Machine Learning

4771

$Pr(Data \mid \theta)$ likelihood

$Pr(\theta)$ prior

$Pr(Data, \theta) = Pr(\theta) P(Data \mid \theta)$

$$Pr(\theta \mid Data) = \frac{Pr(\theta) \cdot P(Data \mid \theta)}{Pr(Data)}$$

$= \int P(\theta) P(Data \mid \theta)$

$$Exp = \frac{\int \theta \, Pr(\theta) \, Pr(Data \mid \theta)}{\int Pr(\theta) \, Pr(Data \mid \theta)}$$

Instructor: Itsik Pe'er

Administration/HW/quiz

- Legibility
- likelihood: Prob(entire data)
  loss: log-likelihood contribution by datapoint
- Limits of distributions

Binomial $\xrightarrow[n\to\infty]{}$ Normal

Geometric $\xrightarrow[p\to 0]{}$ Exp

Sample mean $\xrightarrow[n\to\infty]{}$ expectation

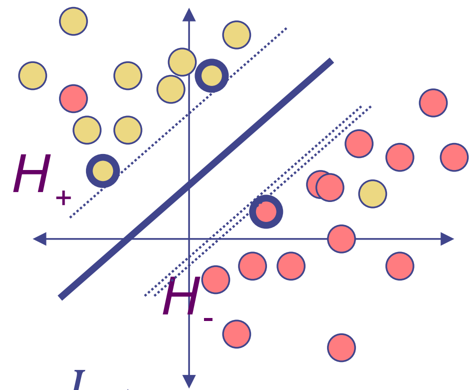
Binomial $\xrightarrow[n\to\infty,\ p\to 0]{}$ Poisson, $n \cdot p = \lambda$

- OLS: Poly, N>D
- EAP(Bernouli)

