Machine Learning

4771

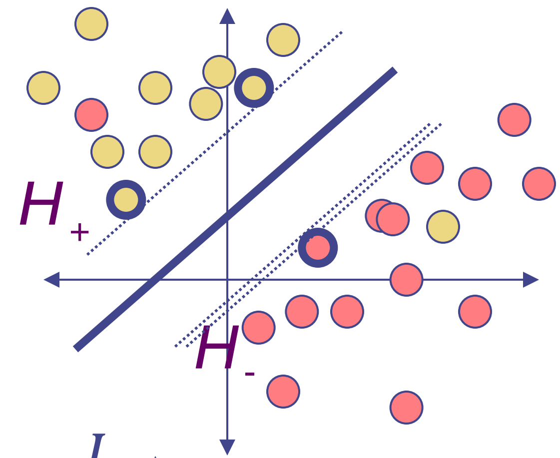
Instructor: Itsik Pe’er

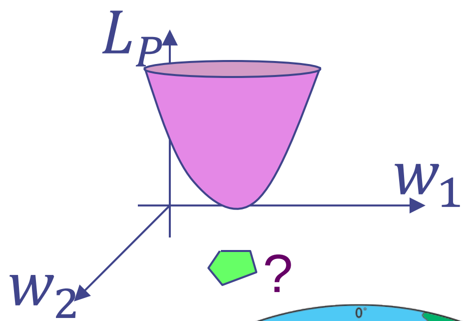
Administration/HW/quiz

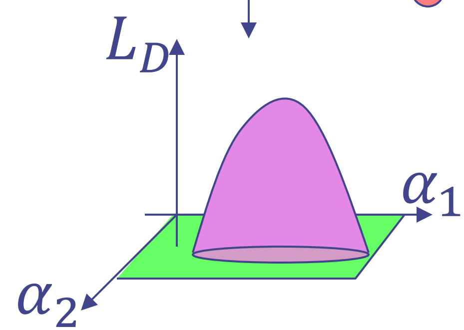
- Legibility
- likelihood: Prob(entire data)
  loss: log-likelihood contribution by datapoint
- Limits of distributions
- OLS: Poly, N>D
- EAP(Bernouli)

