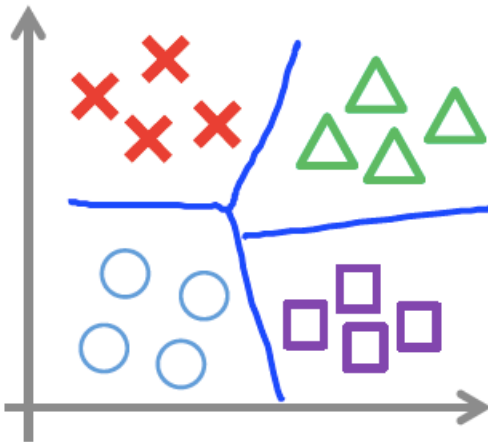
Handwritten notes: $(x^T l + \text{constant})^{\text{degree}}$ with arrows pointing to the constant and degree terms in the polynomial examples.

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

$$\text{sim}(x, l)$$

Multi-class classification

Multi-class classification



$$y \in \{1, 2, 3, \dots, K\}$$

↑

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 $i = 1, 2, \dots, K$), get $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(K)}$
Pick class i with largest $(\theta^{(i)})^T x$

↑ ↑ ... ↑
 $y=1$ $y=2$ $\theta=K$

Logistic regression vs. SVMs

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

→ If n is large (relative to m): (e.g. $n \geq m$, $n = \underline{10,000}$, $m = \underline{10} \dots \underline{1,000}$)

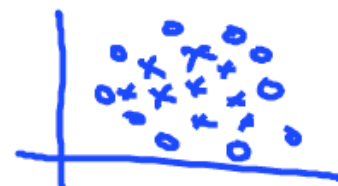
→ Use logistic regression, or SVM without a kernel ("linear kernel")

→ If n is small, m is intermediate: ($n = \underline{1-1,000}$, $m = \underline{10-10,000}$) ←

→ Use SVM with Gaussian kernel

If n is small, m is large: ($n = \underline{1-1,000}$, $m = \underline{50,000+}$)