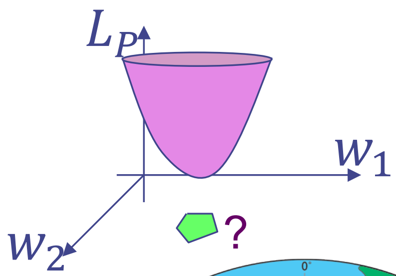
ML: $\frac{h}{N}$ → $\frac{h+1}{N+2}$

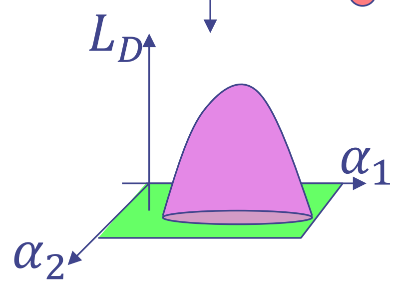
Add 1 smoothing

Reminder: SVM



- Non-separable





- Non linear

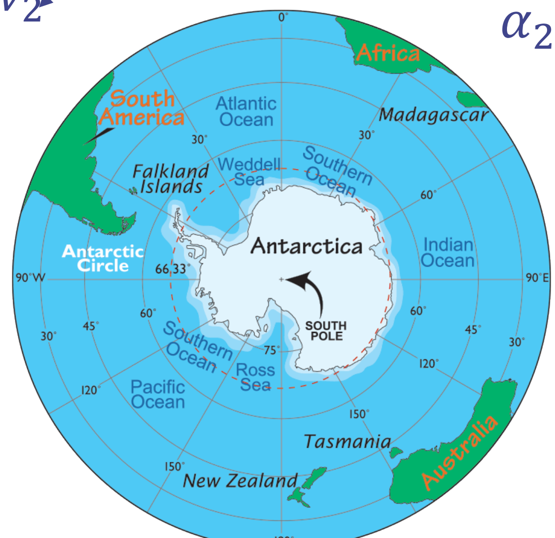
$x^{(1)}$

$x^{(2)}$

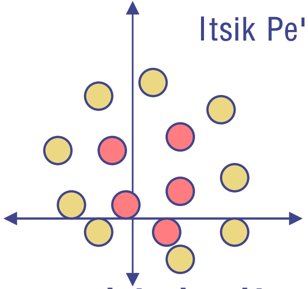
$x(1)^2$

$x(2)^2$

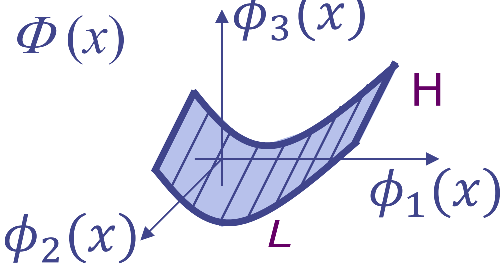
$x(1)x(2)$



Nonlinear SVMs



- What if the problem is not linear?
- We can use our old trick...
- Map d-dimensional x data from L-space to high dimensional H (Hilbert) feature-space via basis functions $\Phi(x)$
- For example, quadratic classifier:



$$x_i \to \phi(x_i) \;\; via \;\; \phi(\vec{x}) = \begin{bmatrix} \vec{x} \\ vec(\vec{x}\vec{x}^T) \end{bmatrix}$$

- Call ϕ's feature vectors computed from original x inputs

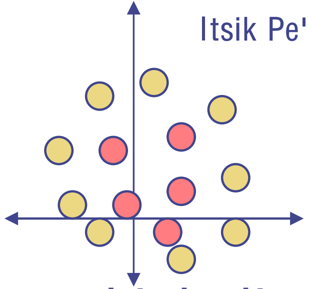
$$\max \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j)$$

$$\text{sign}\left(\sum \alpha_i y_i \phi(x_i)^T \phi(x) + b \right)$$

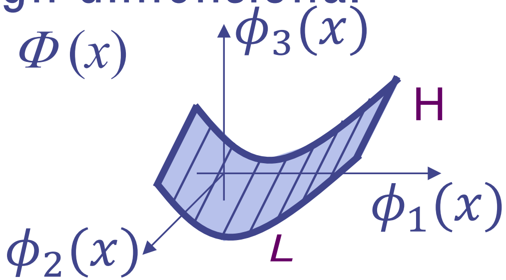
Itsik Pe'er, Columbia University

Nonlinear SVMs

- What if the problem is not linear?



- Map d-dimensional x data from L-space to high dimensional H (Hilbert) feature-space via basis functions $\Phi(x)$
- For example, quadratic classifier:



$$x_i \rightarrow \phi(x_i) \;\; via \;\; \phi(\vec{x}) = \begin{bmatrix} \vec{x} \\ vec(\vec{x}\vec{x}^T) \end{bmatrix}$$

- Call ϕ's feature vectors computed from original x inputs

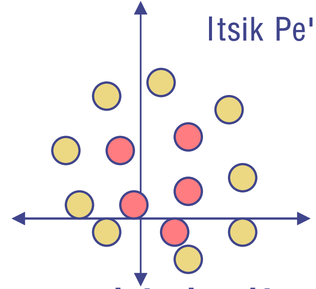
- Dual qp used to be:

$$L_D: \max \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j {x_i}^T x_j \;\; s.t. \alpha_i \geq 0 \,, \sum_i y_i \alpha_i = 0$$

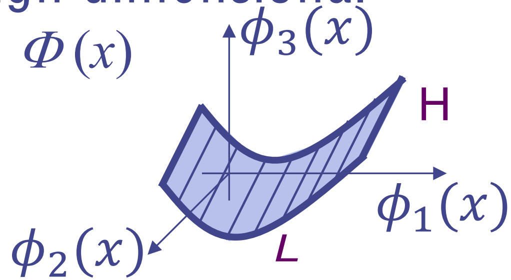
- With linear classifier in original space:

$$f(x) = sign\left(\sum_i \alpha_i y_i {x_i}^T x + b \right)$$

Nonlinear SVMs



- What if the problem is not linear?
- Map d-dimensional x data from L-space to high dimensional H (Hilbert) feature-space via basis functions $\Phi(x)$



