Feature Spaces

- Features can be much more complex

Feature Spaces

- Features can be much more complex
- Drawn from bigger discrete set

Feature Spaces

- Features can be much more complex
- Drawn from bigger discrete set
  - If set is unordered (4 different makes of cars, for example), use binary attributes to encode the values (1000, 0100, 0010, 0001)

Feature Spaces

- Features can be much more complex

- Drawn from bigger discrete set
  - If set is unordered (4 different makes of cars, for example), use binary attributes to encode the values (1000, 0100, 0010, 0001)
  - If set is ordered, treat as real-valued

Feature Spaces

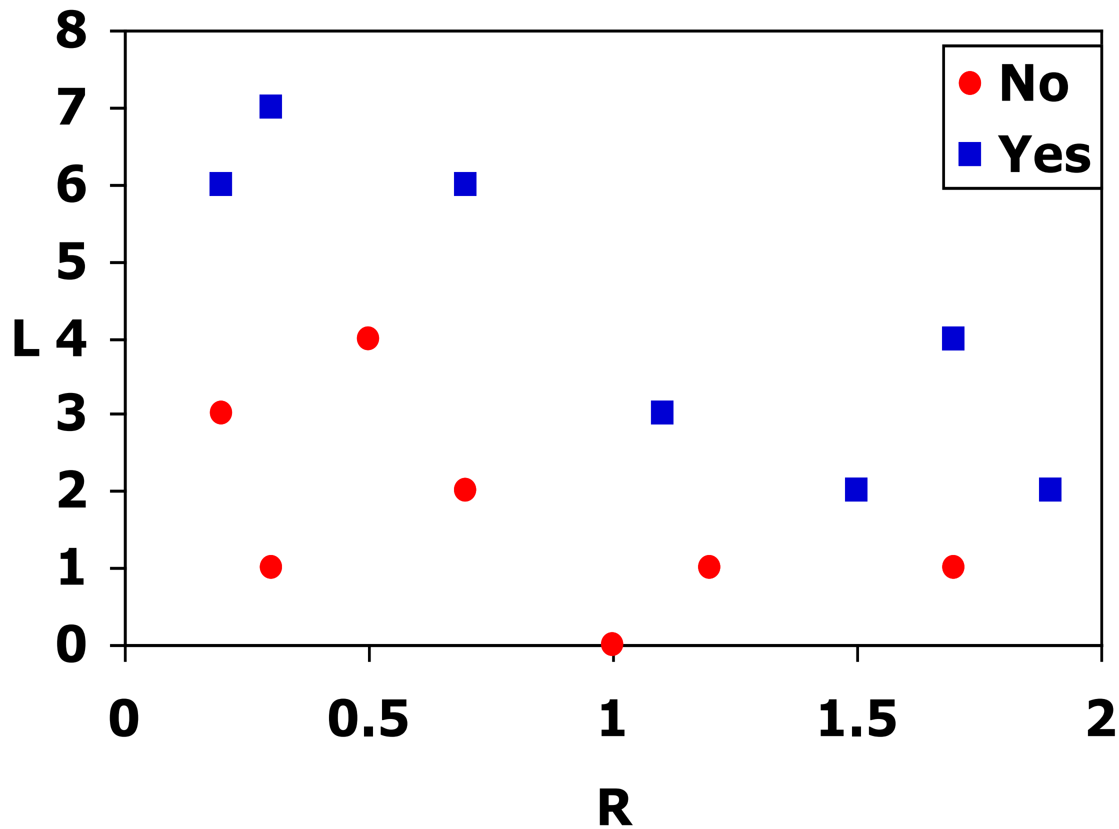
- Features can be much more complex

- Drawn from bigger discrete set
  - If set is unordered (4 different makes of cars, for example), use binary attributes to encode the values (1000, 0100, 0010, 0001)
  - If set is ordered, treat as real-valued

- Real-valued: bias that inputs whose features have "nearby" values ought to have "nearby" outputs

Predicting Bankruptcy

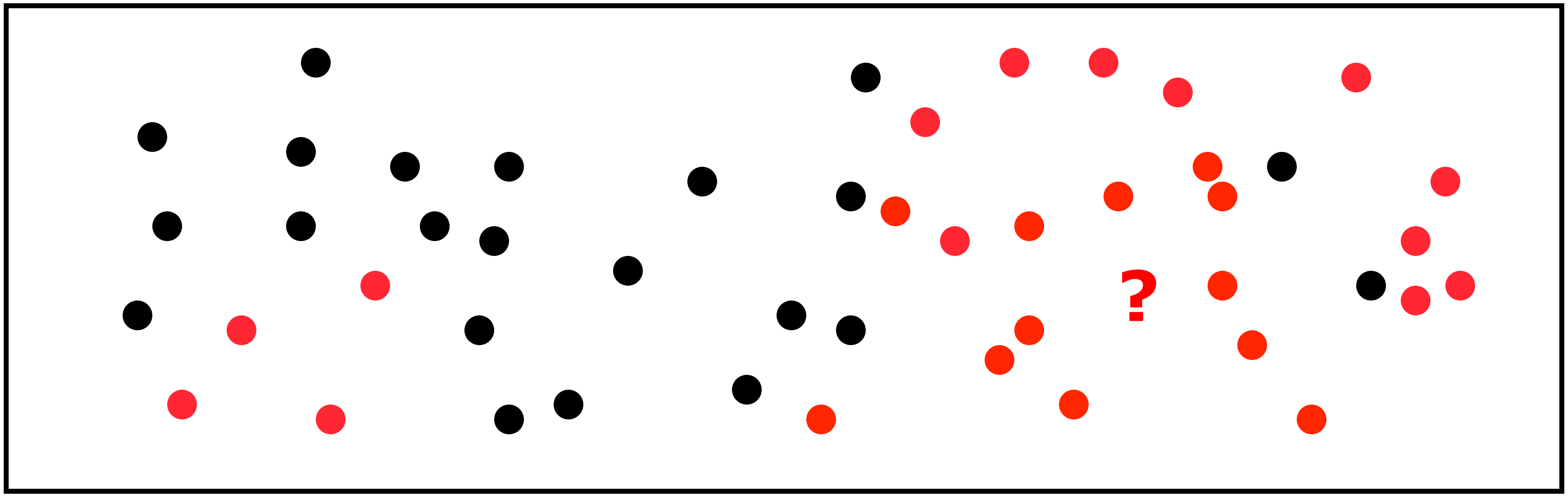
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1.0	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



L: #late payments / year
R: expenses / income

Love thy Nearest Neighbor

- Remember all your data
- When someone asks a question,
  - find the nearest old data point
  - return the answer associated with it



What do we mean by "Nearest"?

- Need a distance function on inputs
- Typically use Euclidean distance (length of a straight line between the points)

$$D(x^i, x^k) = \sqrt{\sum_j (x_j^i - x_j^k)^2}$$

What do we mean by “Nearest”?

- Need a distance function on inputs
- Typically use Euclidean distance (length of a straight line between the points)

$$D(x^i, x^k) = \sqrt{\sum_j (x_j^i - x_j^k)^2}$$

- Distance between character strings might be number of edits required to turn one into the other

Scaling

- What if we're trying to predict a car's gas mileage?
  - f_1 = weight in pounds
  - f_2 = number of cylinders

Scaling

- What if we're trying to predict a car's gas mileage?
  - f_1 = weight in pounds
  - f_2 = number of cylinders
- Any effect of f_2 will be completely lost because of the relative scales

Scaling

- What if we're trying to predict a car's gas mileage?
  - f_1 = weight in pounds
  - f_2 = number of cylinders
- Any effect of f_2 will be completely lost because of the relative scales
- So, re-scale the inputs

Scaling

- What if we're trying to predict a car's gas mileage?
  - f_1 = weight in pounds
  - f_2 = number of cylinders
- Any effect of f_2 will be completely lost because of the relative scales
- So, re-scale the inputs to have mean 0 and variance 1:

$$x' = \frac{x - \overline{x}}{\sigma_x}$$

average

standard deviation

Scaling

- What if we're trying to predict a car's gas mileage?
  - f_1 = weight in pounds
  - f_2 = number of cylinders
- Any effect of f_2 will be completely lost because of the relative scales
- So, re-scale the inputs to have mean 0 and variance 1:

$$x' = \frac{x - \underbrace{\bar{x}}_{\text{average}}}{\underbrace{\sigma_x}_{\text{standard deviation}}}$$

- Or, build knowledge in by scaling features differently

Scaling

- What if we're trying to predict a car's gas mileage?
  - f_1 = weight in pounds
  - f_2 = number of cylinders
- Any effect of f_2 will be completely lost because of the relative scales
- So, re-scale the inputs to have mean 0 and variance 1:

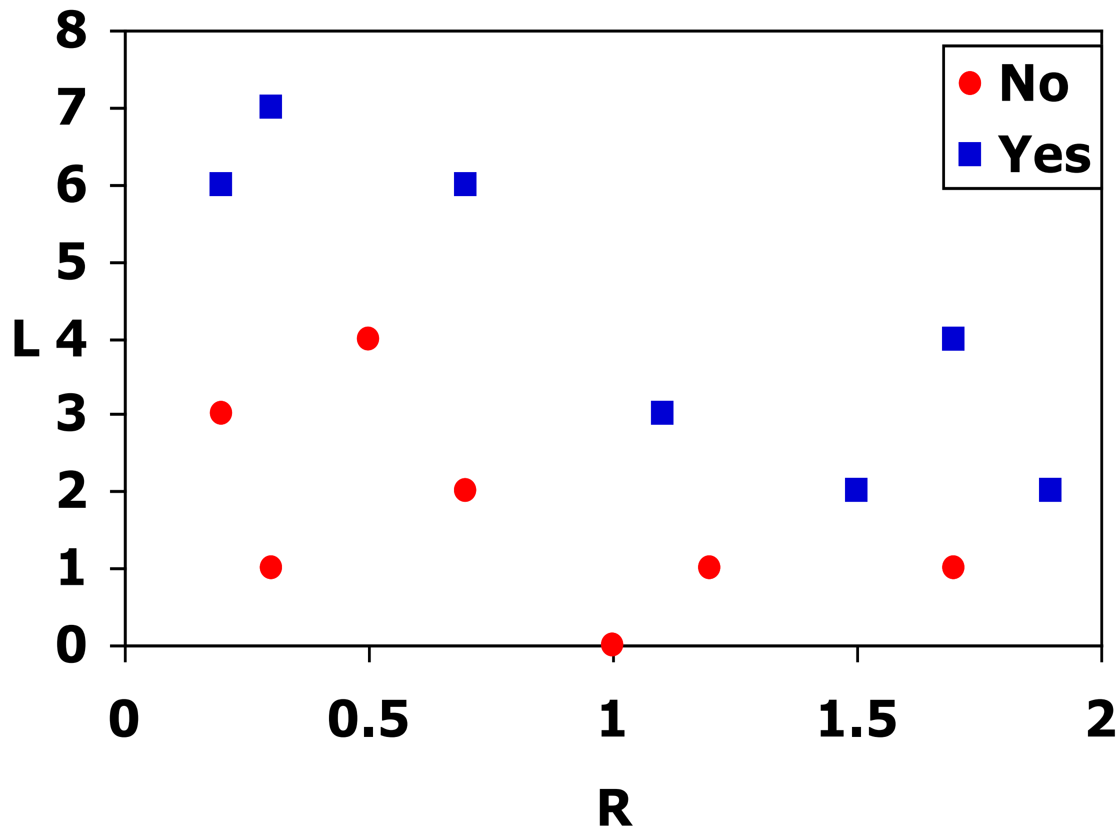
$$x' = \frac{x - \overline{x}}{\sigma_x}$$

average

standard deviation

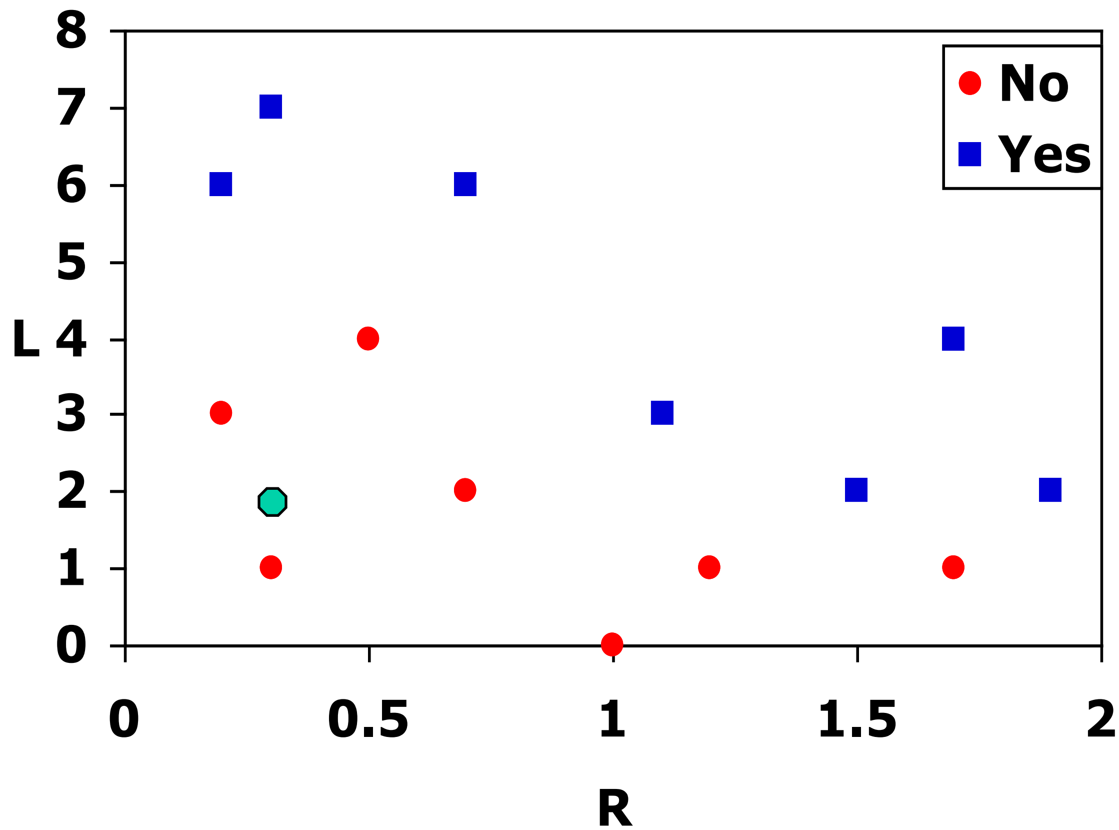
- Or, build knowledge in by scaling features differently
- Or use cross-validation to choose scales

Predicting Bankruptcy



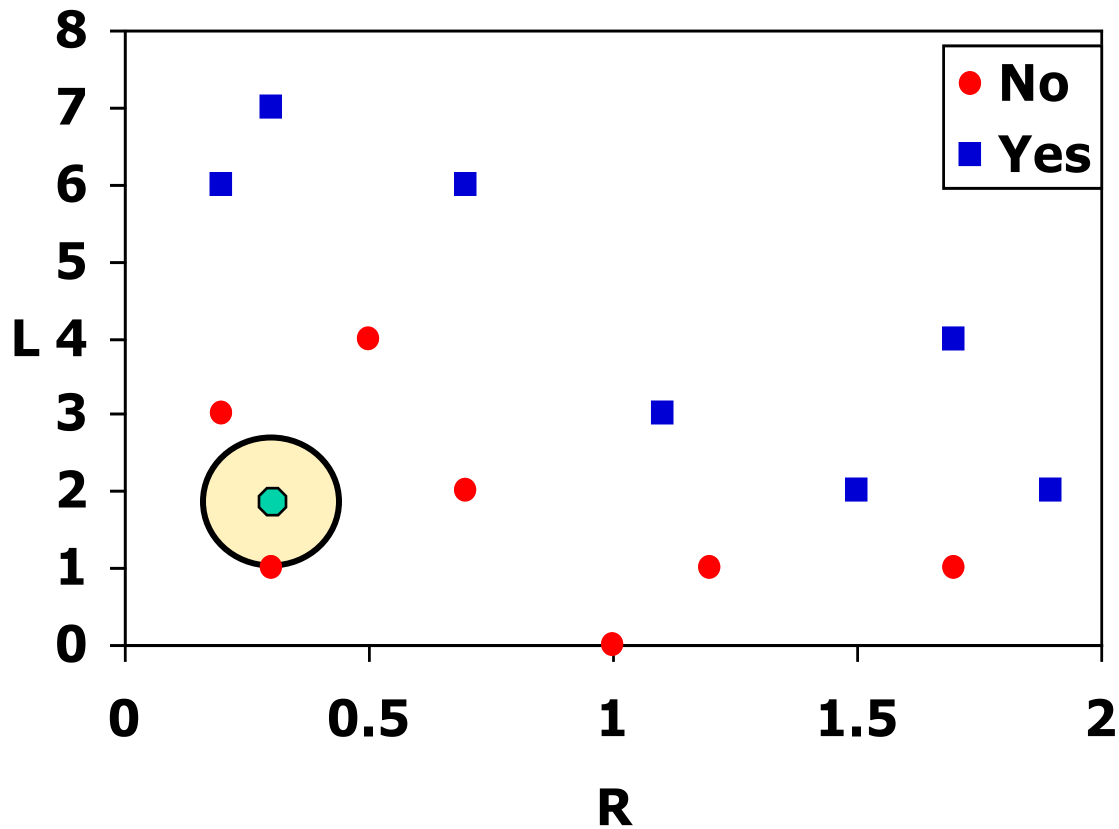
$$D(x^i, x^k) = \sqrt{\sum_j (L^i - L^k)^2 + (5R^i - 5R^k)^2}$$