→ Create/add more features, then use logistic regression or SVM
without a kernel

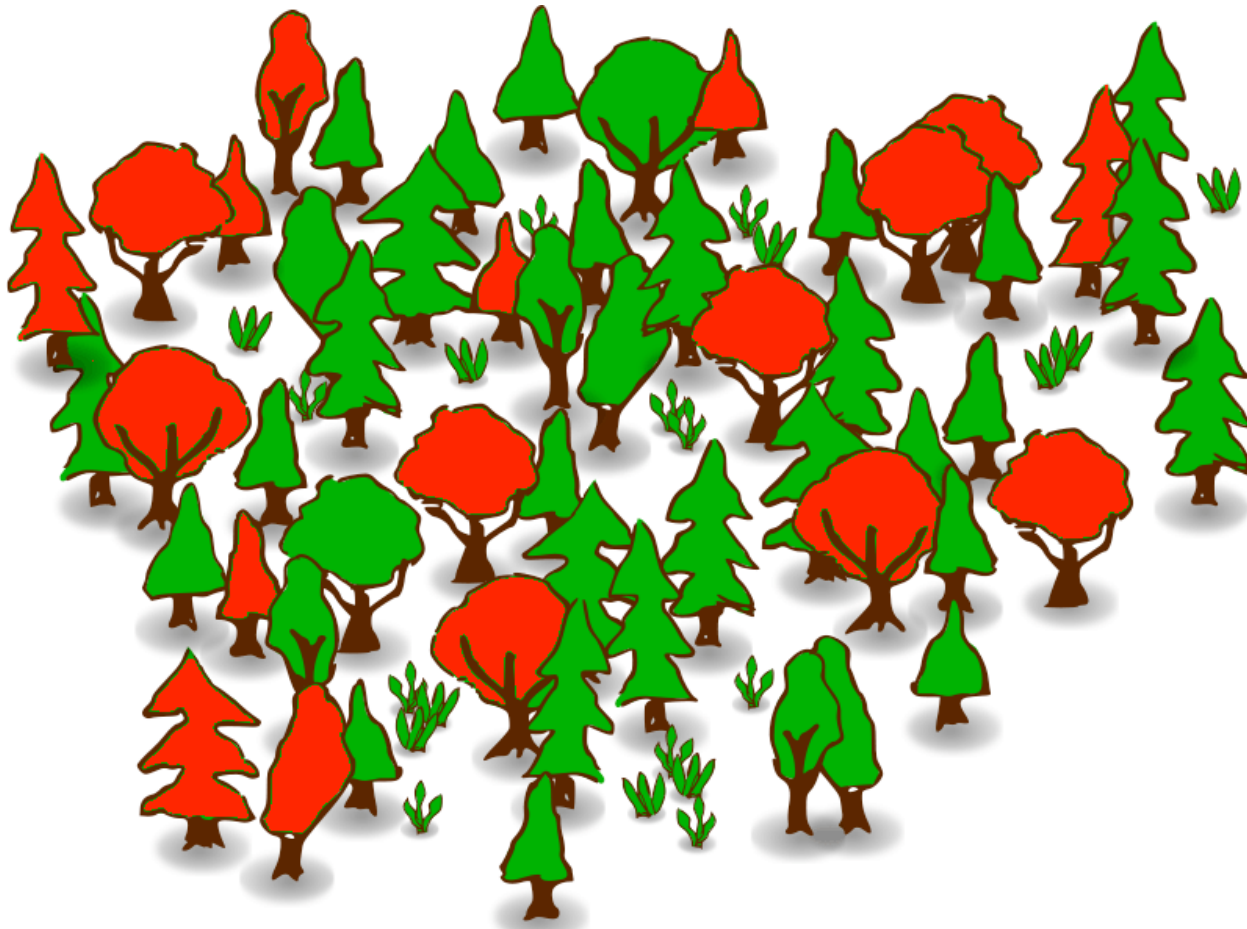


→ Neural network likely to work well for most of these settings, but may be slower to train.

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?

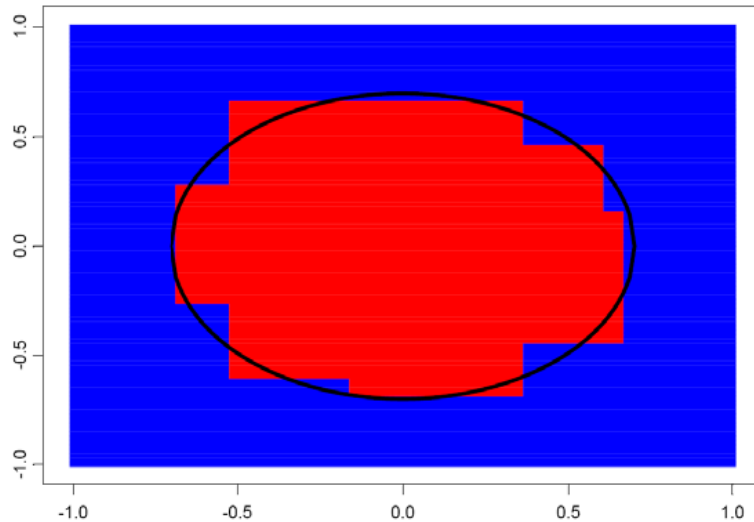
Ensemble methods: boosting

- Idea: given a weak learner, run it multiple times on (reweighted) 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 – α_t
- Final classifier:
 - A linear combination of the votes of the different classifiers weighted by their strength

Ensemble methods: bagging

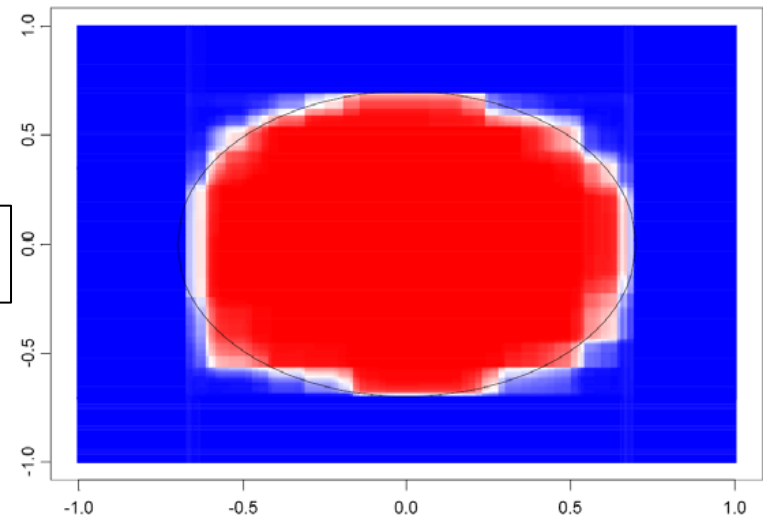
- ***Bagging or bootstrap aggregation*** a technique for reducing the variance of an estimated prediction function.
- Random forest is a bagging classifier with a committee of trees.
- For classification, a *committee* of trees each cast a vote for the predicted class.