- For example, quadratic classifier:

$$x_i \rightarrow \phi(x_i) \;\; via \;\; \phi(\vec{x}) = \begin{bmatrix} \vec{x} \\ vec(\vec{x}\vec{x}^T) \end{bmatrix}$$

- Call ϕ's feature vectors computed from original x inputs
- Replace all x's in the SVM equations with ϕ's
- Now solve the following learning problem:

$$L_D\colon \max \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j \underline{\phi(x_i)^T \phi(x_j)} \quad s.t. \, \alpha_i \geq 0 \, , \sum_i y_i \alpha_i = 0$$

- Which gives a nonlinear classifier in original space:

$$f(x) = sign\left(\sum_i \alpha_i y_i \underline{\phi(x_i)^T \phi(x)} + b \right)$$

Kernels

- One important aspect of SVMs: all math involves only the *inner products* between the ϕ features!

$$f(x) = sign\left(\sum_i \alpha_i y_i \phi(x_i)^T \phi(x_i\ ) + b\right)$$

- Replace all inner products with a general kernel function
- Mercer kernel: accepts 2 inputs and outputs a scalar via:

$$k(x, \tilde{x}) = \langle \phi(x), \phi(\tilde{x}) \rangle = \begin{cases} \phi(x)^T \phi(\tilde{x}) & if\ \phi\ is\ finite \\ \int_t \phi(x,t)\phi(\tilde{x},t)dt & otherwise \end{cases}$$

Kernels

- One important aspect of SVMs: all math involves only the *inner products* between the ϕ features!

$$f(x) = sign\left(\sum_i \alpha_i y_i \phi(x_i)^T \phi(x_i\ ) + b\right)$$

- Replace all inner products with a general kernel function
- Mercer kernel: accepts 2 inputs and outputs a scalar via:

$$k(x,\tilde{x}) = \langle \phi(x), \phi(\tilde{x}) \rangle = \begin{cases} \phi(x)^T \phi(\tilde{x}) & if\ \phi\ is\ finite \\ \int_t \phi(x,t)\phi(\tilde{x},t)dt & otherwise \end{cases}$$

- Mercer's thm: any $k(x,\tilde{x})$ has a $\phi(x)$ if it is "⟨·,·⟩-like"

satisfies Mercer's condition: $\iint g(x)k(x,y)g(y)dxdy \geq 0$

$\forall$ square-integrable g

Kernels

- Mercer kernel: accepts 2 inputs and outputs a scalar via:

$$k(x,\tilde{x}) = \langle \phi(x), \phi(\tilde{x}) \rangle = \begin{cases} \phi(x)^T \phi(\tilde{x}) & if\ \phi\ is\ finite \\ \int_i \phi(x,t)\phi(\tilde{x},t)dt & otherwise \end{cases}$$

- Example: quadratic polynomial $\phi(x) = \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 \end{bmatrix}^T$

$x = [x_1 \ x_2]^T$

$k(x,\tilde{x}) = \phi(x)^T \phi(\tilde{x}) =$

$= x_1^2 \tilde{x}_1^2 + 2 x_1 x_2 \tilde{x}_1 \tilde{x}_2 + x_2^2 \tilde{x}_2^2 =$

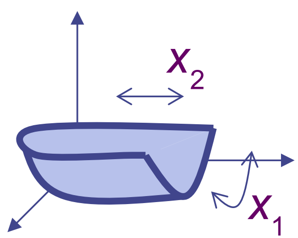
$= (x_1 \tilde{x}_1 + x_2 \tilde{x}_2)^2$

Kernels

- Mercer kernel: accepts 2 inputs and outputs a scalar via:

$$k(x,\tilde{x}) = \langle \phi(x), \phi(\tilde{x}) \rangle = \begin{cases} \phi(x)^T \phi(\tilde{x}) & \text{if } \phi \text{ is finite} \\ \int_t \phi(x,t)\phi(\tilde{x},t)dt & \text{otherwise} \end{cases}$$