Predicting Bankruptcy



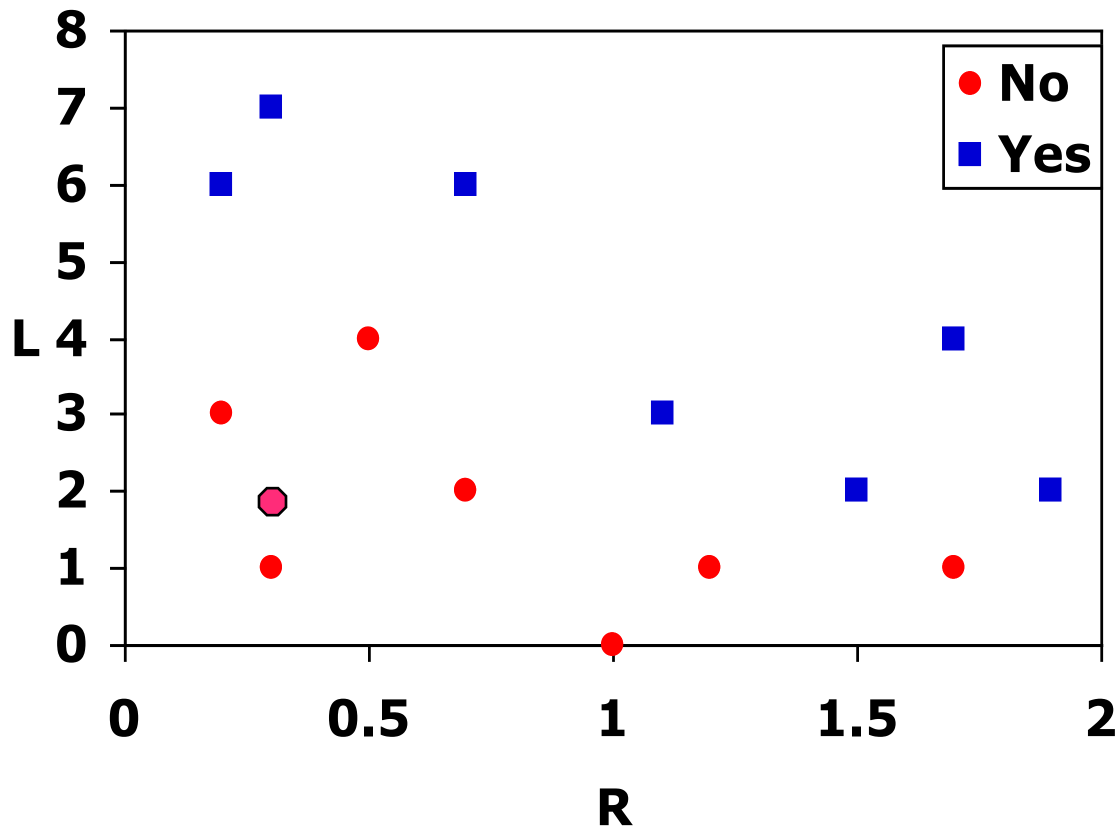
$$D(x^i, x^k) = \sqrt{\sum_j (L^i - L^k)^2 + (5R^i - 5R^k)^2}$$

Predicting Bankruptcy



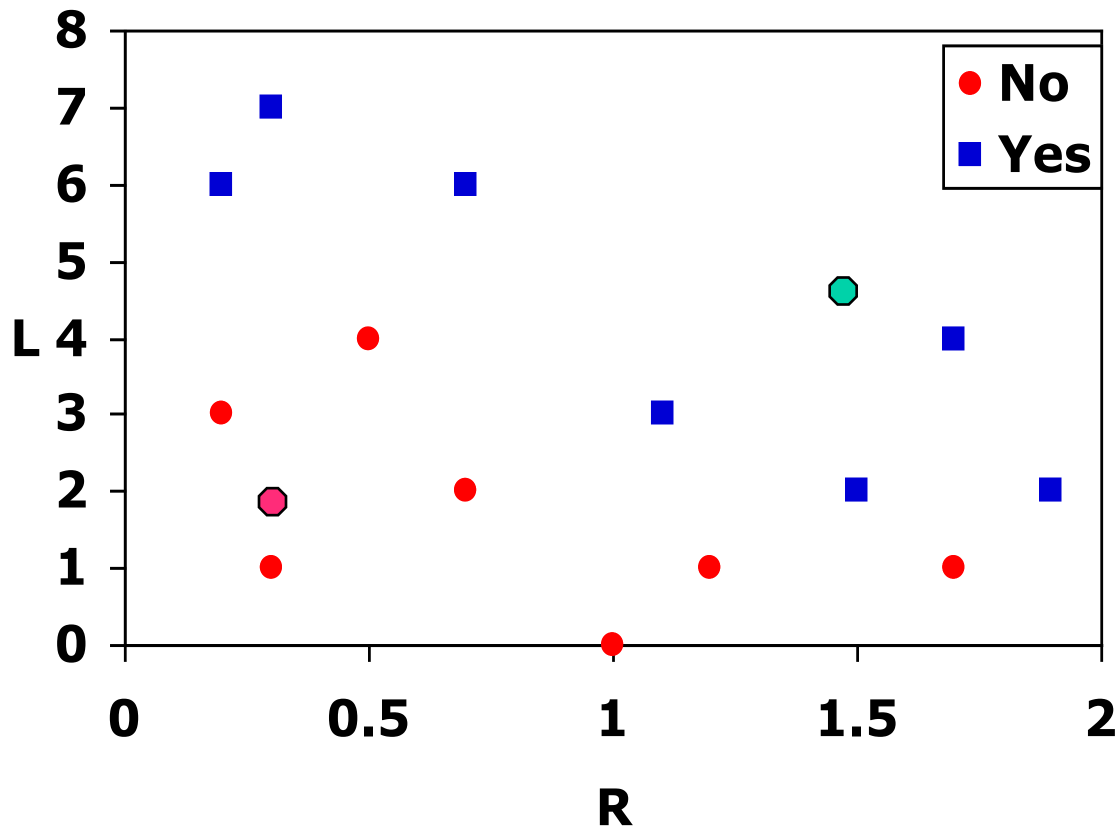
$$D(x^i, x^k) = \sqrt{\sum_j (L^i - L^k)^2 + (5R^i - 5R^k)^2}$$

Predicting Bankruptcy



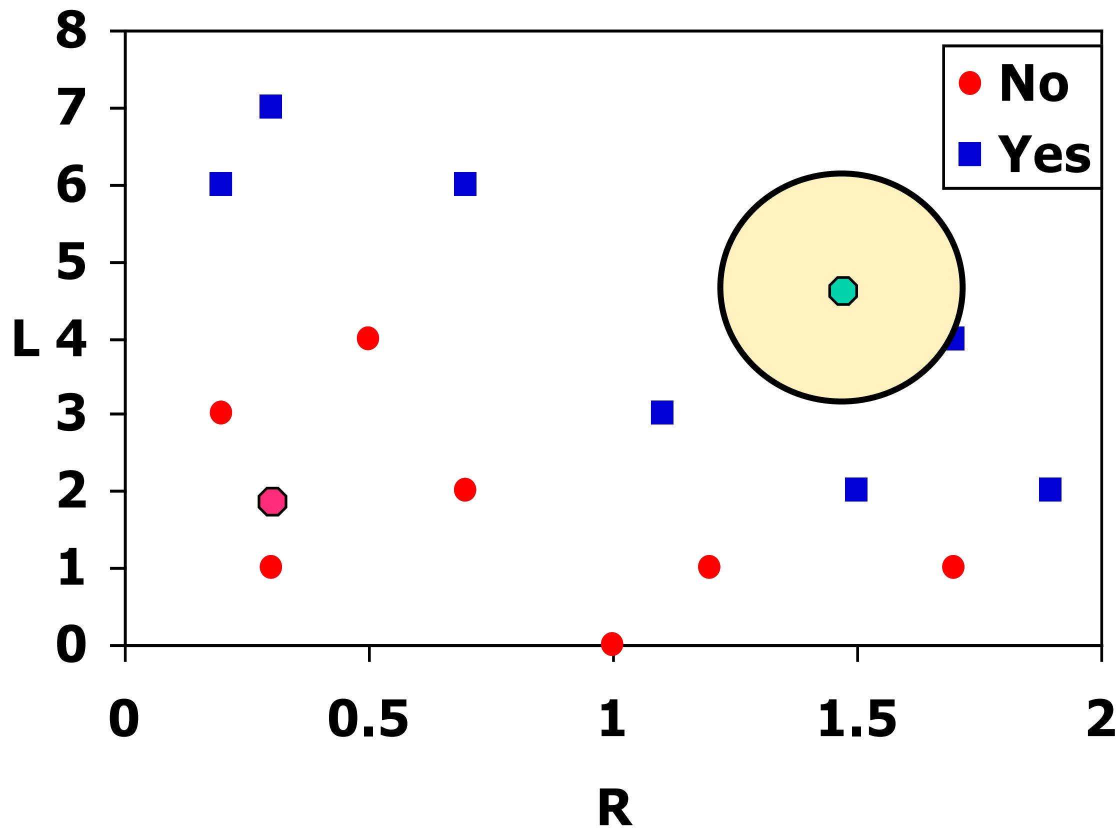
$$D(x^i, x^k) = \sqrt{\sum_j (L^i - L^k)^2 + (5R^i - 5R^k)^2}$$

Predicting Bankruptcy



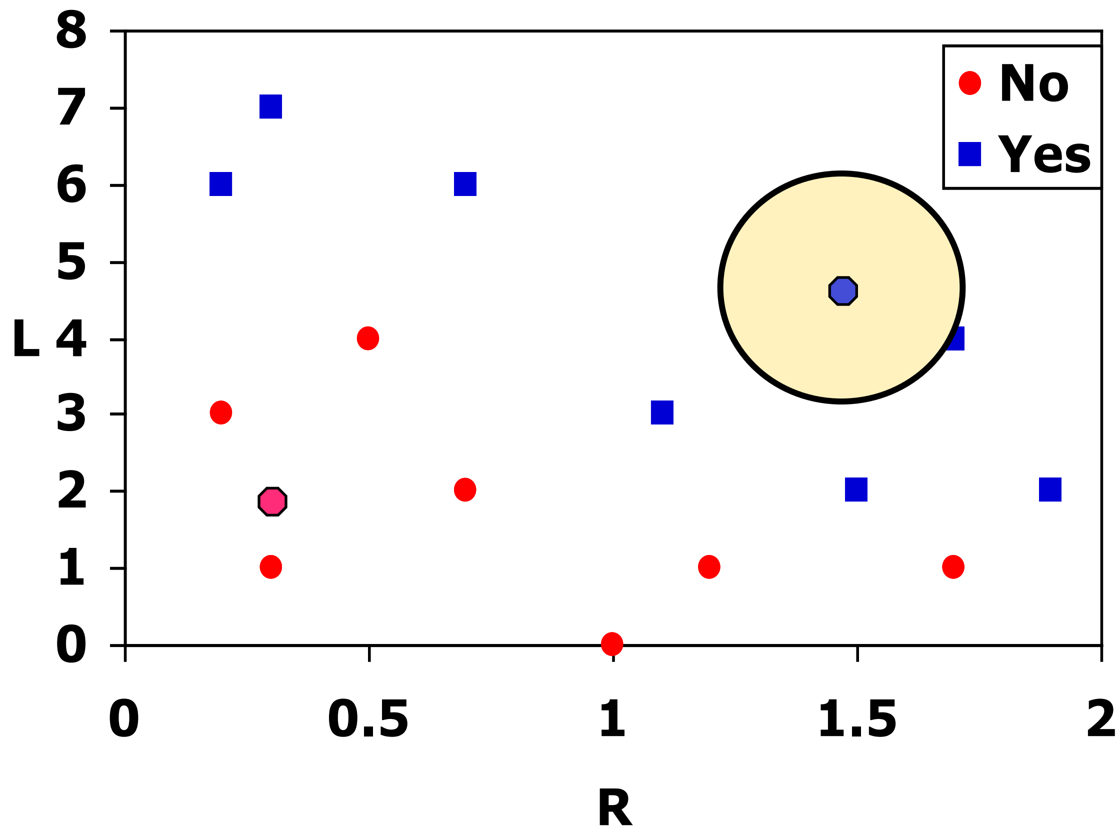
$$D(x^i, x^k) = \sqrt{\sum_j (L^i - L^k)^2 + (5R^i - 5R^k)^2}$$

Predicting Bankruptcy



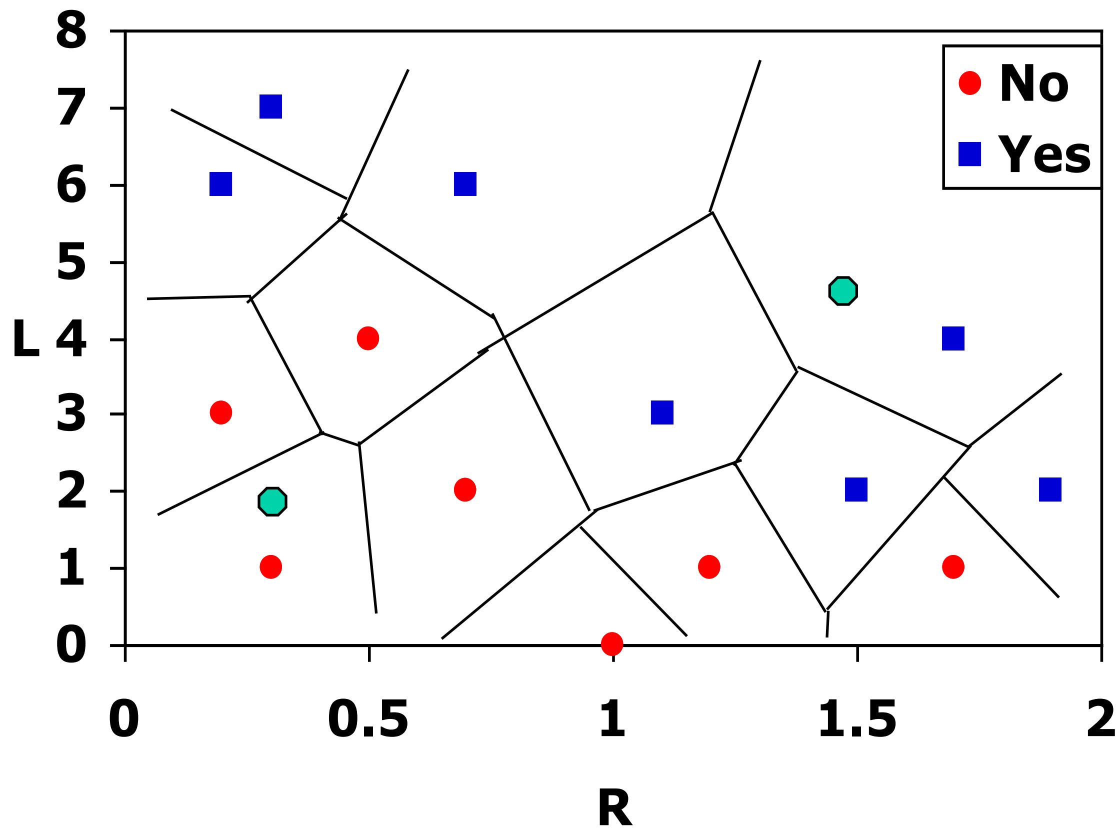
$$D(x^i, x^k) = \sqrt{\sum_j (L^i - L^k)^2 + (5R^i - 5R^k)^2}$$