Reminder: SVM

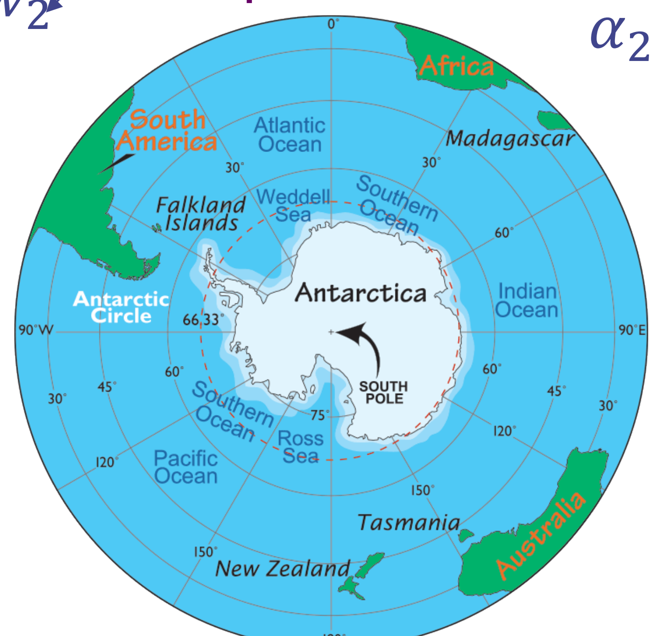
- Non-separable





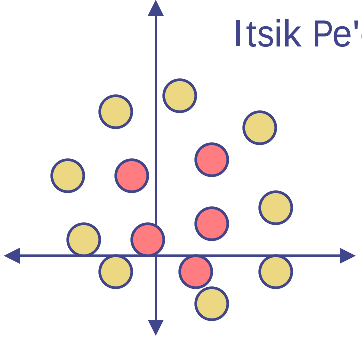


- Non linear



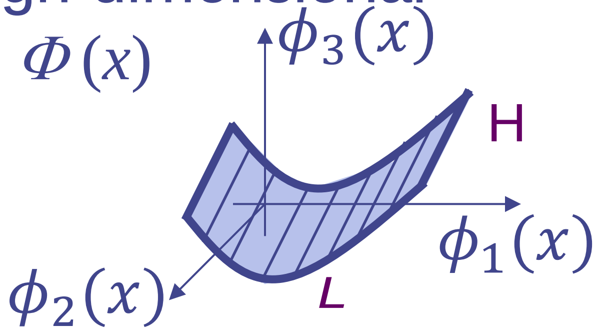
Itsik Pe'er, Columbia University

Nonlinear SVMs



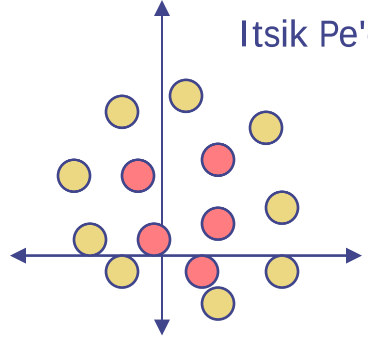
- What if the problem is not linear?
- We can use our old trick...
- Map d-dimensional x data from L-space to high dimensional H (Hilbert) feature-space via basis functions $\Phi(x)$
- For example, quadratic classifier:

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

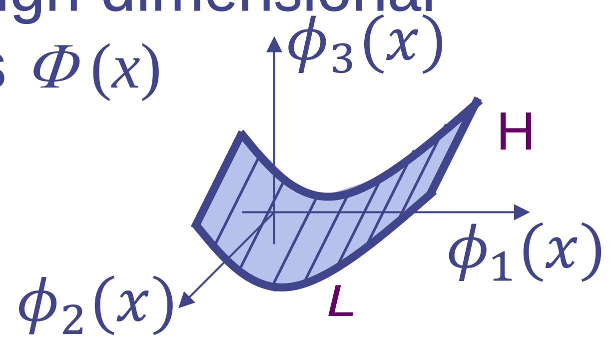


- Call ϕ's feature vectors computed from original x inputs

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:

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

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

- Dual qp used to be:

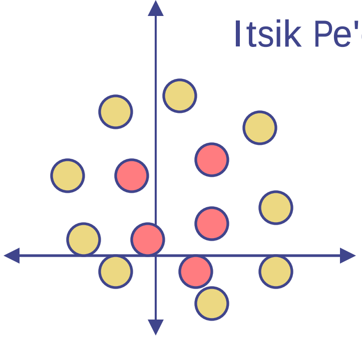
$$L_D: \max \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i{}^T x_j \;\; s.t. \alpha_i \geq 0 \, , \sum_i y_i \alpha_i = 0$$

- With linear classifier in original space:

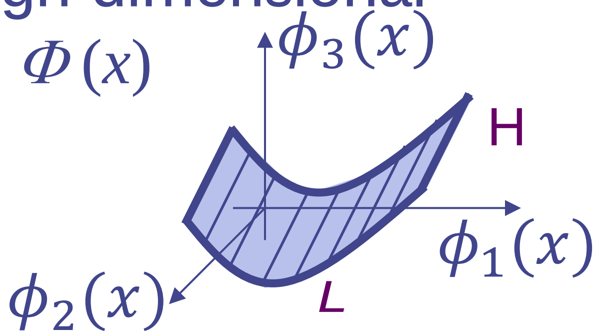
$$f(x) = sign\left(\sum_i \alpha_i y_i x_i{}^T x + b \right)$$

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:

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

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

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

- Which gives a nonlinear classifier in original space:

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

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:

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

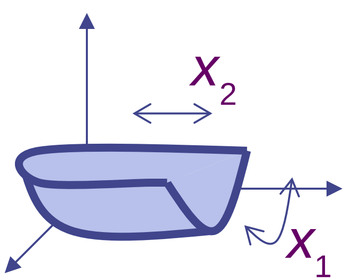
- Example: quadratic polynomial $\phi(x) = \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 \end{bmatrix}^T$

Kernels

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

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

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



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

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 $dim(H)$=183million

Kernels

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

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

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

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

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!