- Example: quadratic polynomial $\phi(x) = \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 \end{bmatrix}^T$



$$\begin{aligned} k(x,\tilde{x}) &= \phi(x)^T \phi(\tilde{x}) \\ &= x_1^2\tilde{x}_1^2 + 2x_1x_2\tilde{x}_1\tilde{x}_2 + x_2^2\tilde{x}_2^2 \\ &= (x_1\tilde{x}_1 + x_2\tilde{x}_2)^2 \end{aligned}$$

Kernels

- Sometimes, many $\Phi(x)$ will produce the same $k(x,x')$
- Sometimes $k(x,x')$ computable but features huge or infinite!
- Example: polynomials

If explicit polynomial mapping, feature space $\Phi(x)$ is huge

d-dimensional data, p-th order polynomial, $\dim(H) = \binom{d+p-1}{p}$

images of size 16x16 with $p=4$ have $dim(H)=183$million

Kernels

but can equivalently just use kernel: $k(x,y) = (x^T y)^p$

$k(x,\tilde{x}) =$

$$\sum \left[\;\right] \underbrace{x_1^{n_1} x_2^{n_2} \cdots x_d^{n_d}}_{\phi(x)} \underbrace{\tilde{x}_1^{n_1} \cdots}_{\phi(\tilde{x})}$$

Kernels

but can equivalently just use kernel: $k(x,y) = (x^T y)^p$

$$k(x,\tilde{x}) = (x\tilde{x})^p = \left(\sum_i x_i \tilde{x}_i\right)^p$$

Multinomial Theorem

$$\propto \sum_r \frac{p!}{r_1!\,r_2!\,r_3!\,\ldots(p-\sum_i r_i)!} x_1^{r_1} x_2^{r_2} \ldots x_d^{r_d} \tilde{x}_1^{r_1} \tilde{x}_2^{r_2} \ldots \tilde{x}_d^{r_d}$$

w = weight on term

$$\propto \sum_r \left(\sqrt{w_r}\, x_1^{r_1} x_2^{r_2} \ldots x_d^{r_d}\right)\left(\sqrt{w_r}\, \tilde{x}_1^{r_1} \tilde{x}_2^{r_2} \ldots \tilde{x}_d^{r_d}\right)$$

$$\propto \phi(x)\ \phi(\tilde{x})$$

Equivalent!

$f(x) = \text{sign}\left(\sum \alpha_i y_i k(x_i, x) + b\right)$

$\sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j k(x_i, x_j)$

Kernels

- Replace each $x_i^T x_j \to k(x_i, x_j)$, for example:

P-th Order Polynomial Kernel: $k(x, \tilde{x}) = (x^T \tilde{x} + 1)^P$

RBF Kernel (infinite!): $k(x, \tilde{x}) = \exp\left(-\frac{1}{2\sigma^2} \|x - \tilde{x}\|^2\right)$

Sigmoid (hyperbolic tan) Kernel: $k(x, \tilde{x}) = \tanh(\kappa x^T \tilde{x} - \delta)$

- Using kernels we get generalized inner product SVM:

L_D: $\max \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$ $s.t.$ $\alpha_i \in [0, C], \sum_i \alpha_i y_i = 0$

$$f(x) = sign\left(\sum_i \alpha_i y_i k(x_i, x) + b\right)$$

$\sum \alpha_i y_i k(x_i, x_j) + b > 0 \quad \alpha_j = 0$

Kernels

$\sum \alpha_i y_i k(x_i, x_j) + b < 0 \sim \quad \alpha_j = C$

- Using kernels we get generalized inner product SVM:

L_D: $\max \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$ $s.t.$ $\alpha_i \in [0, C], \sum_i \alpha_i y_i = 0$

support vectors

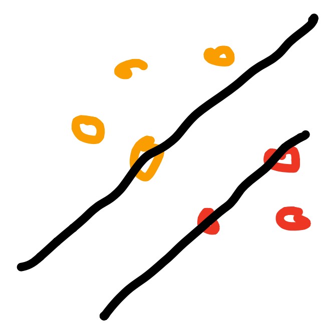
$$f(x) = sign\left(\sum_i \alpha_i y_i k(x_i, x) + b\right)$$