Predicting Bankruptcy



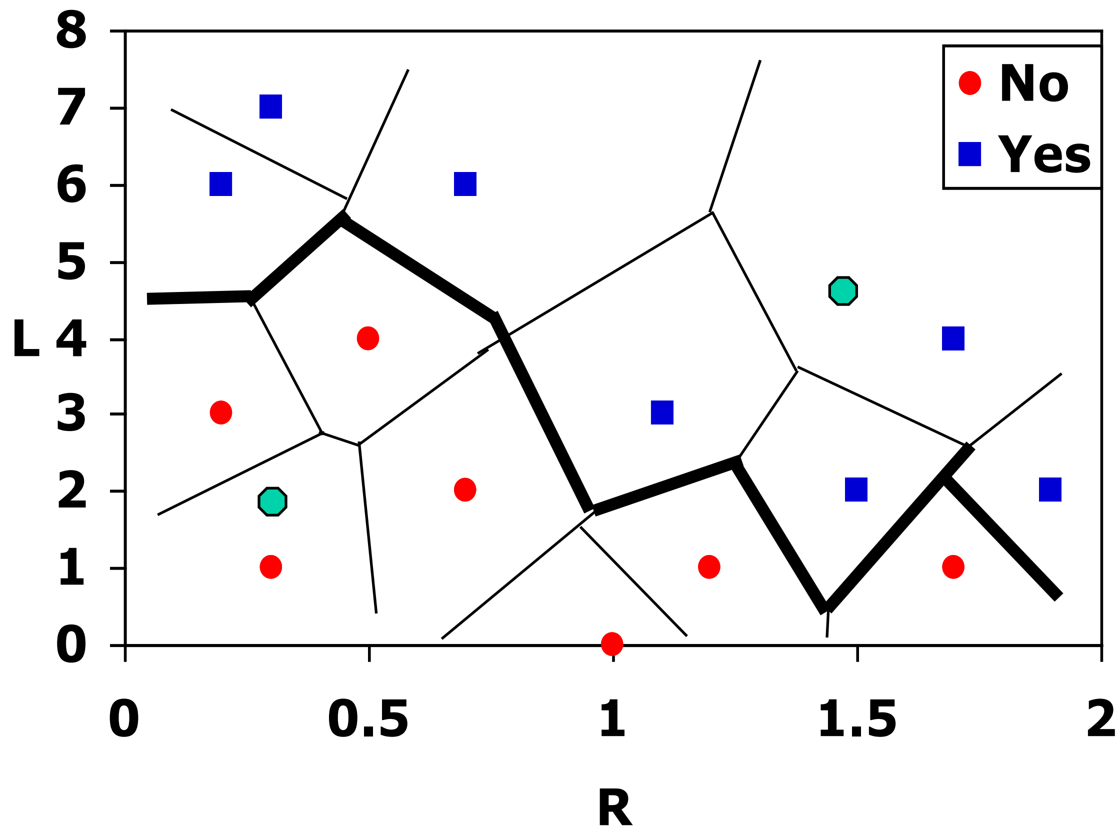
$$D(x^i, x^k) = \sqrt{\sum_j (L^i - L^k)^2 + (5R^i - 5R^k)^2}$$

Hypothesis



$$D(x^i, x^k) = \sqrt{\sum_j (L^i - L^k)^2 + (5R^i - 5R^k)^2}$$

Hypothesis



$$D(x^i, x^k) = \sqrt{\sum_j (L^i - L^k)^2 + (5R^i - 5R^k)^2}$$

Time and Space

- Learning is fast

Time and Space

- Learning is fast
- Lookup takes about m*n computations

Time and Space

- Learning is fast
- Lookup takes about m*n computations
  - storing data in a clever data structure (KD-tree) reduces this, on average, to log(m)*n

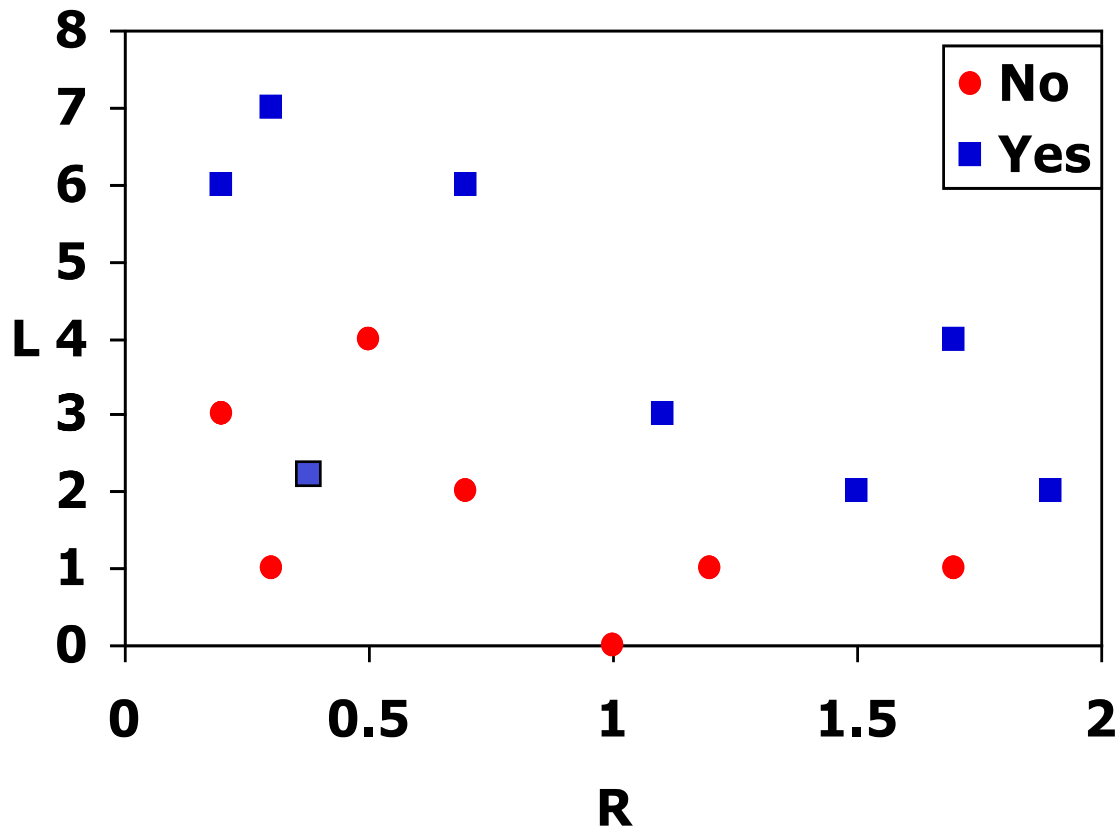
Time and Space

- Learning is fast
- Lookup takes about m*n computations
    - storing data in a clever data structure (KD-tree) reduces this, on average, to log(m)*n
- Memory can fill up with all that data

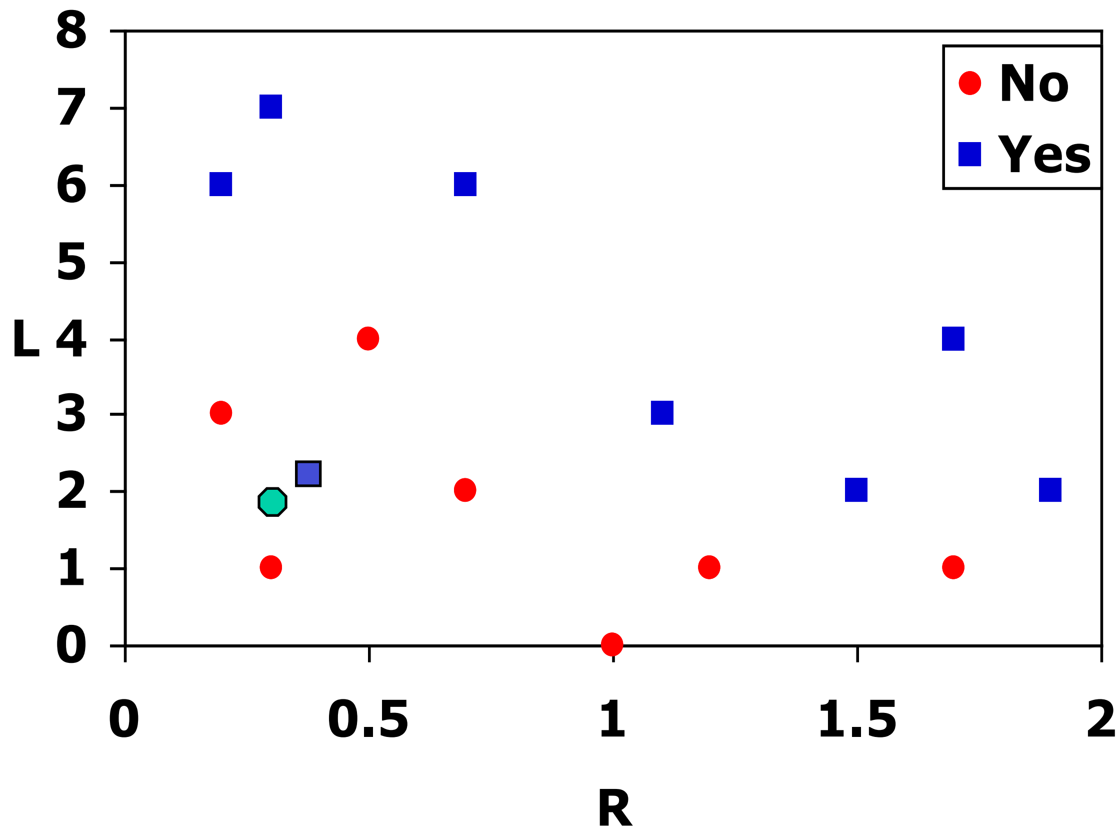
Time and Space

- Learning is fast
- Lookup takes about m*n computations
  - storing data in a clever data structure (KD-tree) reduces this, on average, to log(m)*n
- Memory can fill up with all that data
  - delete points that are far away from the boundary