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

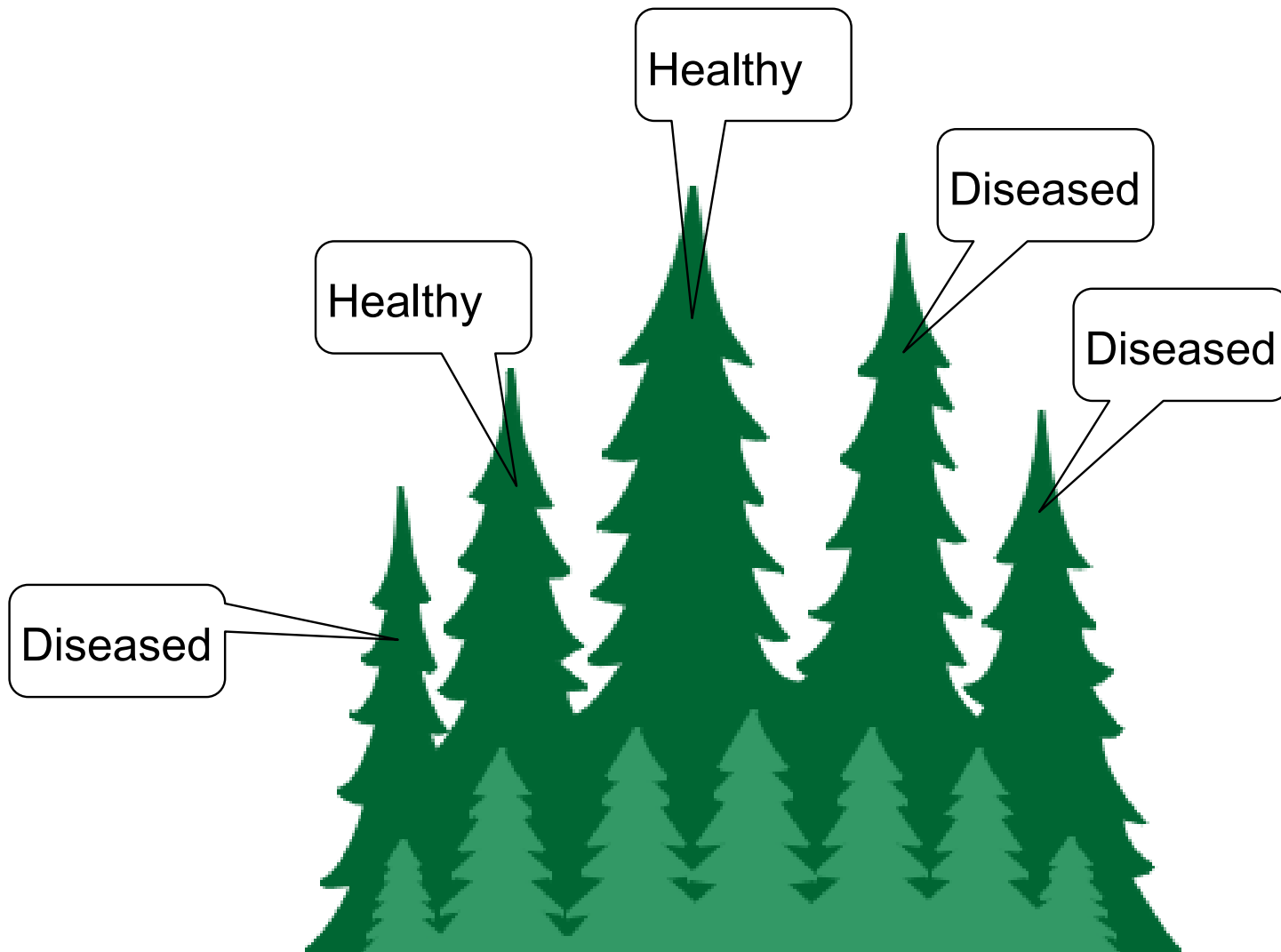
Random Forest Algorithm

- (a) Draw a **bootstrap sample** \mathbf{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 $\{T_b\}_1^B$.