$\sum \alpha_i y_i k(x_i, x_j) + b = 0 \qquad 0 < \alpha_j < C$

- Still qp solver, just use Gram matrix K (positive definite)

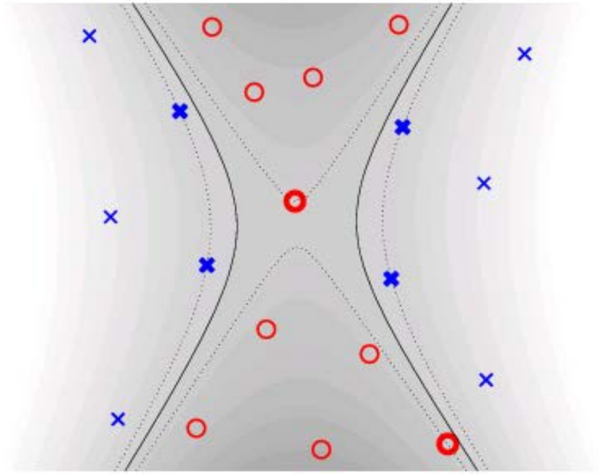
$$K_{i,j} = k(x_i, x_j)$$

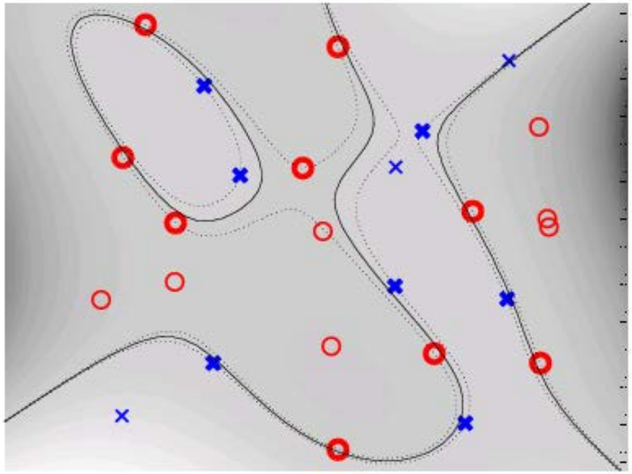
$$K = \begin{bmatrix} k(x_1,x_1) & k(x_2,x_1) & k(x_3,x_1) \\ k(x_1,x_2) & k(x_2,x_2) & k(x_3,x_2) \\ k(x_1,x_3) & k(x_2,x_3) & k(x_3,x_3) \end{bmatrix}$$



Kernelized SVMs

- Polynomial kernel:
  $k(x, \tilde{x}) = (x^T \tilde{x} + 1)^P$
- Radial basis function kernel:
  $k(x, \tilde{x}) = \exp\left(-\frac{1}{2\sigma^2} \|x - \tilde{x}\|^2\right)$