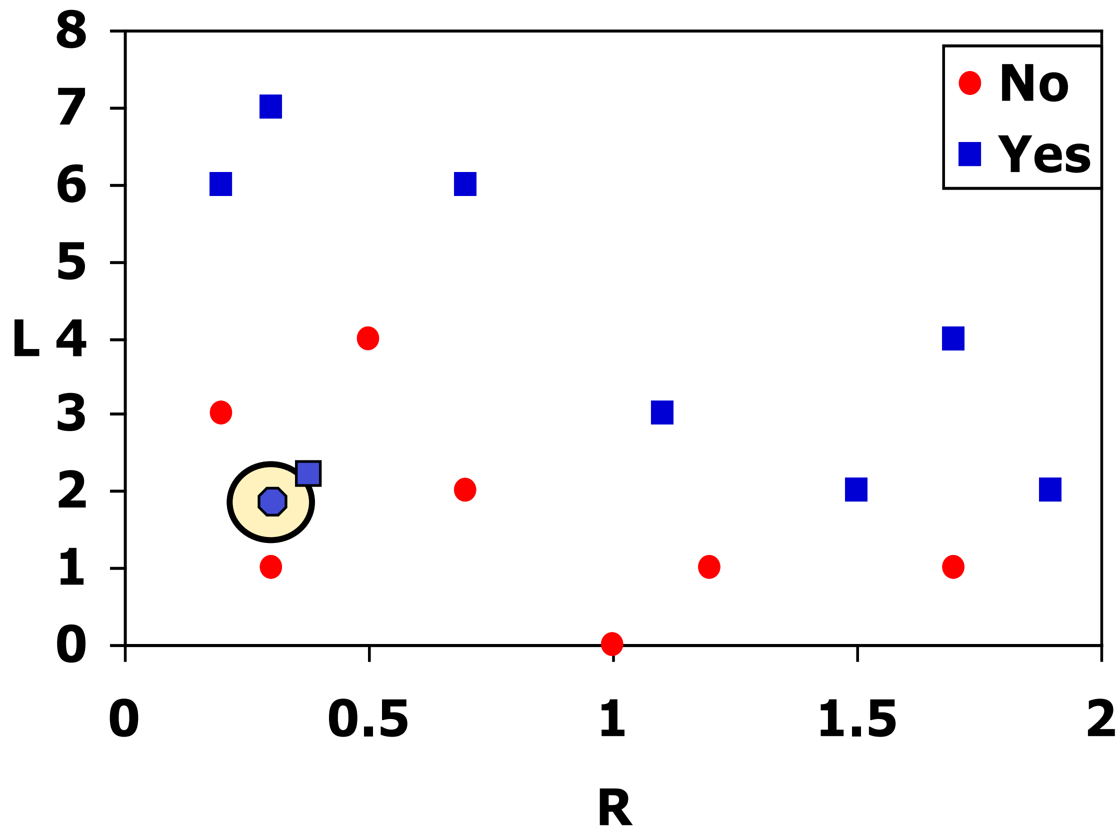
Noise



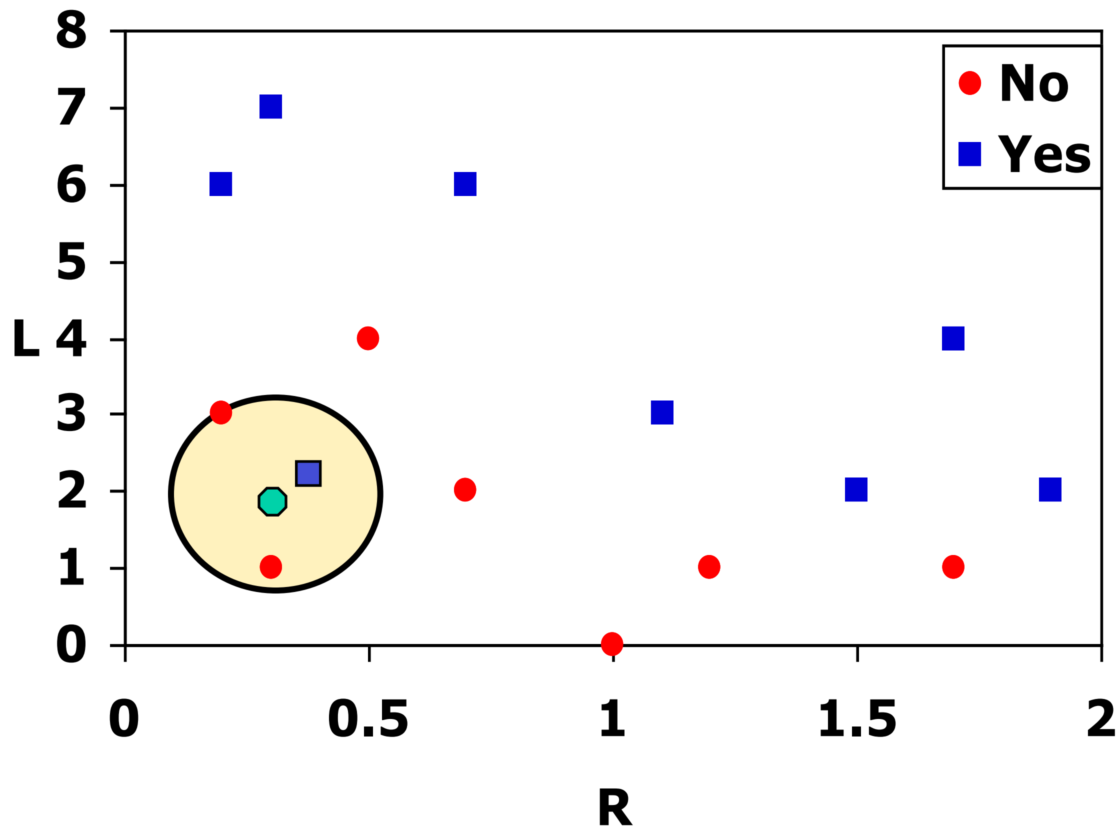
Noise



Noise

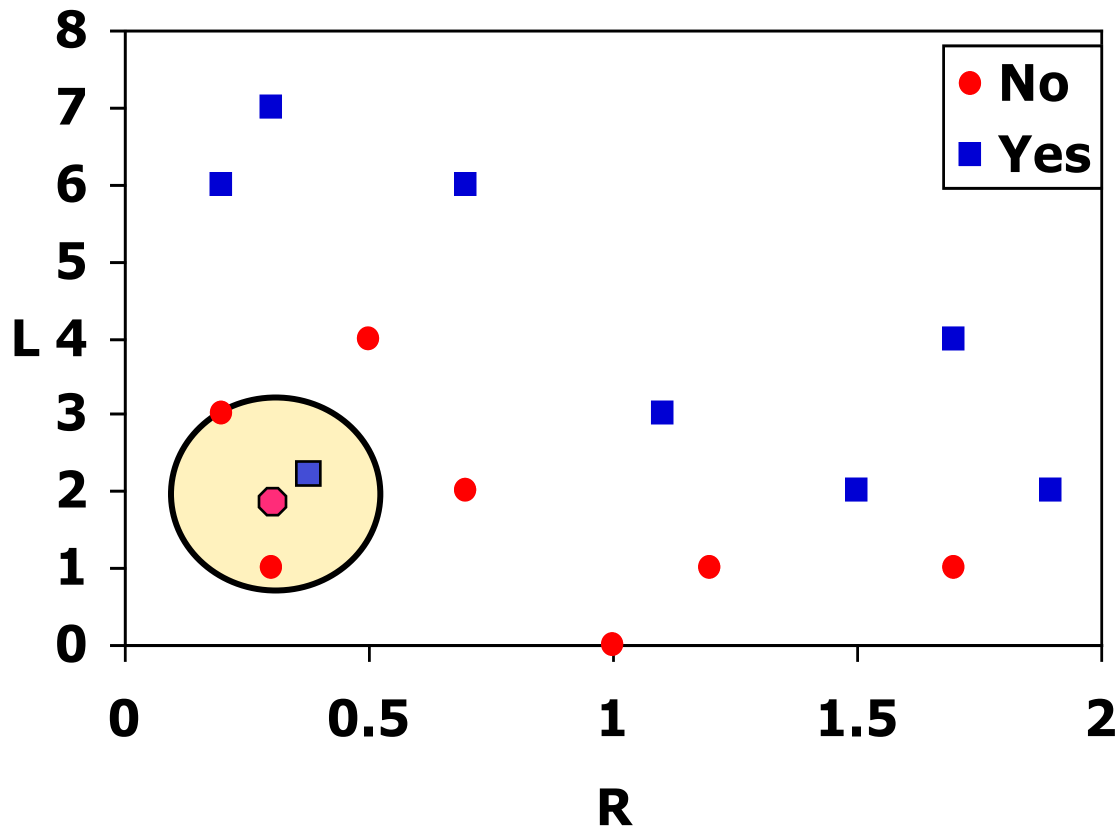


k-Nearest Neighbor



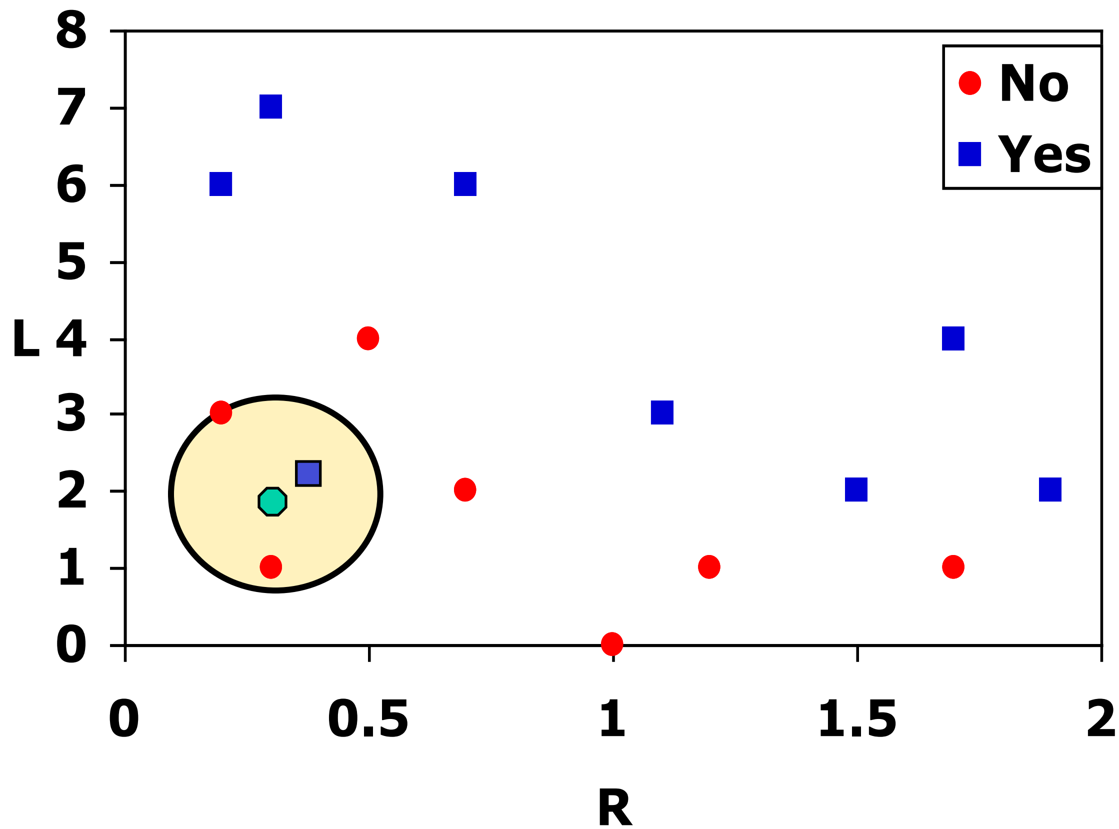
- Find the k nearest points

k-Nearest Neighbor



- Find the k nearest points
- Predict output according to the majority

k-Nearest Neighbor



- Find the k nearest points
- Predict output according to the majority
- Choose k using cross-validation

Curse of Dimensionality

- Nearest neighbor is great in low dimensions (up to about 6)
- As n increases, things get weird:

Curse of Dimensionality

- Nearest neighbor is great in low dimensions (up to about 6)
- As n increases, things get weird:
  - In high dimensions, almost all points are far away from one another
  - They're almost all near the boundaries

Curse of Dimensionality

- Nearest neighbor is great in low dimensions (up to about 6)
- As n increases, things get weird:
  - In high dimensions, almost all points are far away from one another
  - They're almost all near the boundaries
- Imagine sprinkling data points uniformly within a 10-dimensional unit cube
  - To capture 10% of the points, you'd need a cube with sides of length .63!

Curse of Dimensionality

- Nearest neighbor is great in low dimensions (up to about 6)
- As n increases, things get weird:
  - In high dimensions, almost all points are far away from one another
  - They' re almost all near the boundaries
- Imagine sprinkling data points uniformly within a 10-dimensional unit cube
  - To capture 10% of the points, you' d need a cube with sides of length .63!
- Cure: feature selection or more global models

Test Domains

Test Domains

- Heart Disease: predict whether a person has significant narrowing of the arteries, based on tests
  - 26 features
  - 297 data points

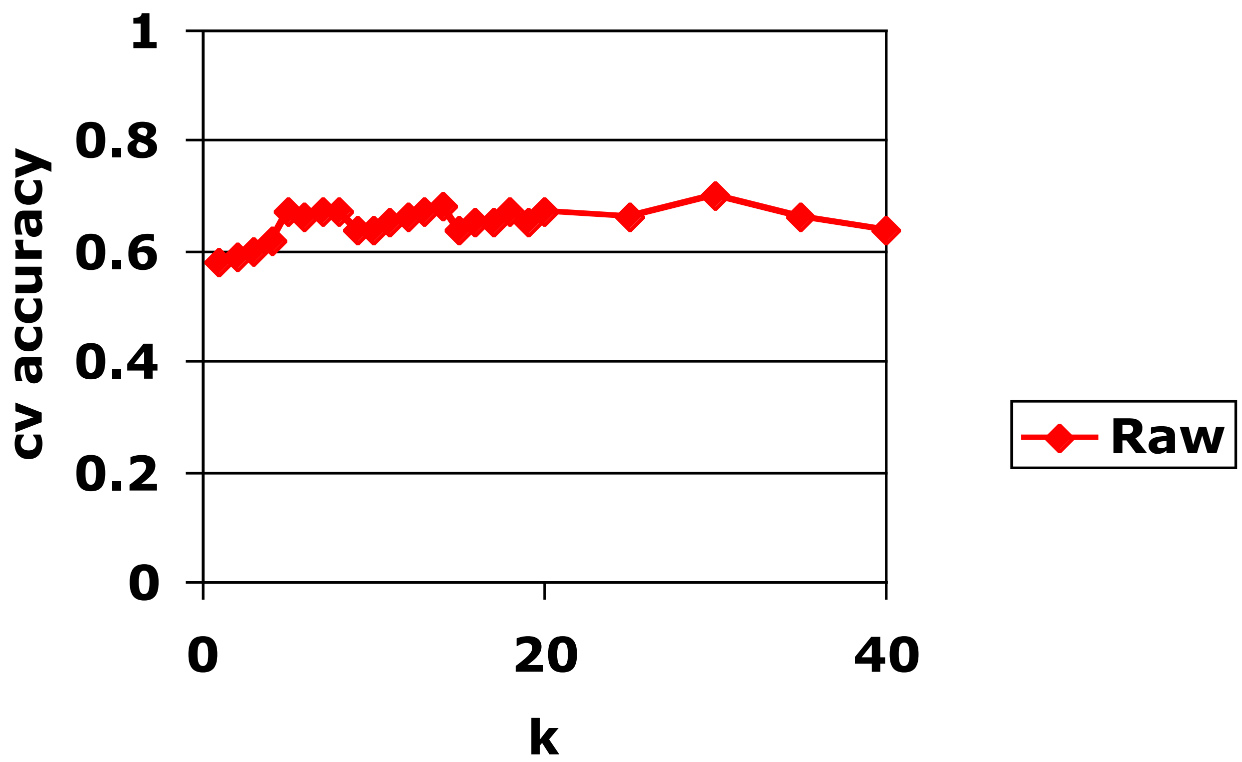
Test Domains

- Heart Disease: predict whether a person has significant narrowing of the arteries, based on tests
  - 26 features
  - 297 data points

- Auto MPG: predict whether a car gets more than 22 miles per gallon, based on attributes of car
  - 12 features
  - 385 data points

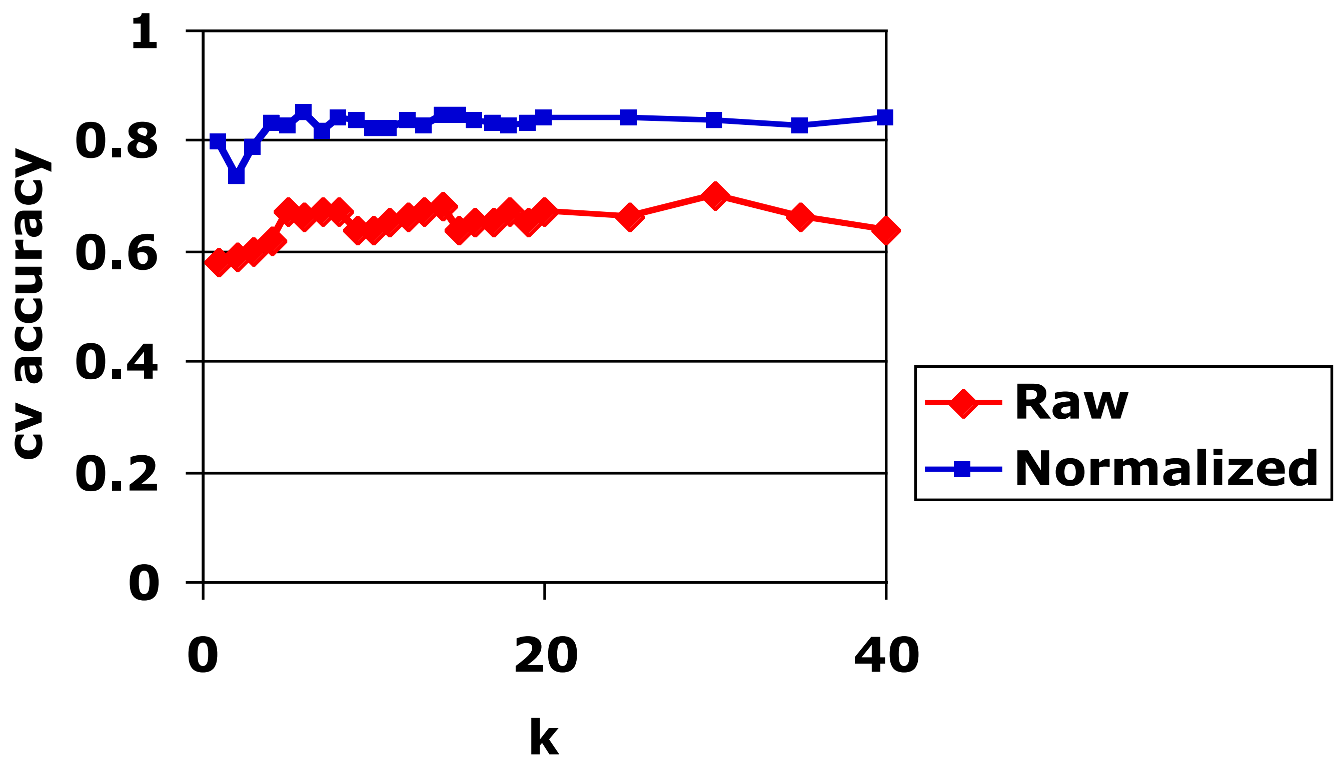
Heart Disease

- Relatively insensitive to k



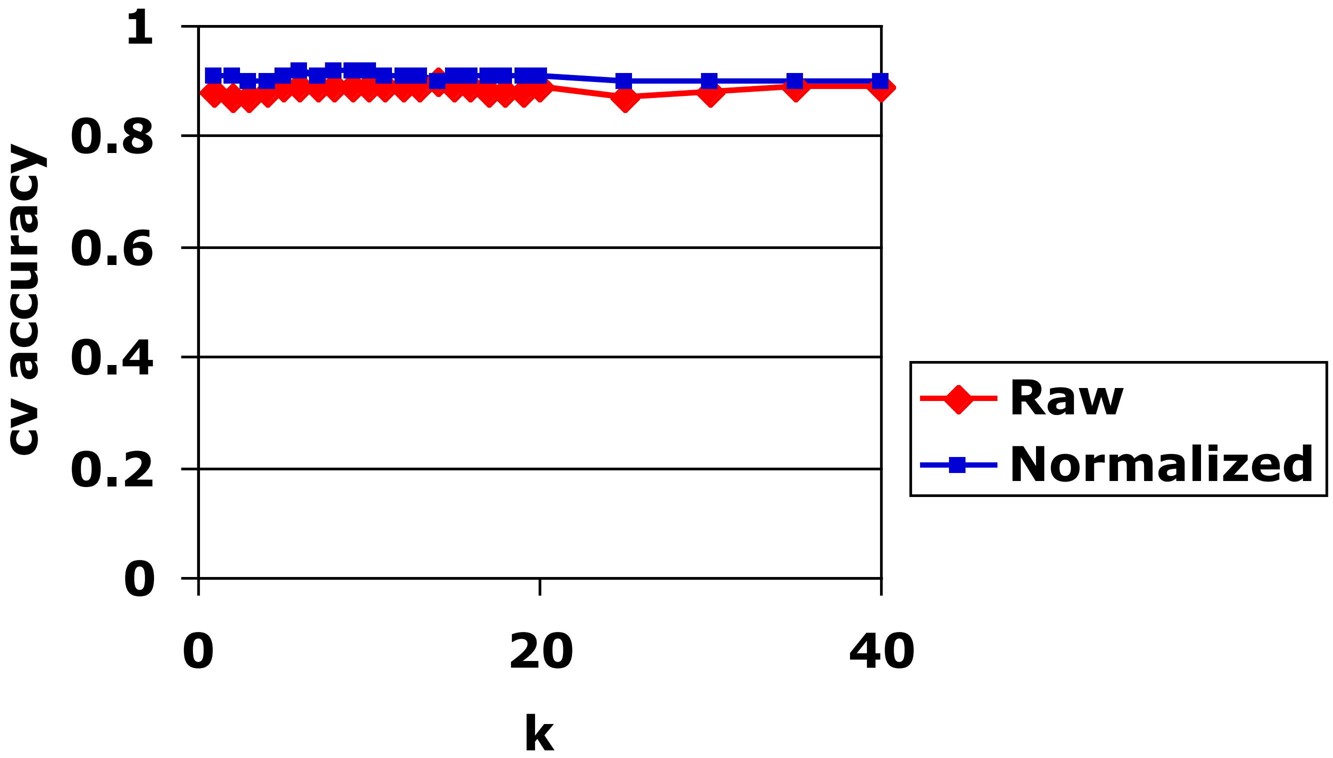
Heart Disease

- Relatively insensitive to k
- Normalization matters!