Kernels

- Replace each $x_i^T x_j \to k\left(x_i, x_j\right)$, 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: $\max \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j k\left(x_i, x_j\right)$ $s.t.$ $\alpha_i \in [0, C], \sum_i \alpha_i y_i = 0$

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

Kernels

- Using kernels we get generalized inner product SVM:

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

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

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

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

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

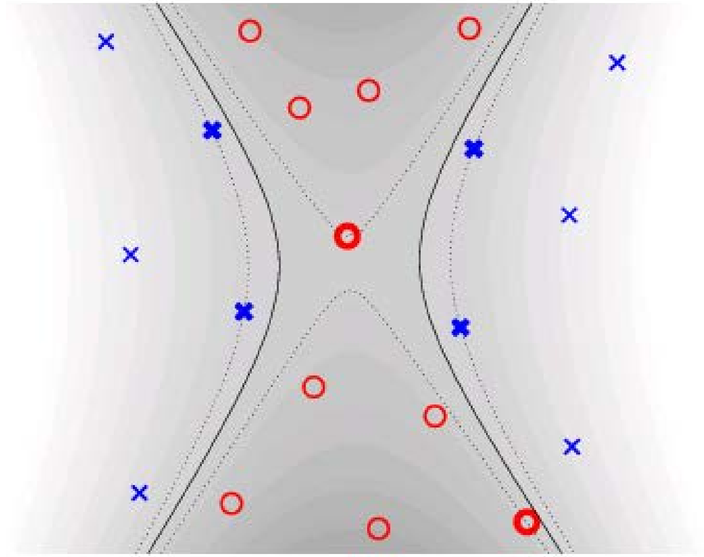
Kernelized SVMs

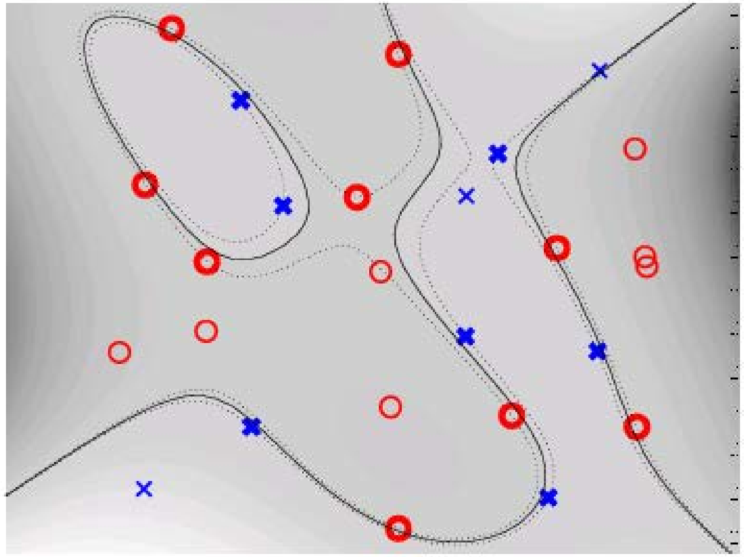
- Polynomial kernel:

$$k(x,\tilde{x}) = (x^T\tilde{x}+1)^P$$

- Radial basis function kernel:

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



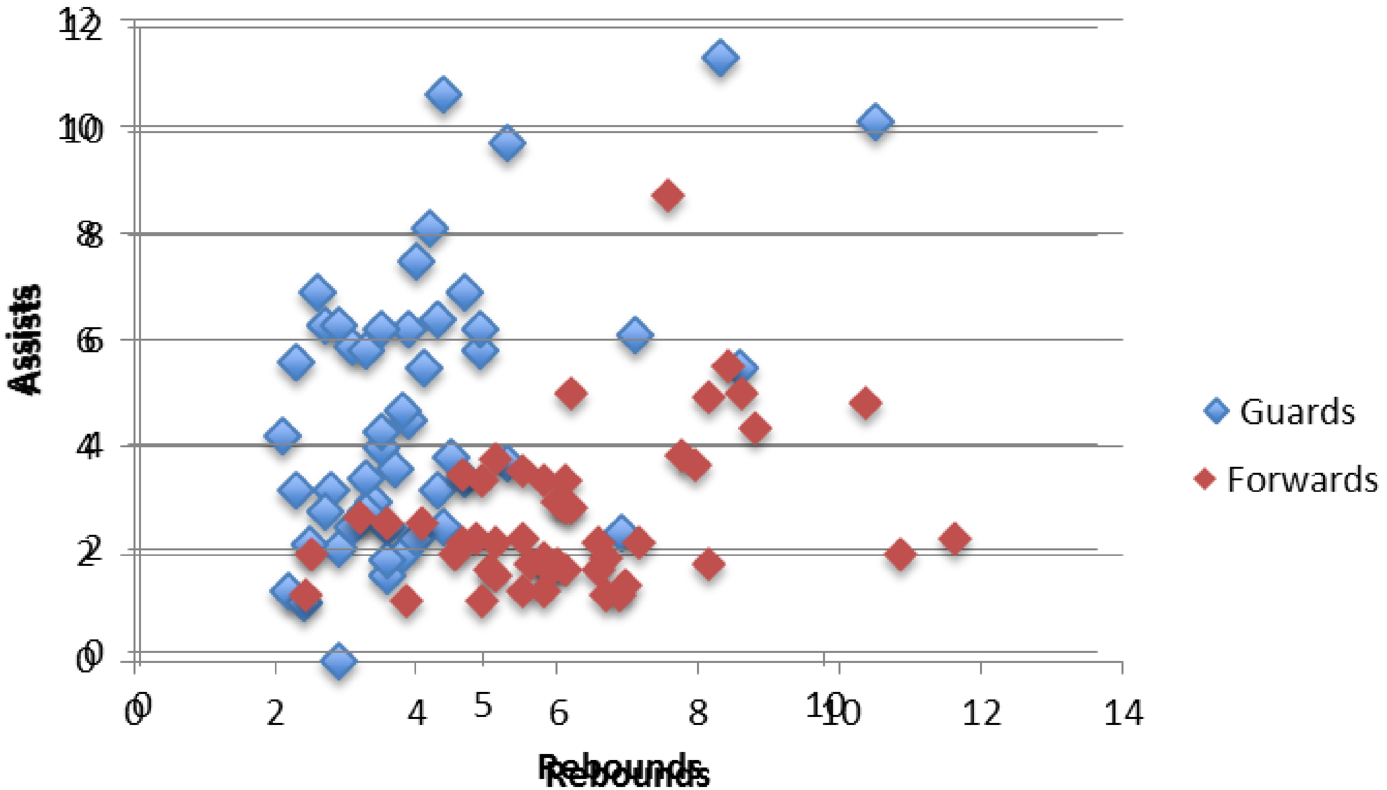


- Edit distance: no explicit feature set

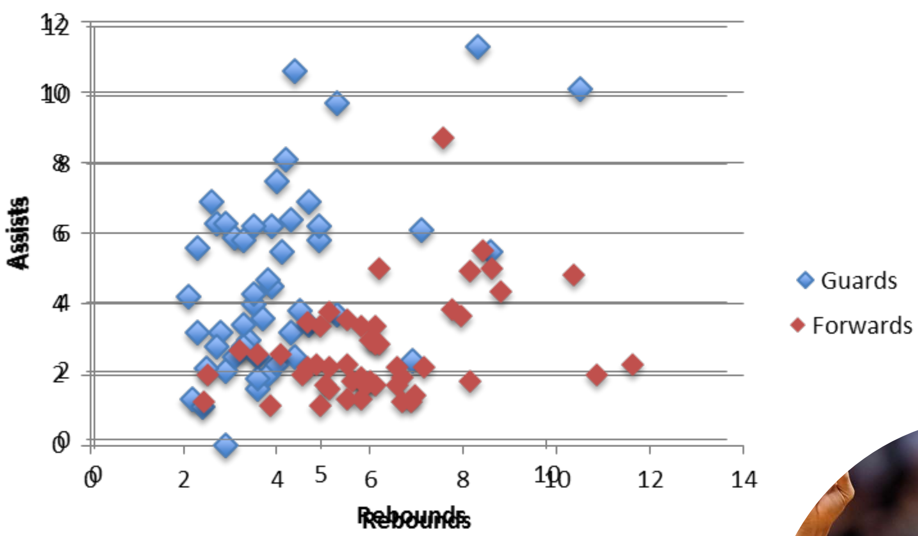
Summary

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