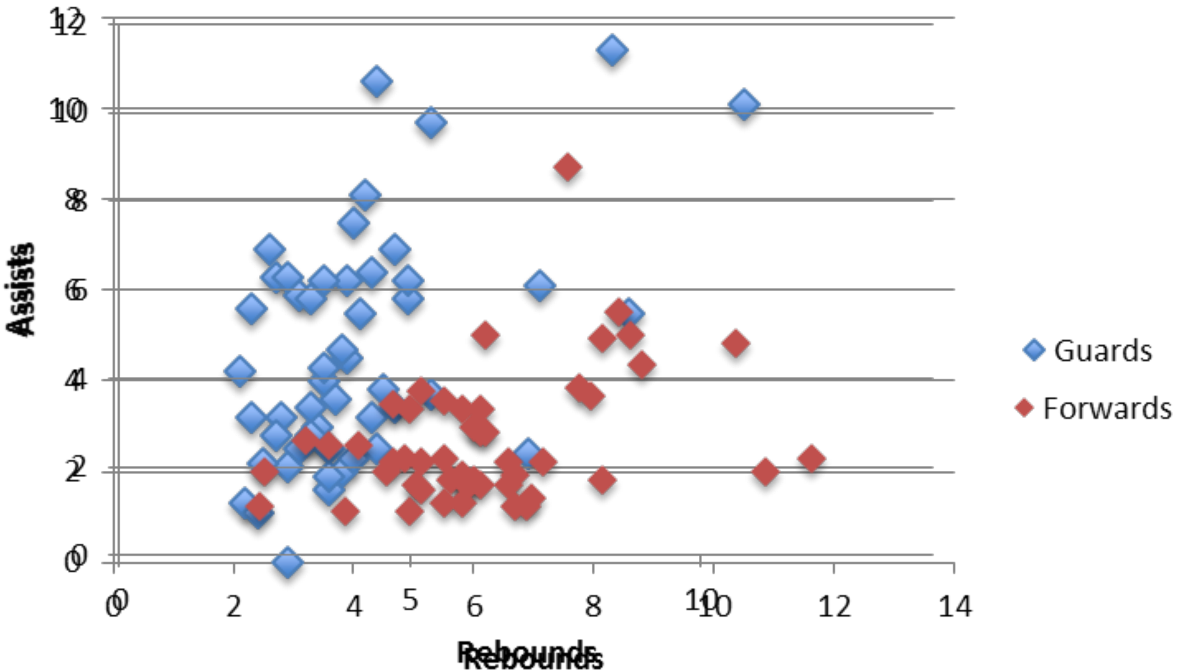


- Edit distance: no explicit feature set

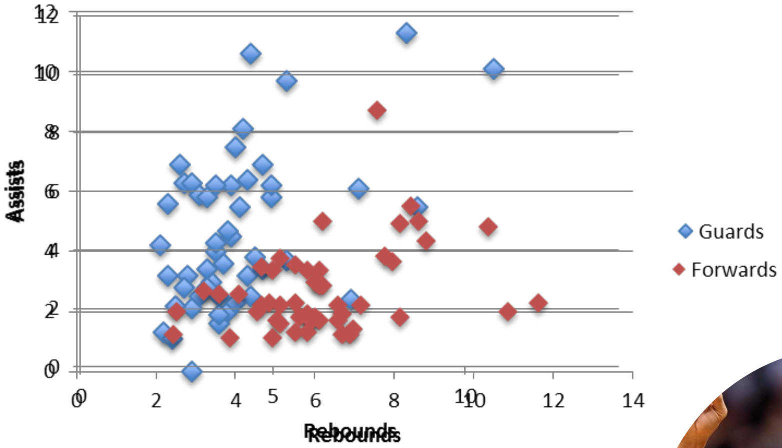
Summary

- Kernels extend SVM: linear $\rightarrow$ nonlinear
- Other ways?