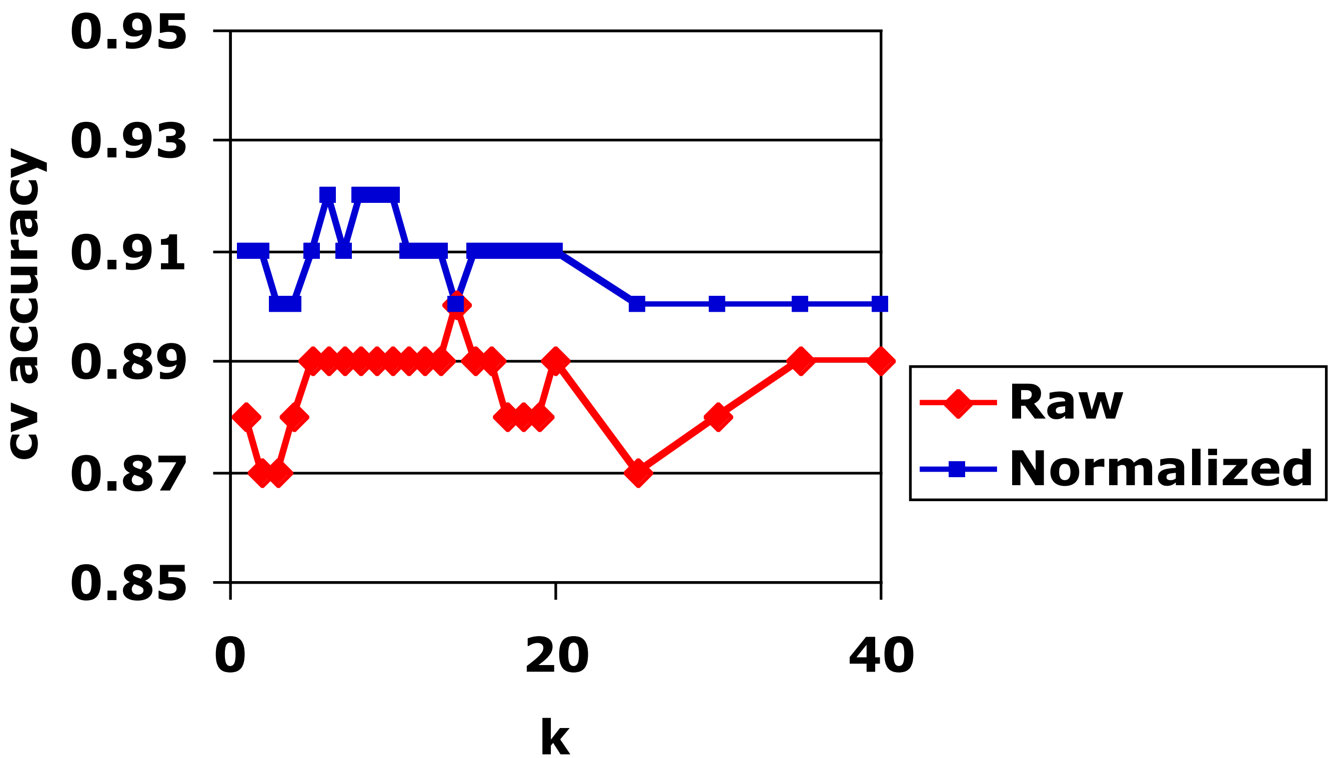
Auto MPG

- Relatively insensitive to k
- Normalization doesn't matter much



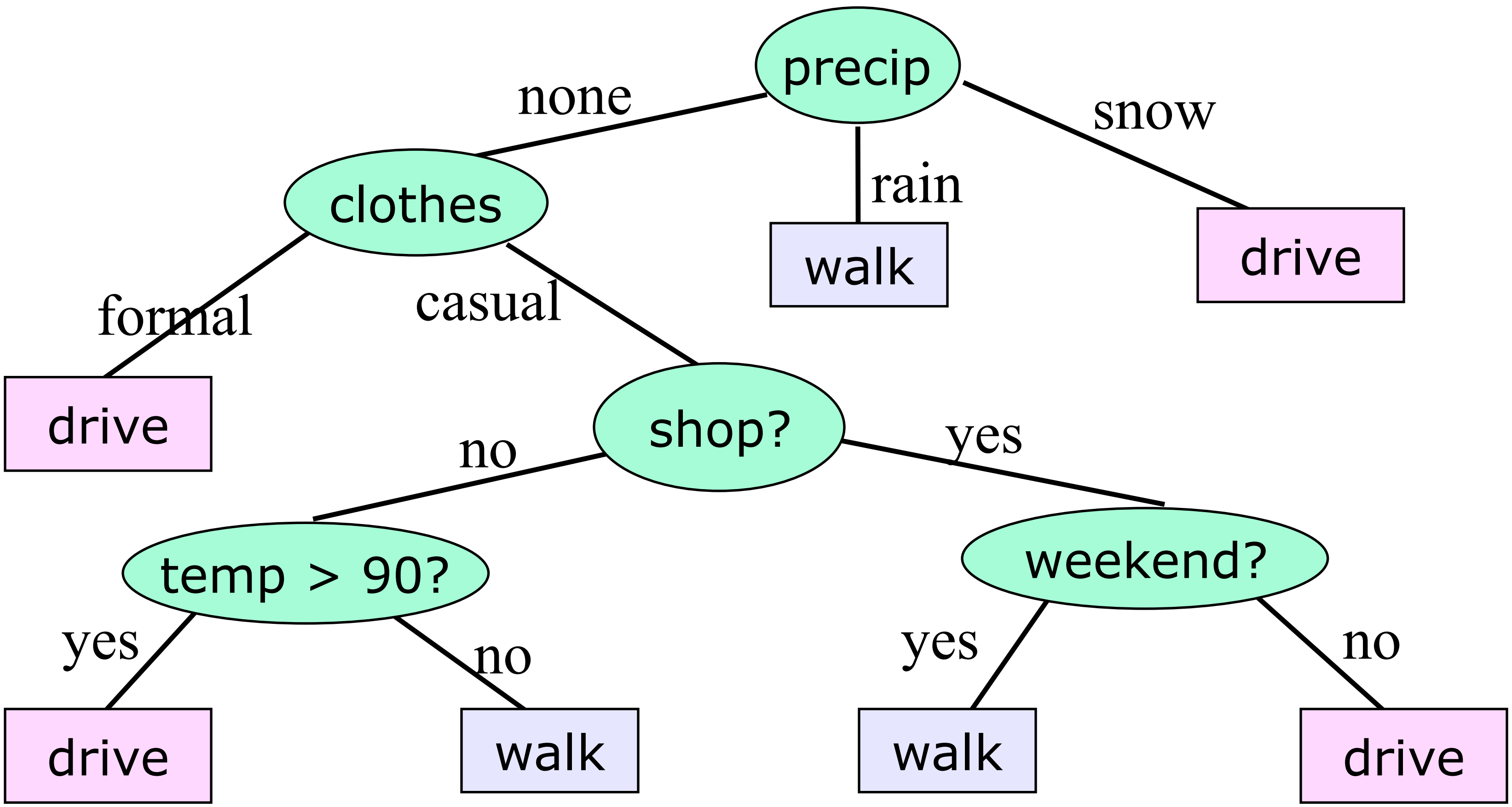
Auto MPG

- Now normalization matters a lot!
- Watch the scales on your graphs



Remember Decision Trees

Use all the data to build a tree of questions with answers at the leaves



Numerical Attributes

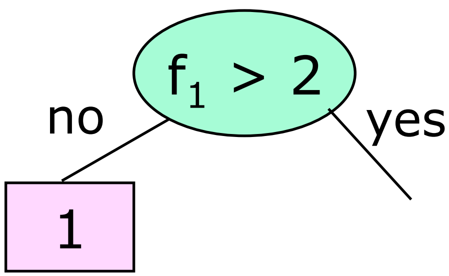
- Tests in nodes can be of the form x_j > constant

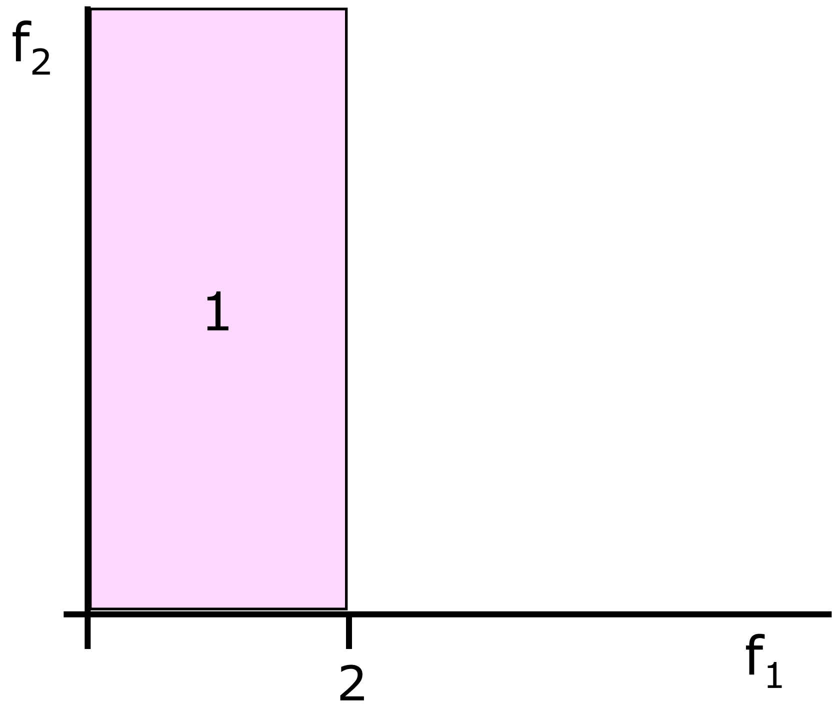
Numerical Attributes

- Tests in nodes can be of the form x_j > constant
- Divides the space into axis-aligned rectangles

Numerical Attributes

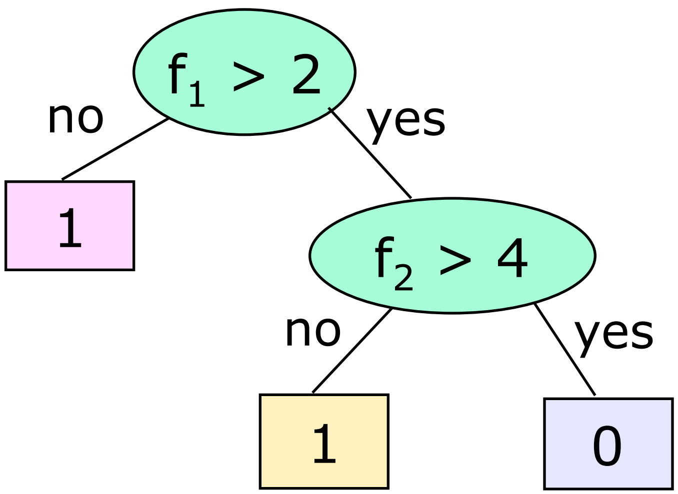
- Tests in nodes can be of the form x_j > constant
- Divides the space into axis-aligned rectangles

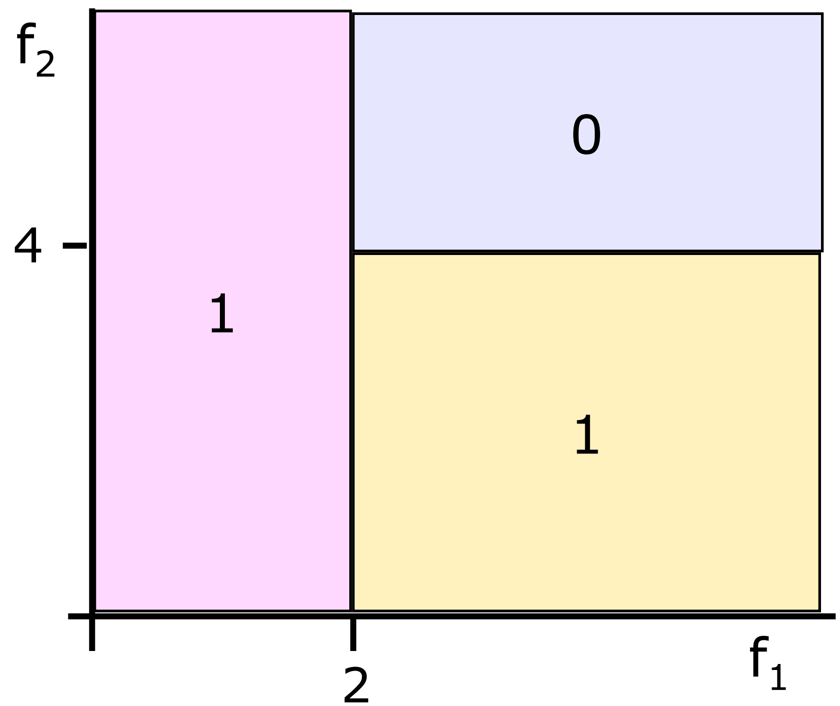




Numerical Attributes

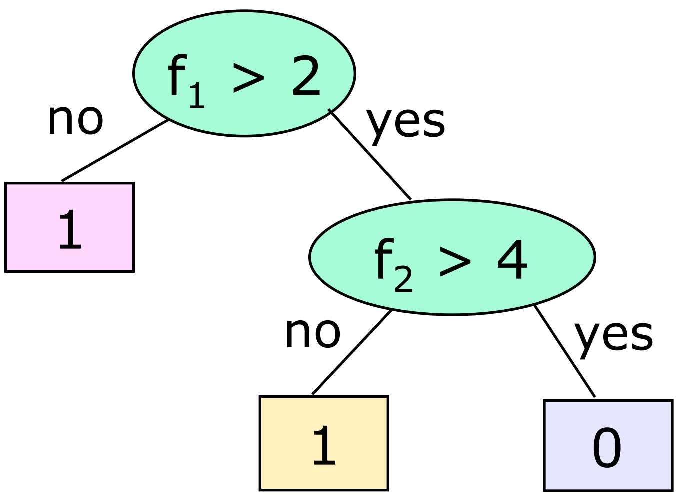
- Tests in nodes can be of the form x_j > constant
- Divides the space into axis-aligned rectangles

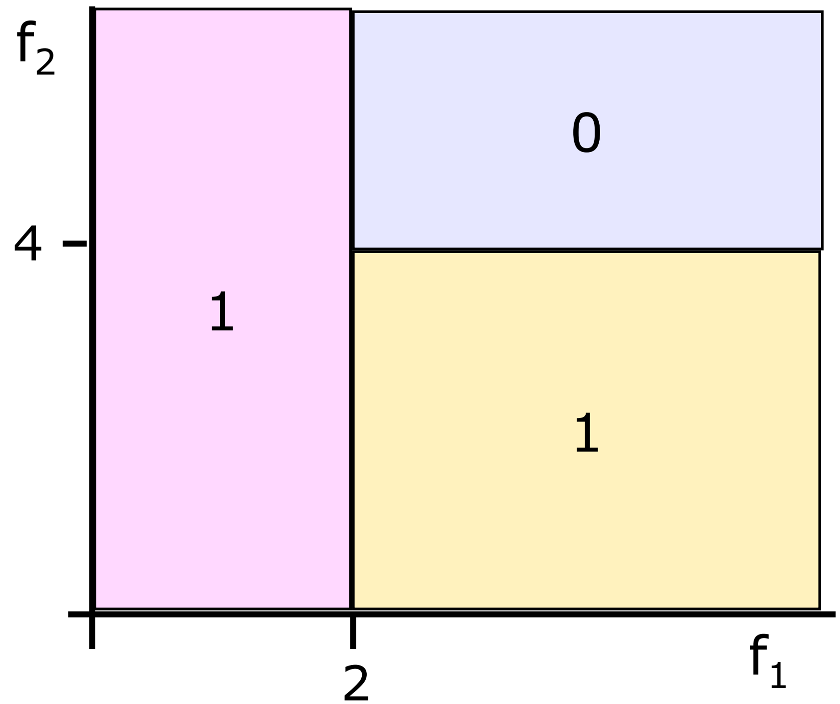




Numerical Attributes

- Tests in nodes can be of the form x_j > constant
- Divides the space into axis-aligned rectangles