Example: Classifying Players



Example: Classifying Players

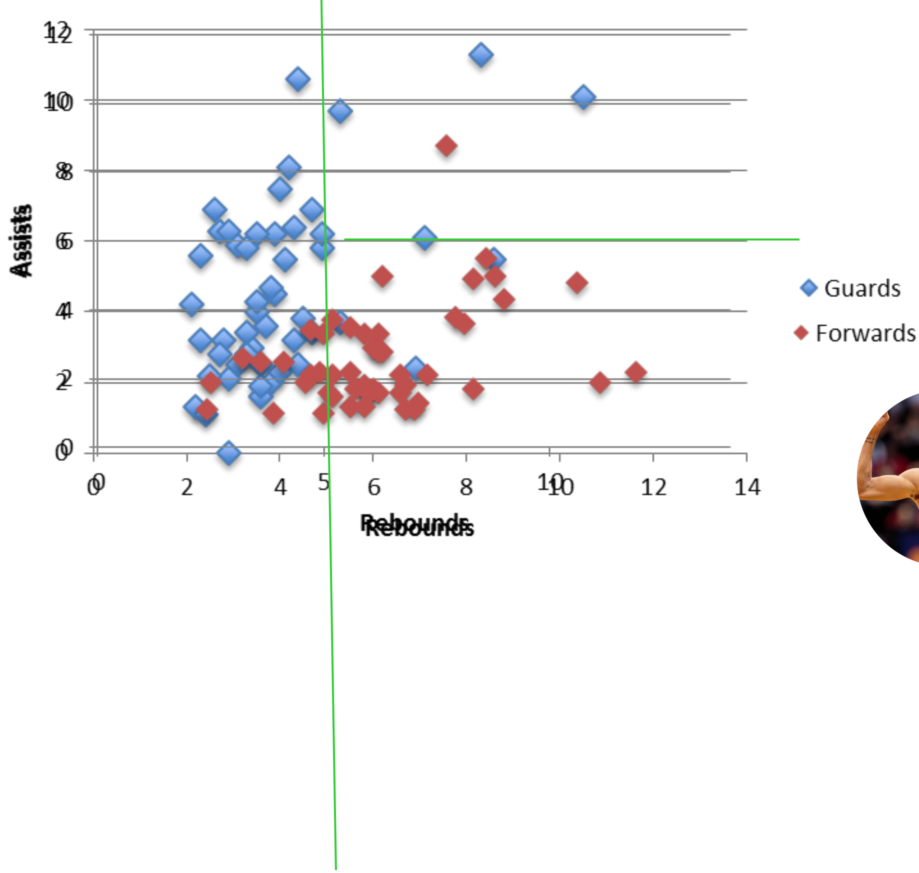


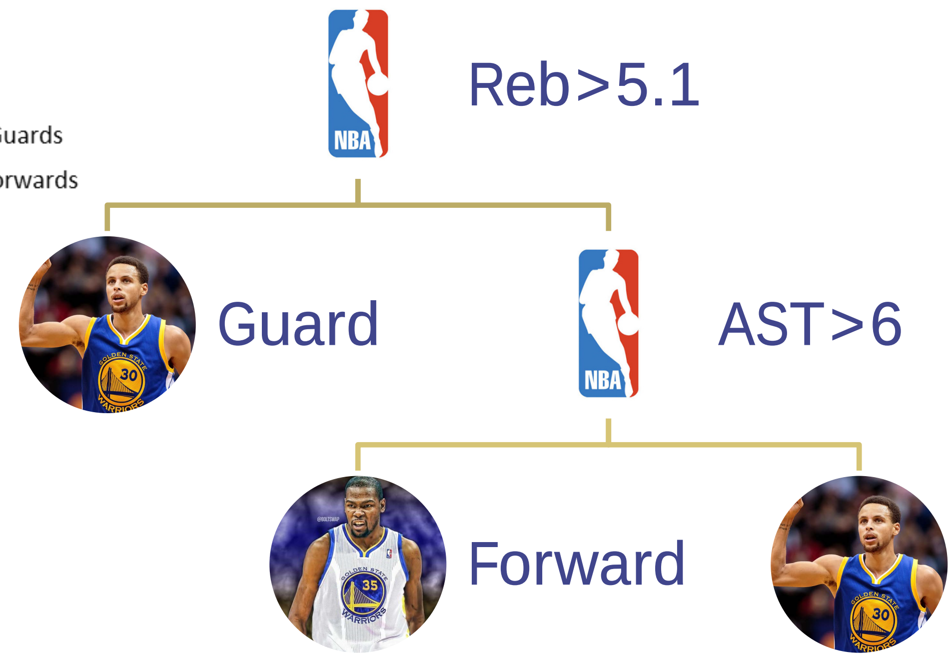
- Axis-parallel criteria: Easy to find
- Start with default:

Guard

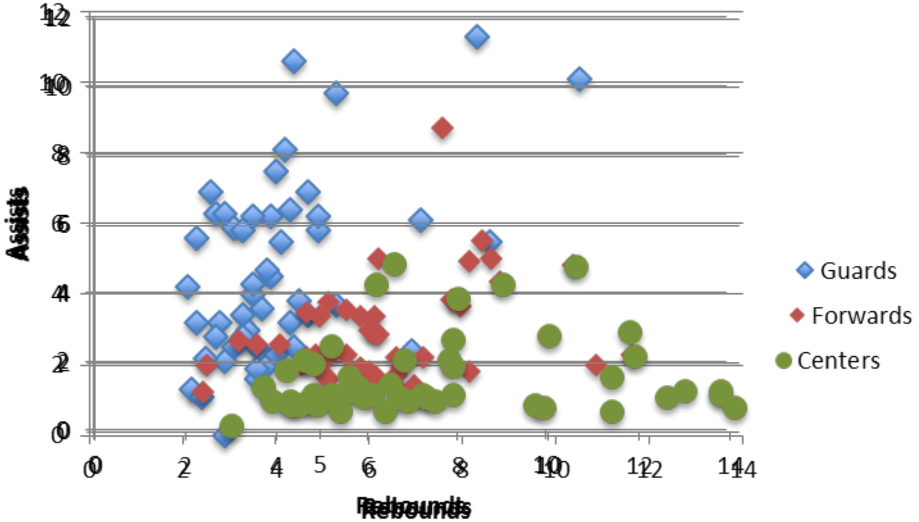
Example: Classifying Players

- Leaf decides based on plurality

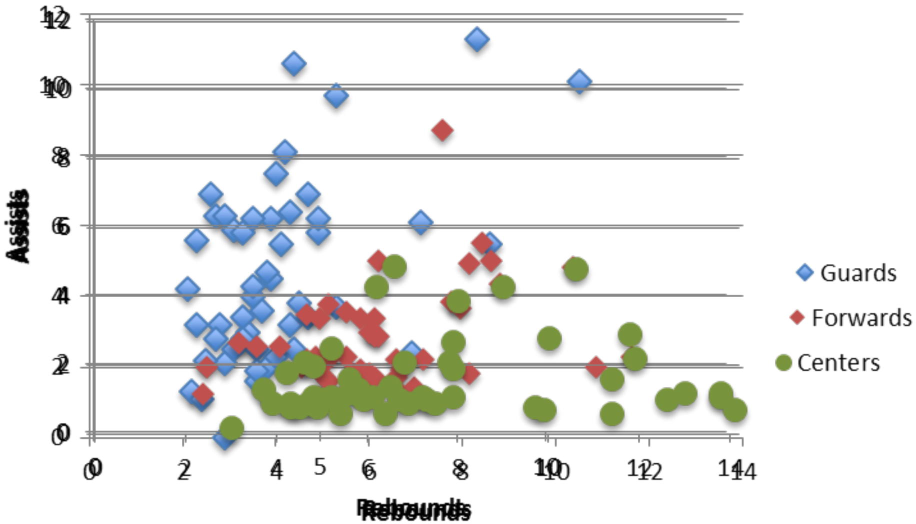




Multiple Classes



Multiple Classes



Greedy algorithm:

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

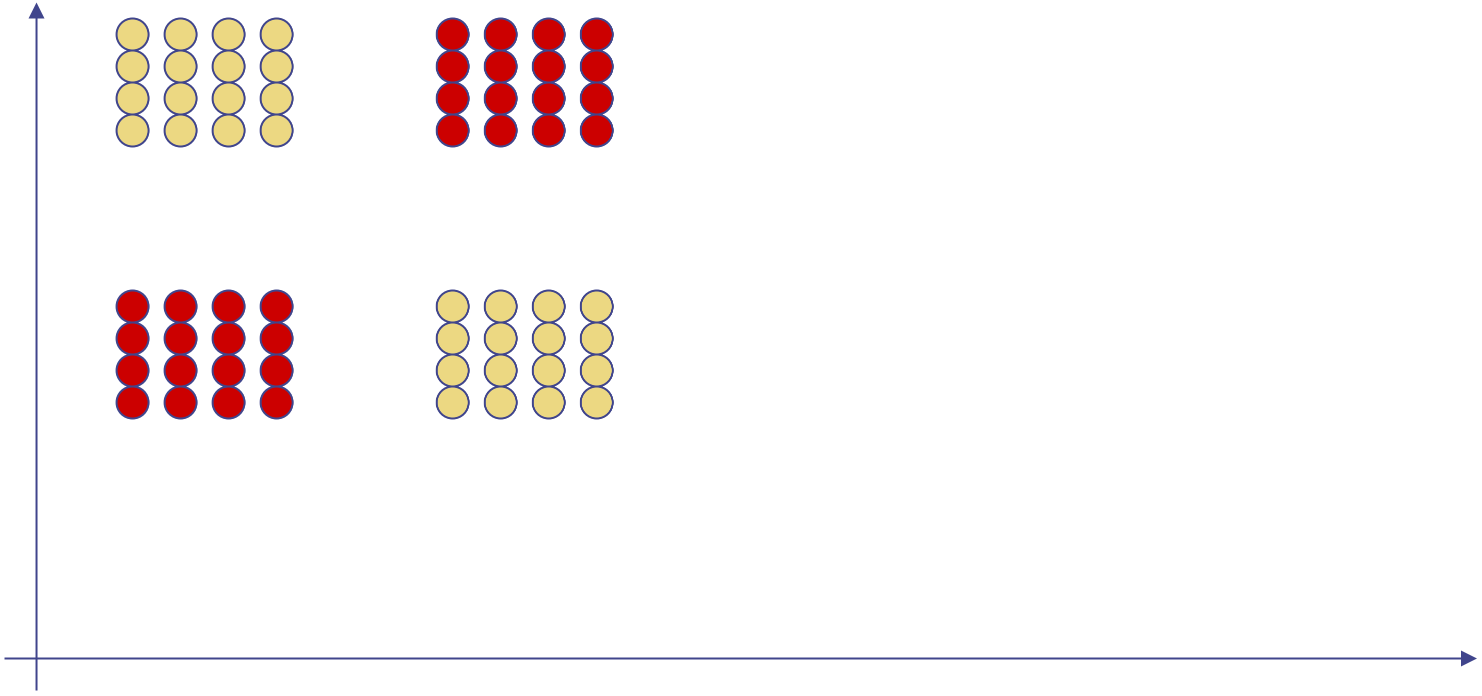
Objective: Certainty

Choose leaf and split to minimize a measure of uncertainty:

- Classification error: $1 - \max p_i$
- Gini index: $1 - \sum p_i^2$
- Entropy: $-\sum p_i \log p_i$