Example: Classifying Players



Example: Classifying Players

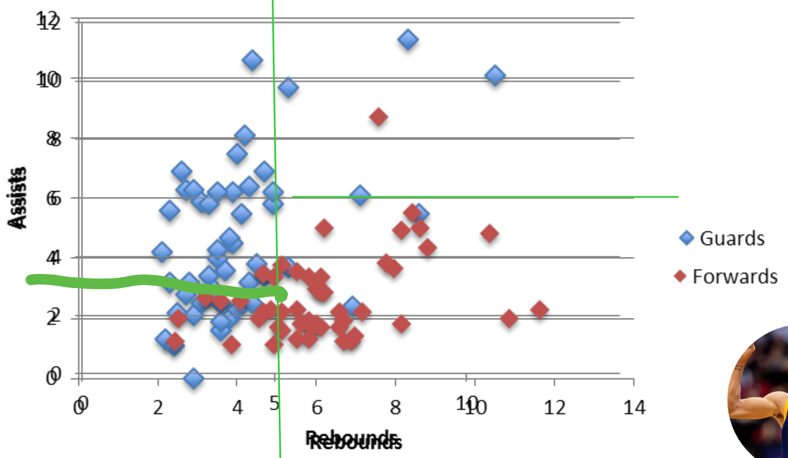


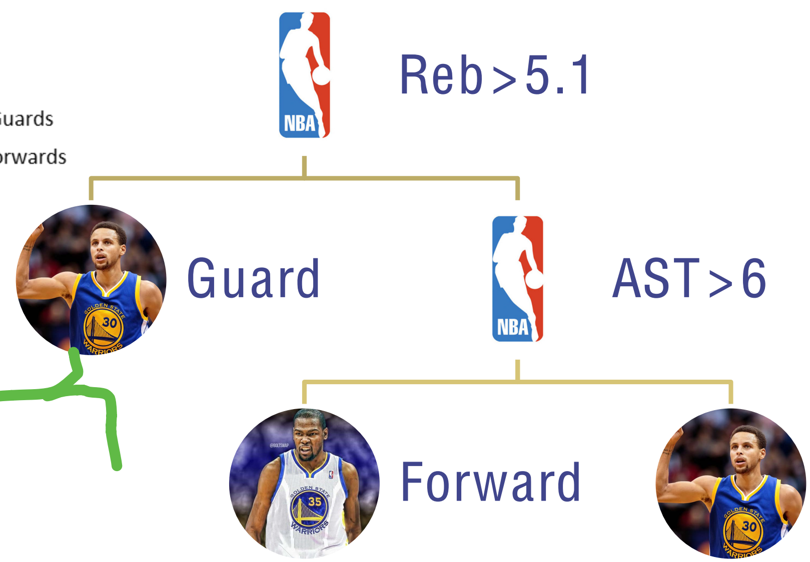
- Axis-parallel criteria: Easy to find
- Start with default:

Guard

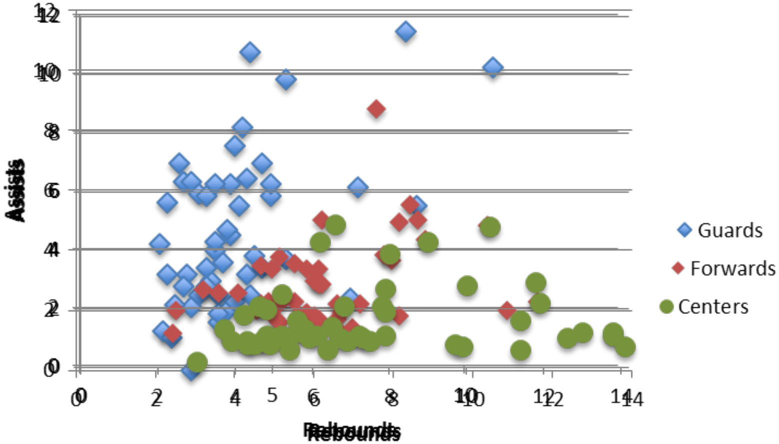
Example: Classifying Players

- Leaf decides based on plurality

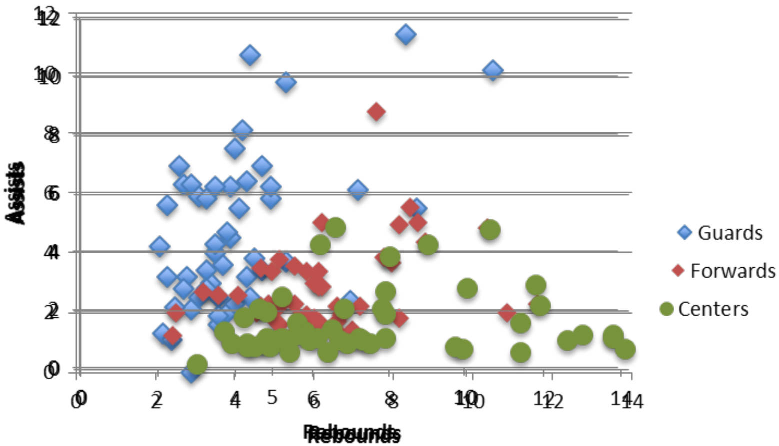




Multiple Classes



Multiple Classes



Greedy algorithm:

- Init: empty tree
- While (!stopping())
  - Choose leaf
  - Split leaf

uncertainty