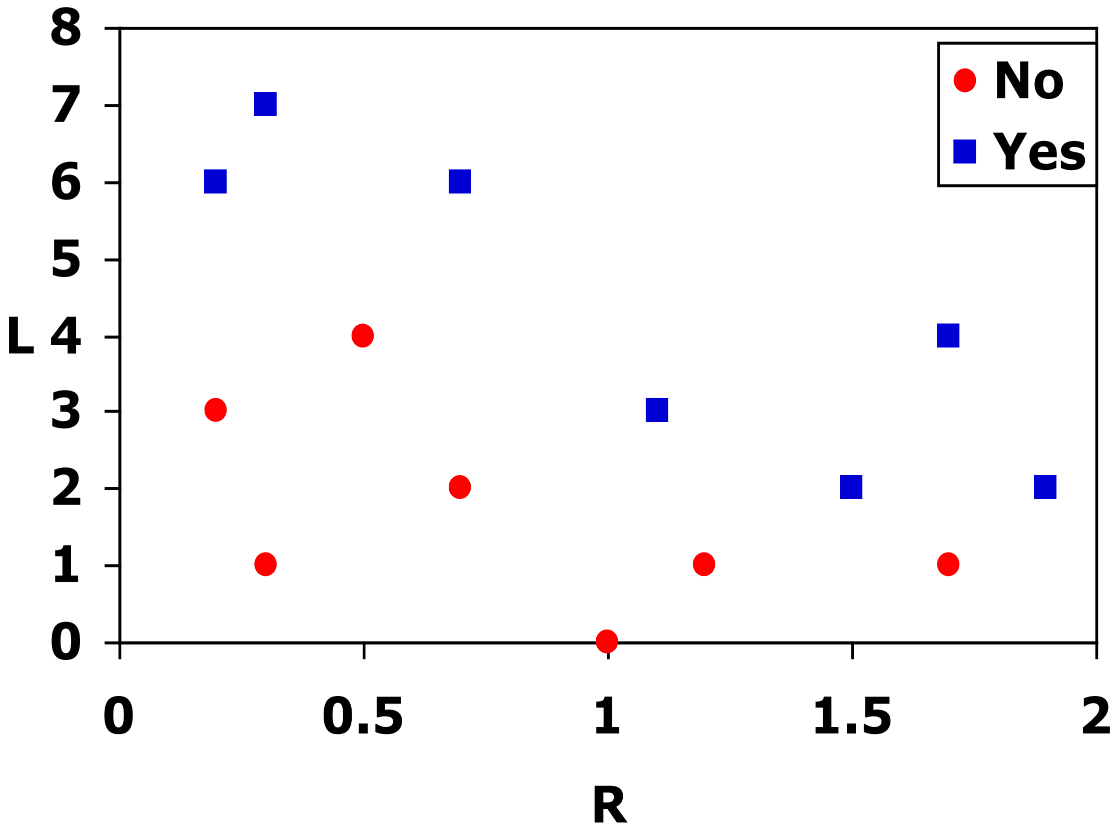




- Non-axis aligned hypotheses can be smaller but hard to find

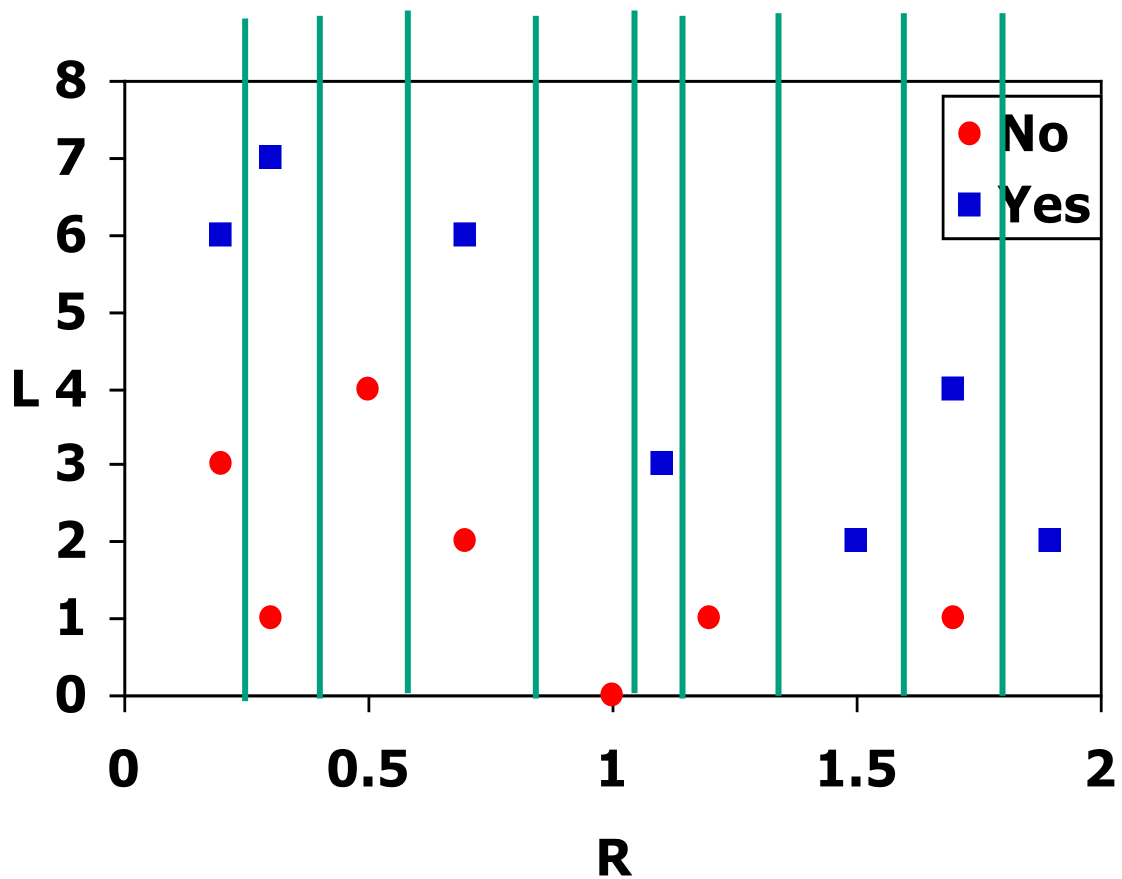
Considering Splits

- Consider a split between each point in each dimension



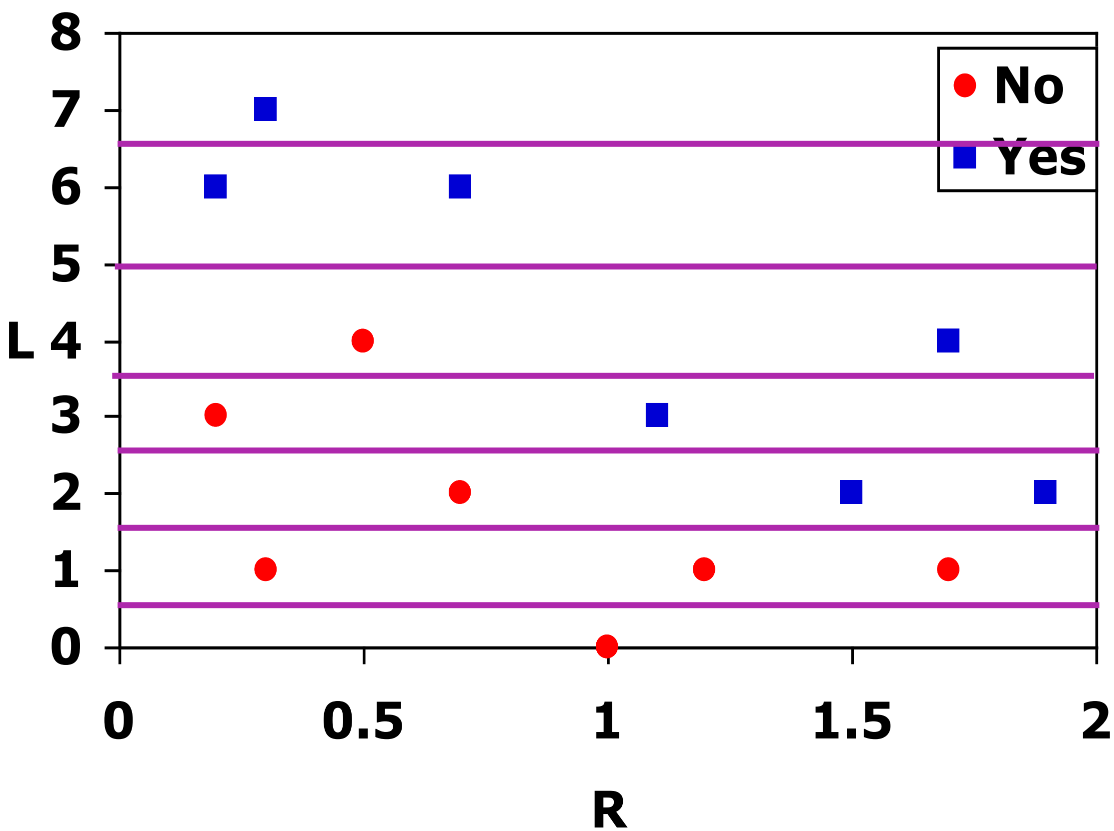
Considering Splits

- Consider a split between each point in each dimension



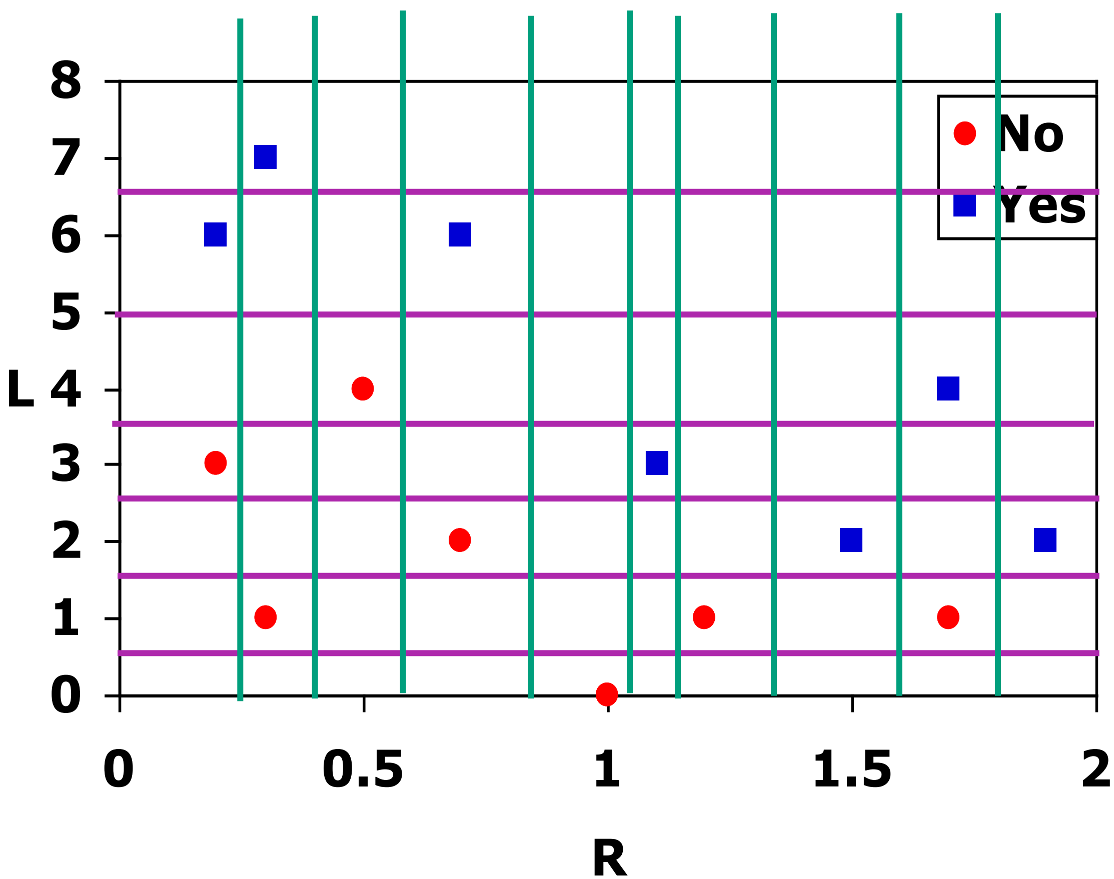
Considering Splits

- Consider a split between each point in each dimension



Considering Splits

- Choose split that minimizes average entropy of child nodes



Bankruptcy Example

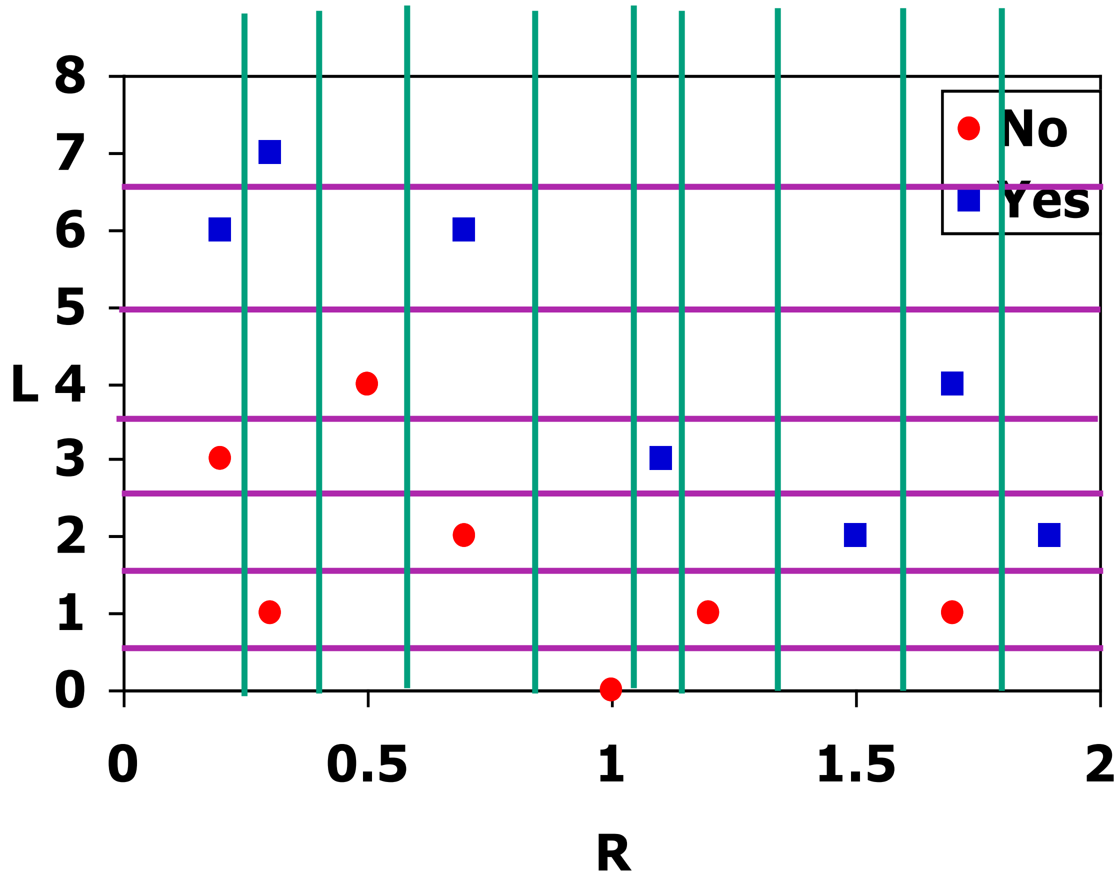
L<y	NL	PL	NR	PR	AE
6.5	7	6	0	1	0.93
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.93

neg to left

pos to left

neg to right

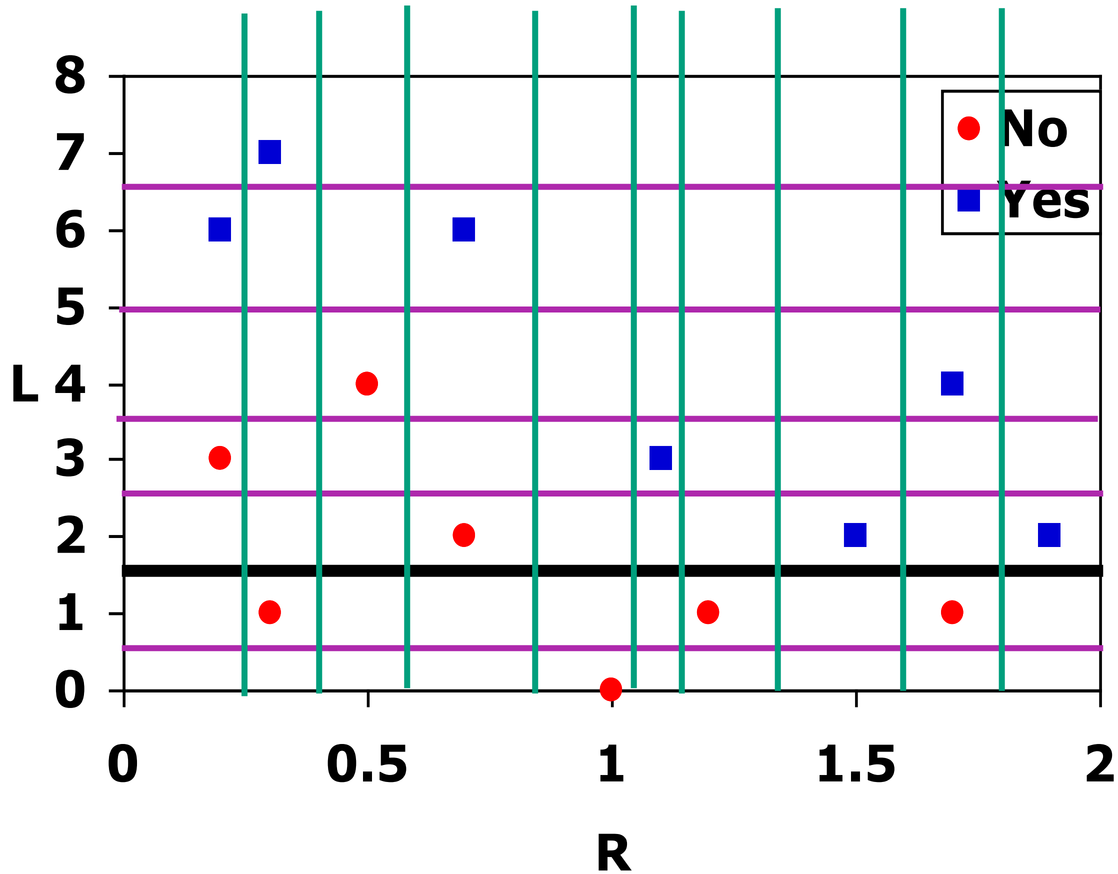
pos to right



AE	1.00	1.00	0.98	0.98	0.94	0.98	0.92	0.98	0.92
R<x	0.25	0.40	0.60	0.85	1.05	1.15	1.35	1.60	1.80