Stopping Criteria

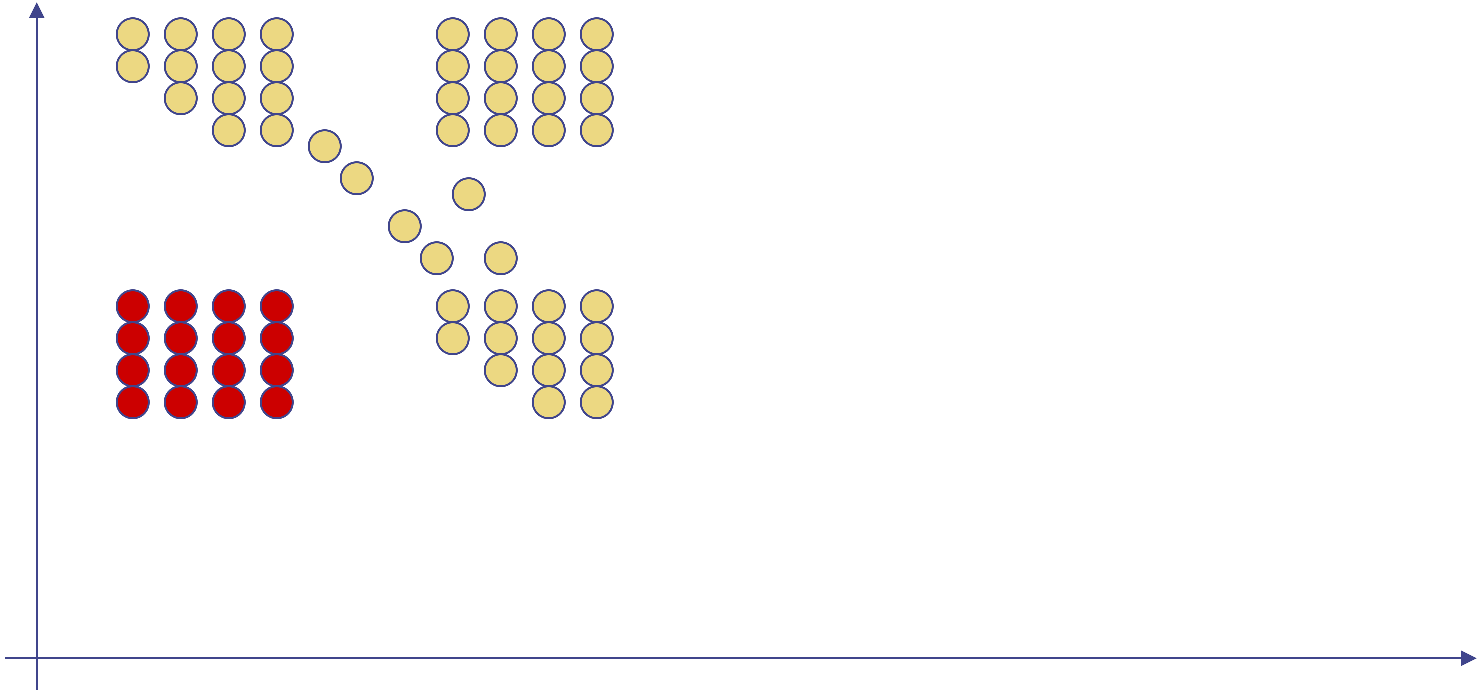
Example:
No split reduces uncertainty



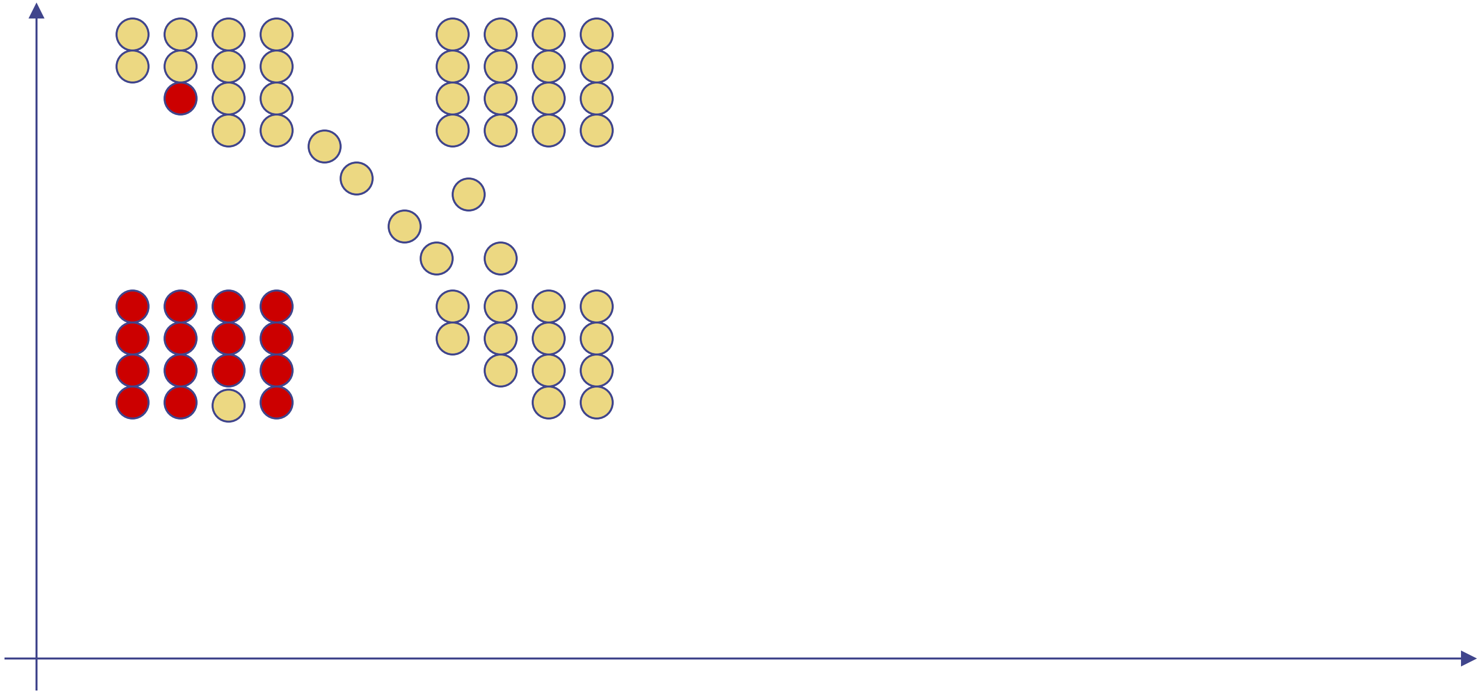
Stopping Criteria

- No improvement?
- Certain tree size
- When leaves are pure

Example:
Clear split if clean data



Example:
Overfit if noisy data



Stopping Criteria

- No improvement?
- Certain tree size
- When leaves are pure
  - Overfitting. Requires pruning
  - Address by validation set.
  - Prune the pure-training tree

Summary

- Decision trees form greedily