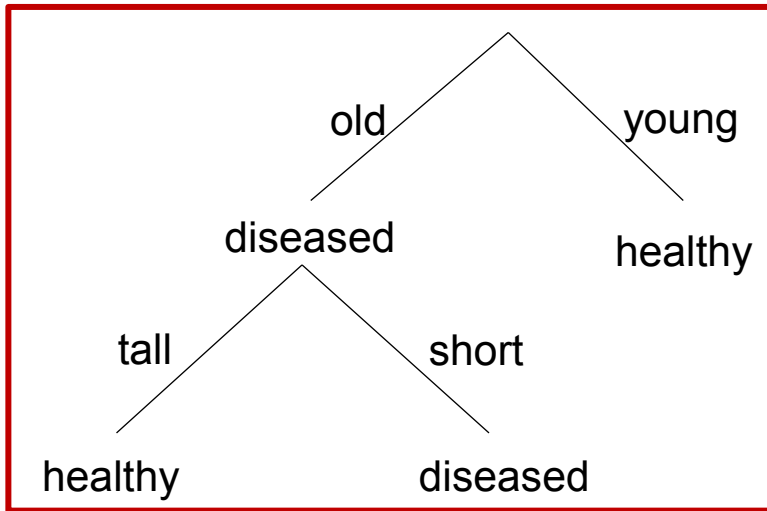
To make a prediction at a new point x :

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

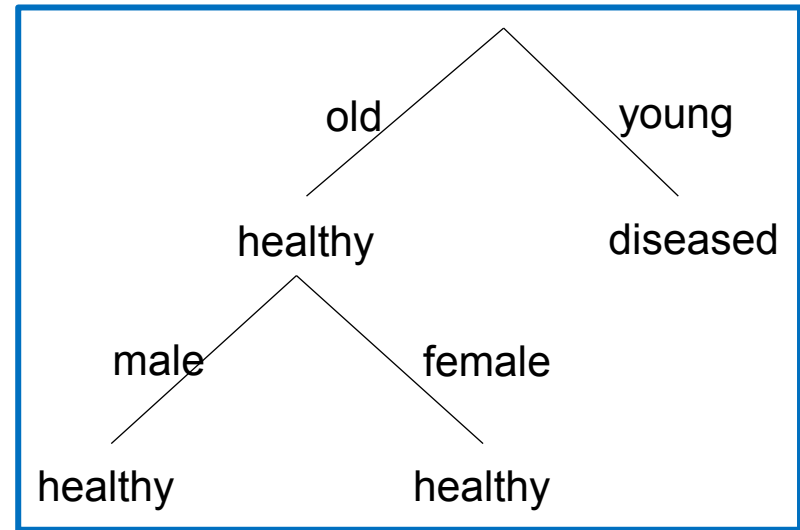
Classification: Let $\hat{C}_b(x)$ be the class prediction of the b th random-forest



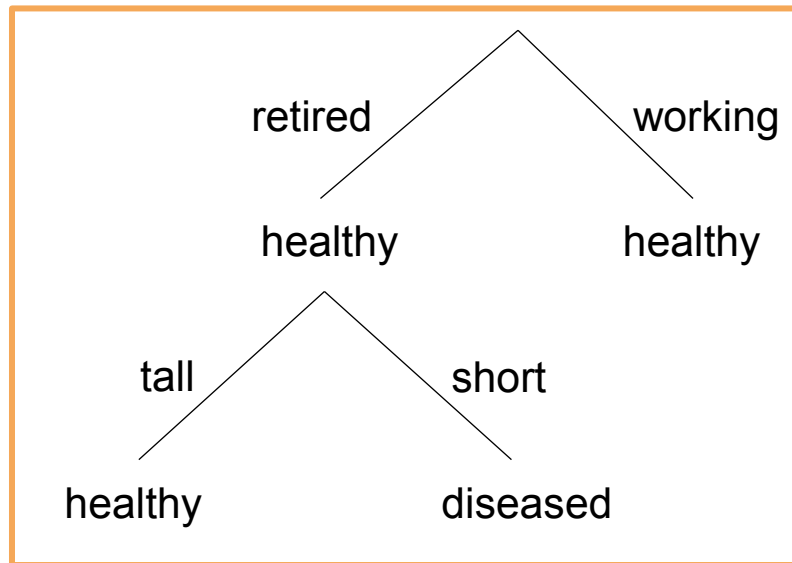
Tree 1



Tree 2



Tree 3



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 + Bias²
 - 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:

$$\begin{aligned}
 \text{Var} \left(\frac{1}{B} \sum_{i=1}^B T_i(c) \right) &= \frac{1}{B^2} \sum_{i=1}^B \sum_{j=1}^B \text{Cov}(T_i(x), T_j(x)) \\
 &= \frac{1}{B^2} \sum_{i=1}^B \left(\sum_{j \neq i}^B \text{Cov}(T_i(x), T_j(x)) + \text{Var}(T_i(x)) \right) \\
 &= \frac{1}{B^2} \sum_{i=1}^B \left((B-1)\sigma^2 \cdot \rho + \sigma^2 \right) \\
 &= \frac{B(B-1)\rho\sigma^2 + B\sigma^2}{B^2} \\
 &= \frac{(B-1)\rho\sigma^2}{B} + \frac{\sigma^2}{B} \\
 &= \rho\sigma^2 - \frac{\rho\sigma^2}{B} + \frac{\sigma^2}{B} \\
 &\Rightarrow \boxed{\rho\sigma^2} + \boxed{\sigma^2 \frac{1-\rho}{B}}
 \end{aligned}$$

Decreases, if ρ decreases, i.e., if m decreases

De-correlation gives better accuracy

Decreases, if number of trees B increases (irrespective of ρ)

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 (eg 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

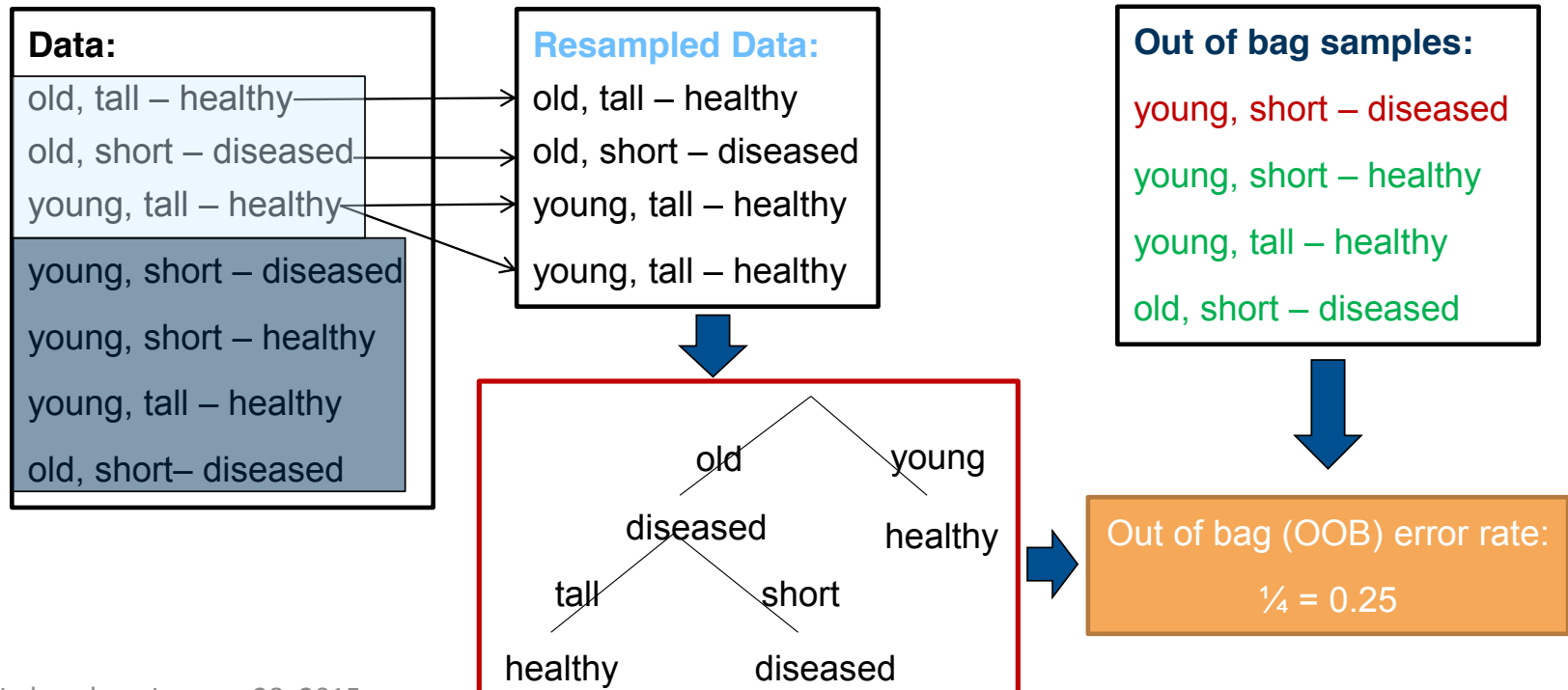
- A randomly selected subset of variables is used to split each node
- The number of variables used is decided by the user (**default=sqrt(p)**)
- Smaller subset of variables produces less correlation but lower predictive power
Optimum range of values is often quite wide