Bankruptcy Example

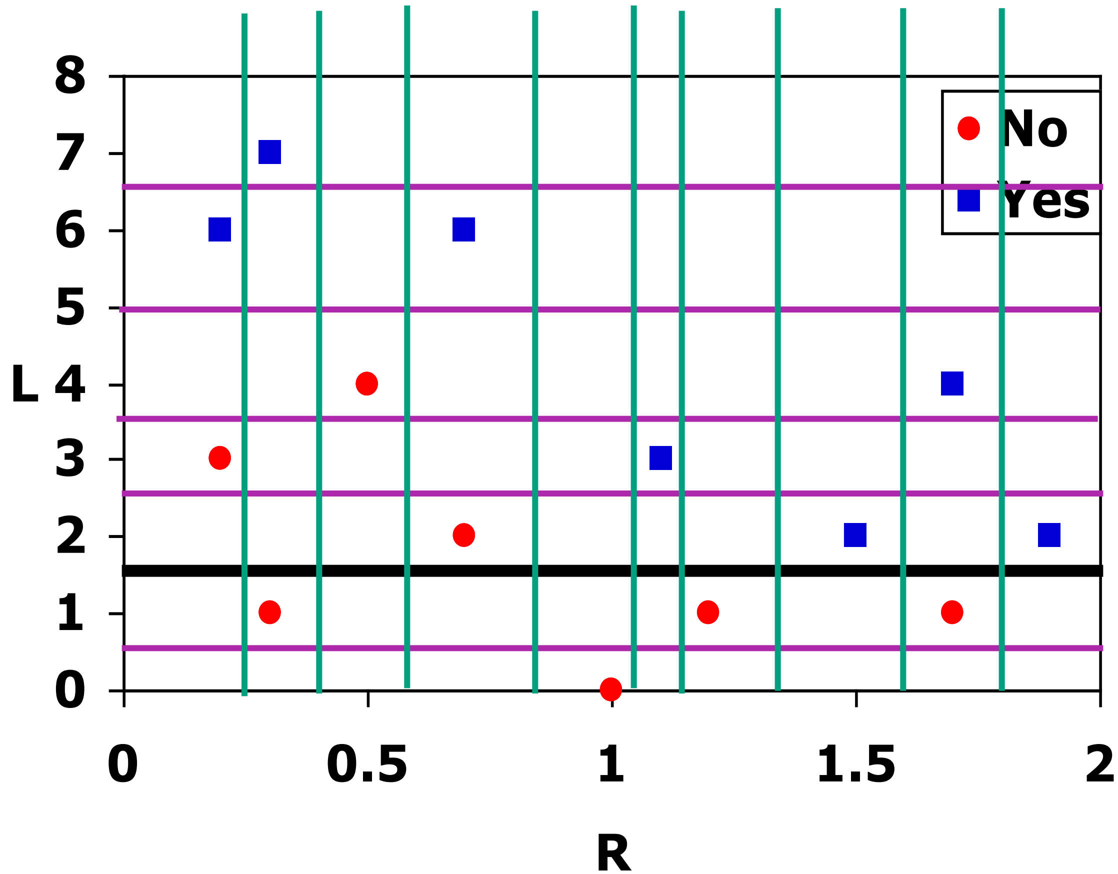
L<y	NL	PL	NR	PR	AE
6.5	7	6	0	1	0.93
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.93

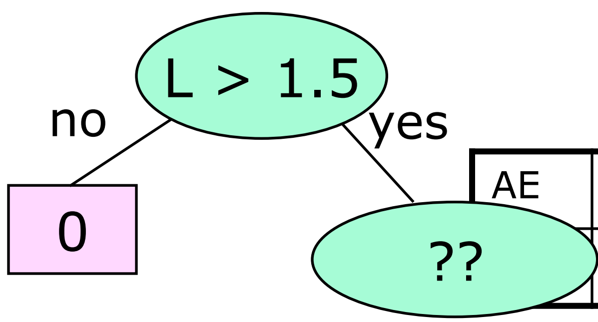


AE	1.00	1.00	0.98	0.98	0.94	0.98	0.92	0.98	0.92
R<x	0.25	0.40	0.60	0.85	1.05	1.15	1.35	1.60	1.80

Bankruptcy Example

L<y	NL	PL	NR	PR	AE
6.5	7	6	0	1	0.93
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.93