Outline

- Support Vector Machines
 - ✓ – Optimization Objective
 - ✓ – Large Margin Intuition
 - ✓ – Kernels
- Random Forest
 - ✓ – Ensemble Methods
 - ✓ – Algorithm
 - ✓ – Node split
 - OOB error

Generalization error \approx 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



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

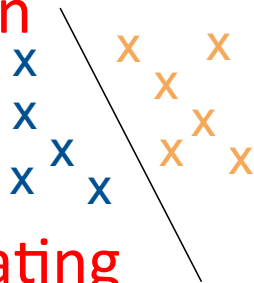
- + Can model nonlinear class boundaries
- + OOB error “for free” (no CV needed)
- + Works on continuous and categorical responses (regression / classification)
- + Gives variable importance
- + Very good performance

- “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

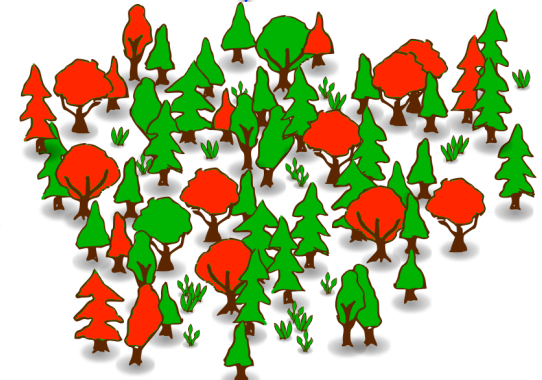
```
System.out.println("hello, world!");
```

preprocessing

fuzzy AST parser



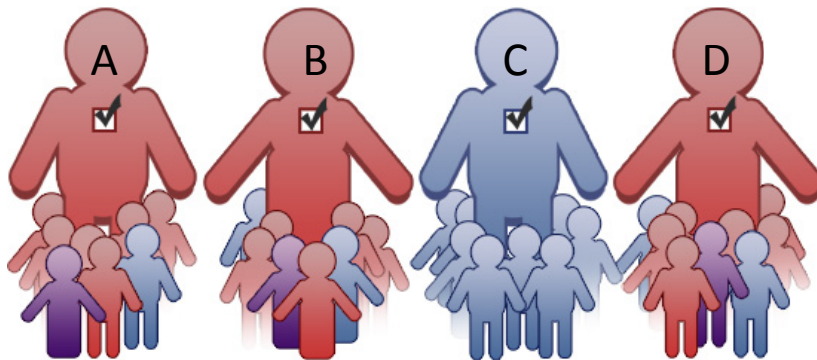
Extract features



Random Forest

classification

majority
vote



Results

Application	Classes	Instances	Result
Stylometric plagiarism detection	250 class	2250	95.3%
Copyright investigation	Two-class	360	98.9%
Authorship verification	Two-class/One-class	960	93.2%

- 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

Presenters & papers

Qiong Feng

Jiang, Yue, et al. "Comparing design and code metrics for software quality prediction." Proceedings of the 4th international workshop on Predictor models in software engineering. ACM, 2008.

Avichal Chum

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

Amelia Solon

E.T. Solovey, M. Zec, E. Garcia Perez, B. Reimer, B. Mehler. Classifying Driver Workload Using Physiological and Driving Performance Data: Two Field Studies. Proc. ACM Conference on Human Factors in Computing Systems CHI '14, ACM Press (2014).