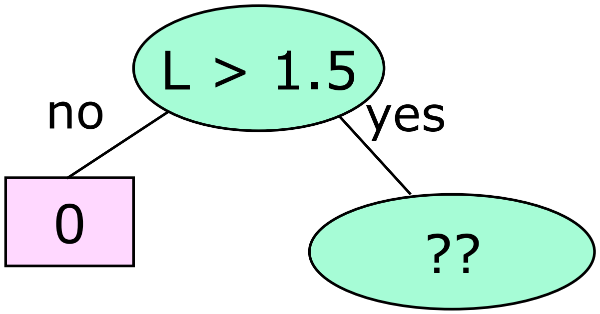


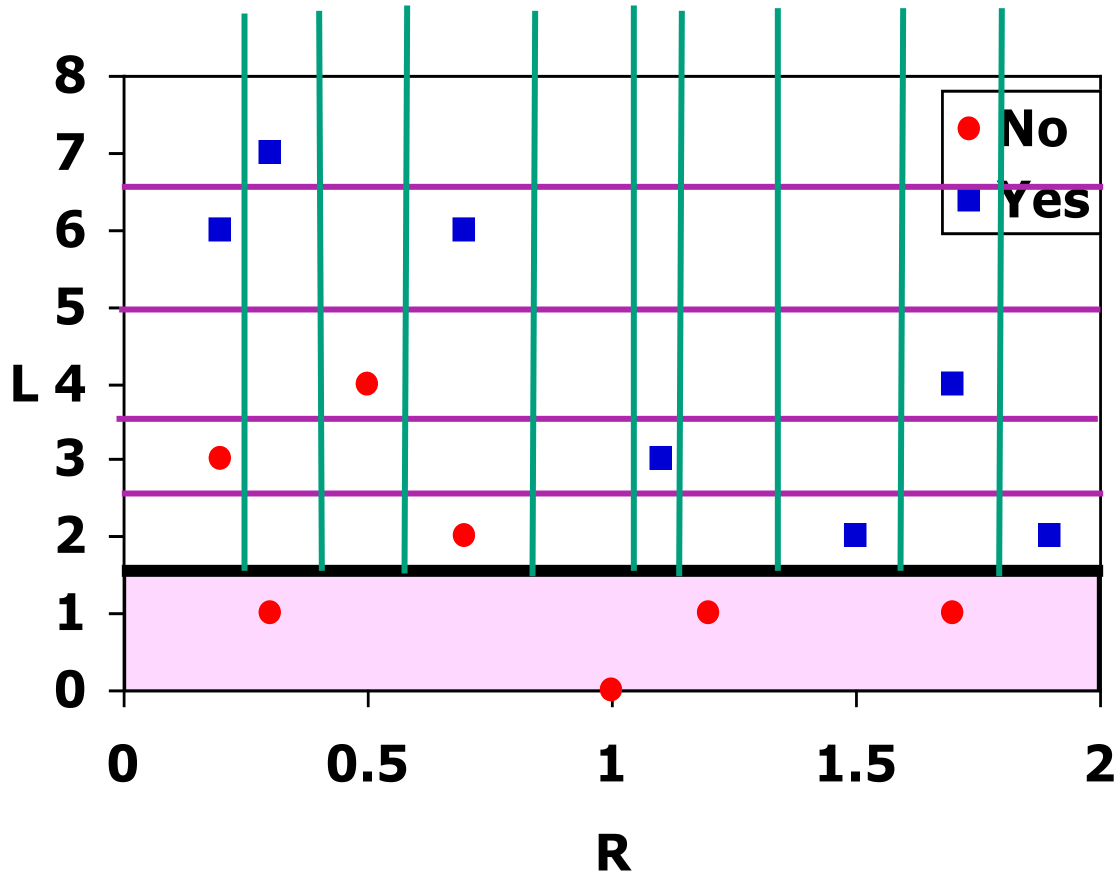


AE	1.00	1.00	0.98	0.98	0.94	0.98	0.92	0.98	0.92
	0.25	0.40	0.60	0.85	1.05	1.15	1.35	1.60	1.80

Bankruptcy Example

L<y	NL	PL	NR	PR	AE
6.5	6	3	0	1	0.83
5.0	4	3	0	3	0.69
3.5	3	2	4	1	0.85
2.5	2	1	5	2	0.88

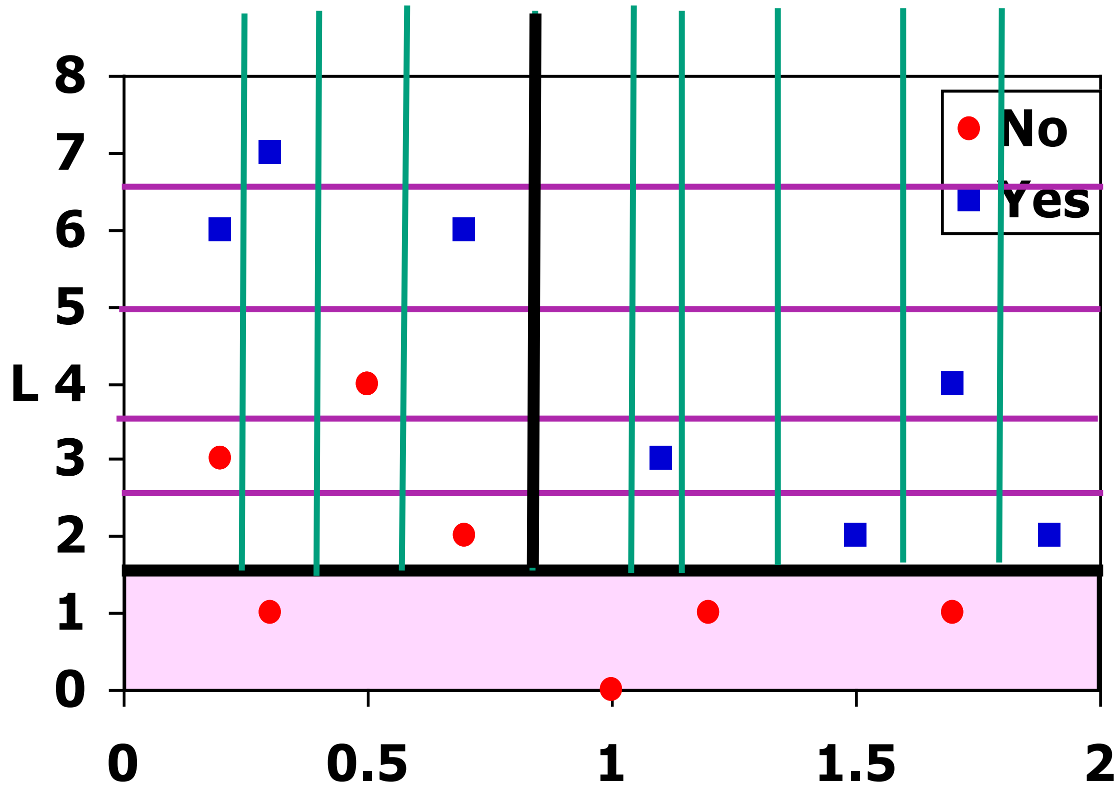


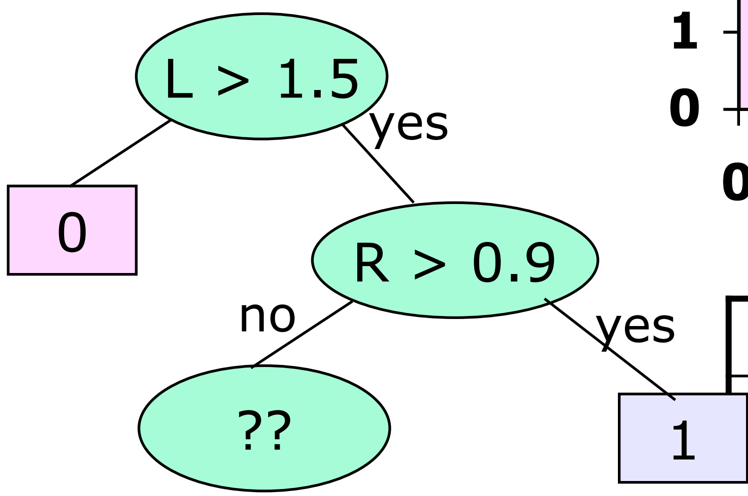


AE	0.85	0.88	0.79	0.60	0.69	0.76	0.83
R<x	0.25	0.40	0.60	0.90	1.30	1.60	1.80

Bankruptcy Example

L<y	NL	PL	NR	PR	AE
6.5	6	3	0	1	0.83
5.0	4	3	0	3	0.69
3.5	3	2	4	1	0.85
2.5	2	1	5	2	0.88

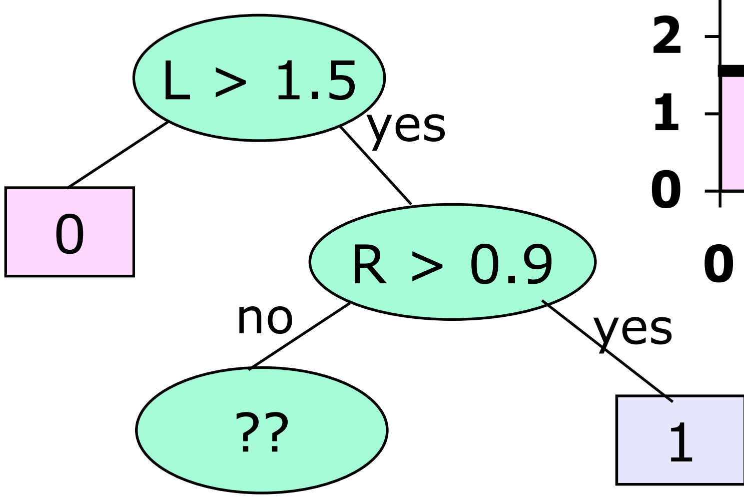


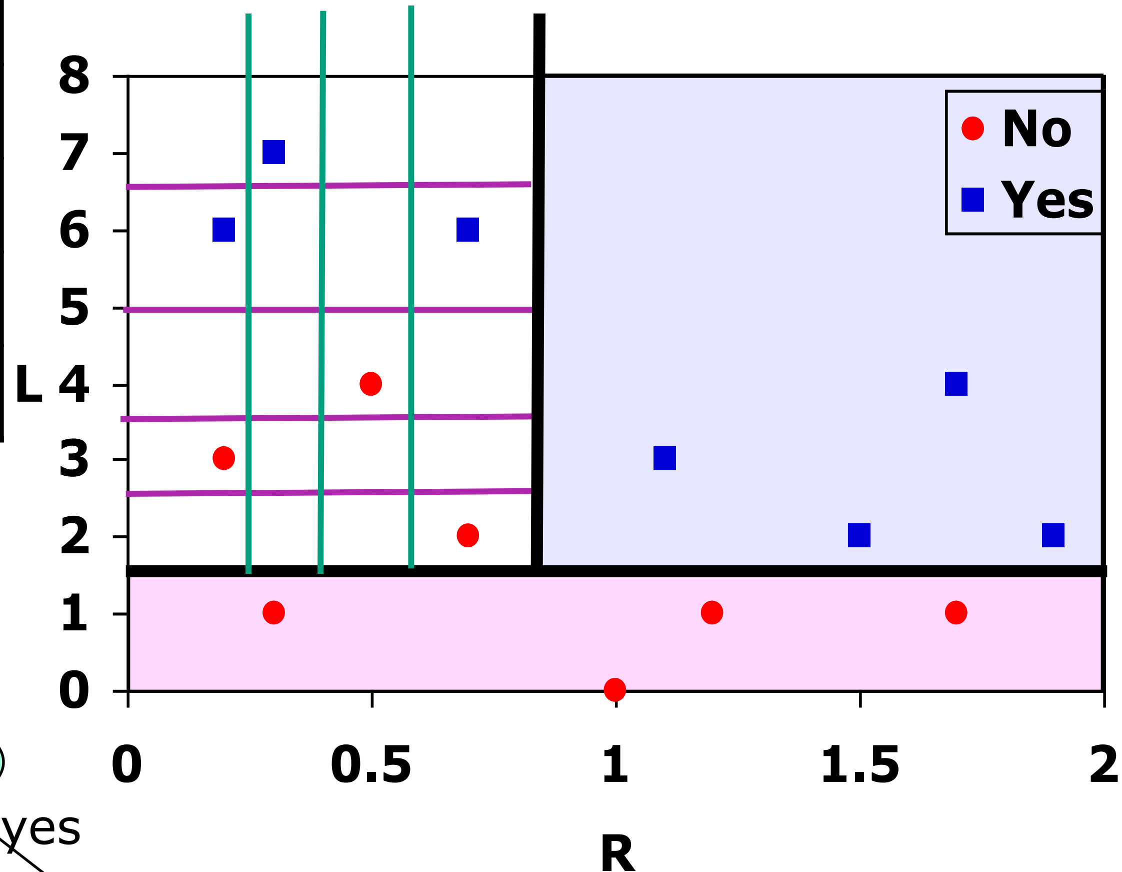


AD	0.85	0.88	0.79	0.60	0.69	0.76	0.83
R<x	0.25	0.40	0.60	0.90	1.30	1.60	1.80

Bankruptcy Example

L<y	NL	PL	NR	PR	AE
6.5	3	2	0	1	0.81
5.0	3	0	0	3	0.00
3.5	2	0	1	3	0.54
2.5	1	0	2	3	0.81

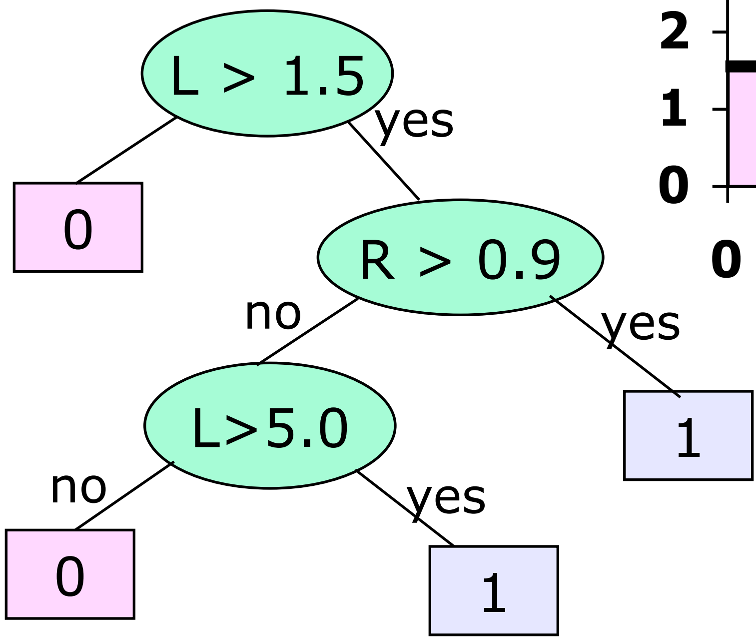


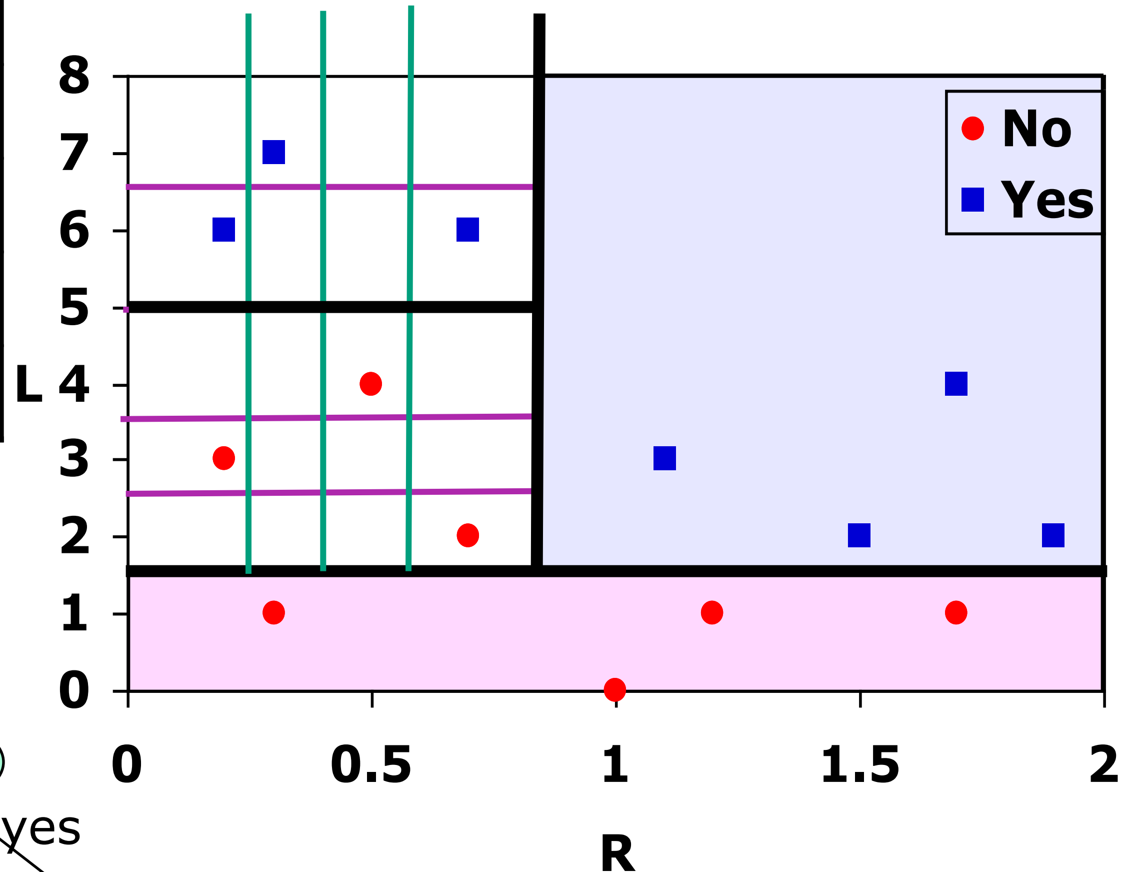


AE	1.00	0.92	1.00
R<x	0.25	0.40	0.60

Bankruptcy Example

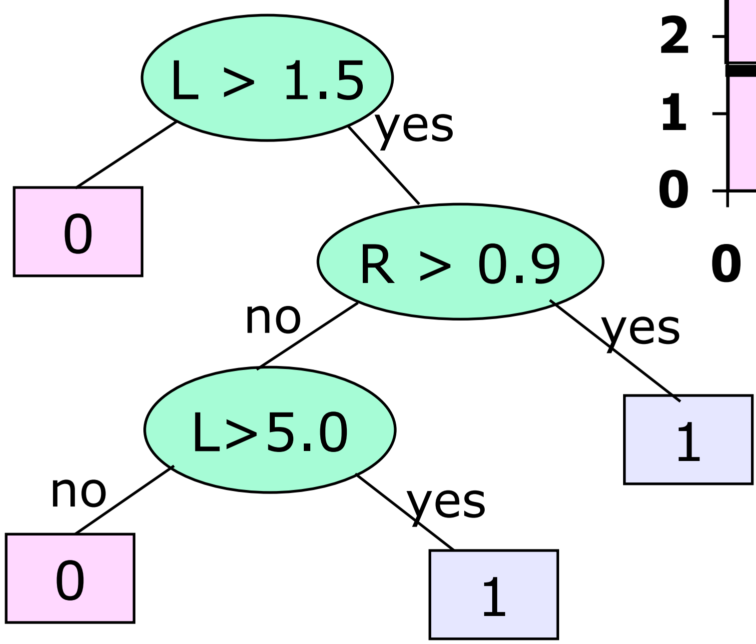
L<y	NL	PL	NR	PR	AE
6.5	3	2	0	1	0.81
5.0	3	0	0	3	0.00
3.5	2	0	1	3	0.54
2.5	1	0	2	3	0.81

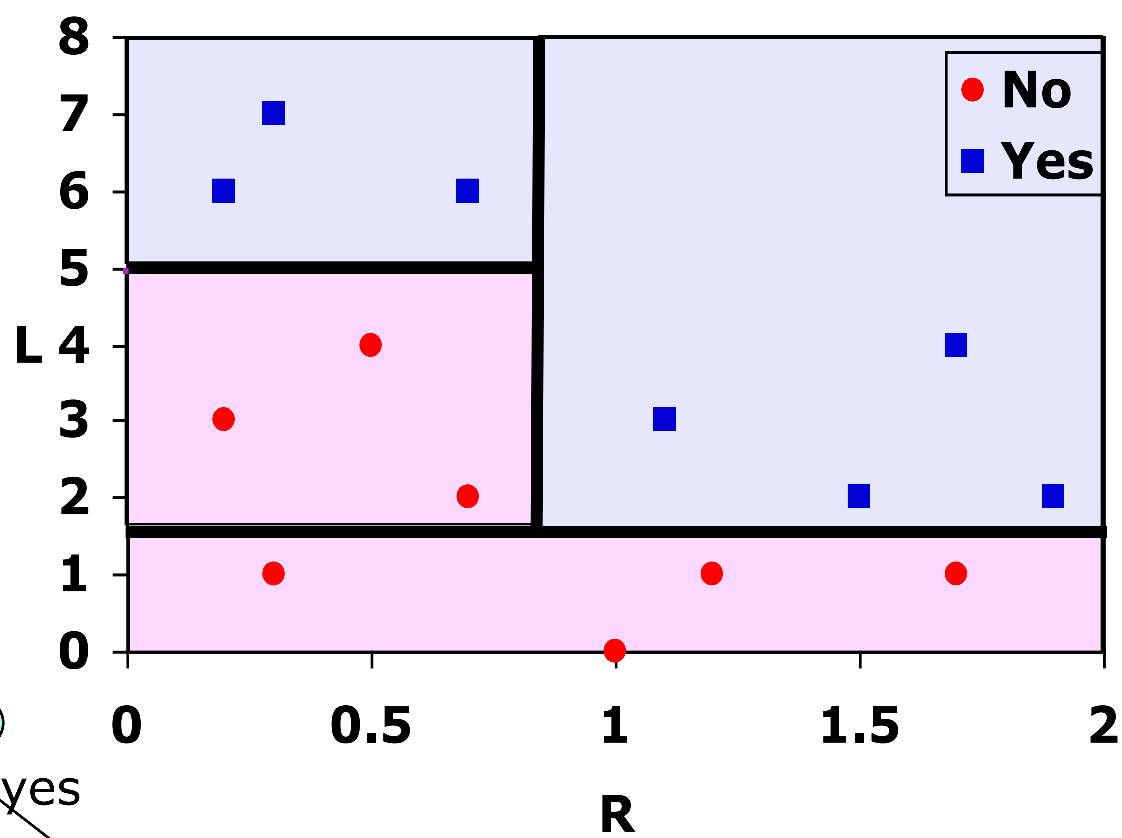




AE	1.00	0.92	1.00
R<x	0.25	0.40	0.60

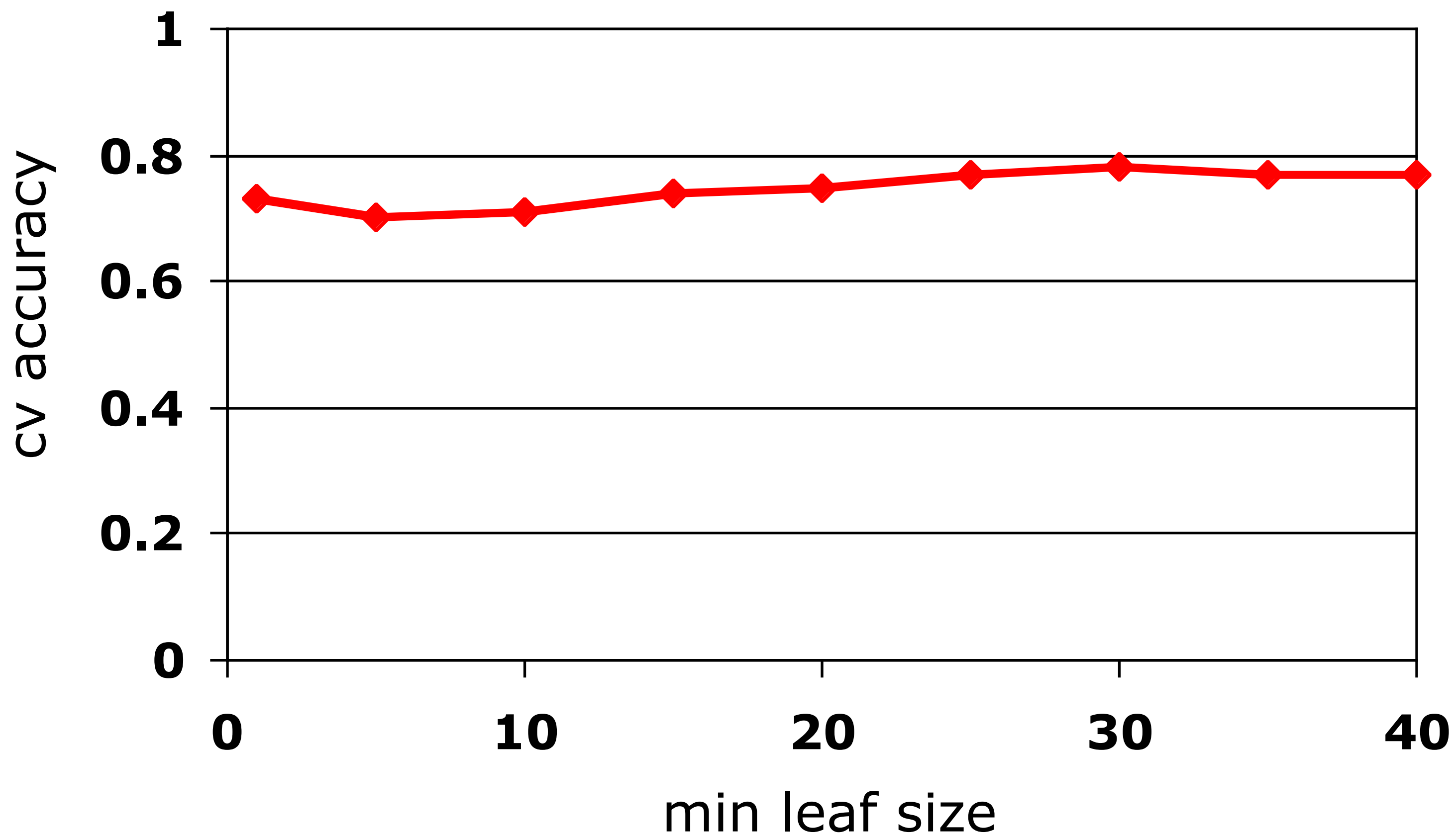
Bankruptcy Example



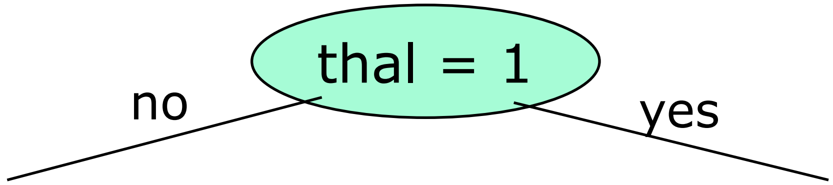


Heart Disease

- Best performance (.77) slightly worse than nearest neighbor (.81)

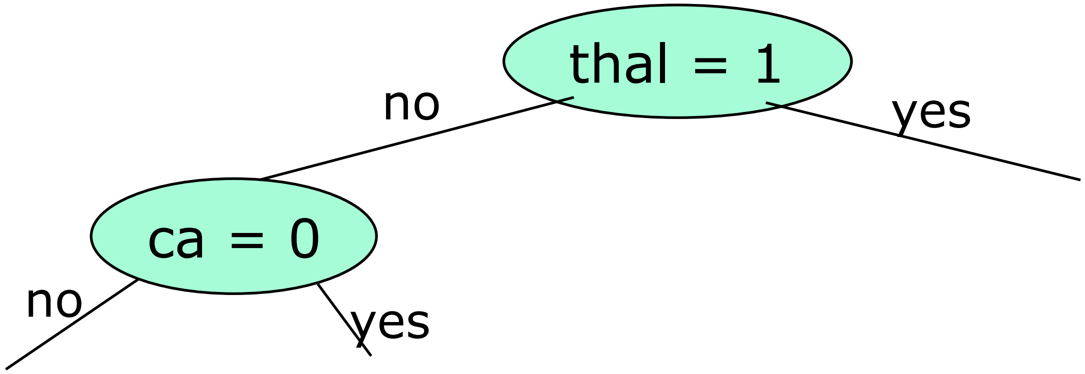


Heart Disease



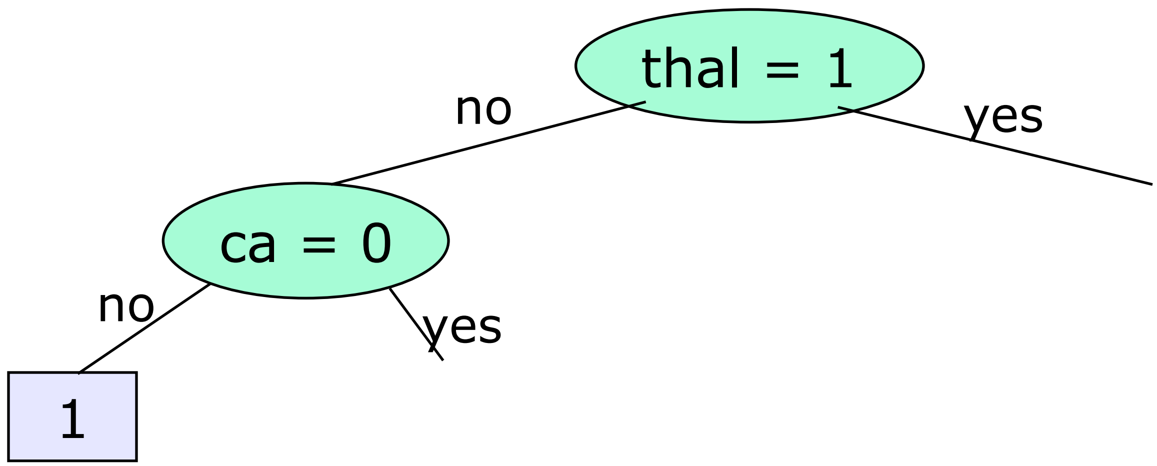
thal = 1: normal exercise thallium scintigraphy test

Heart Disease



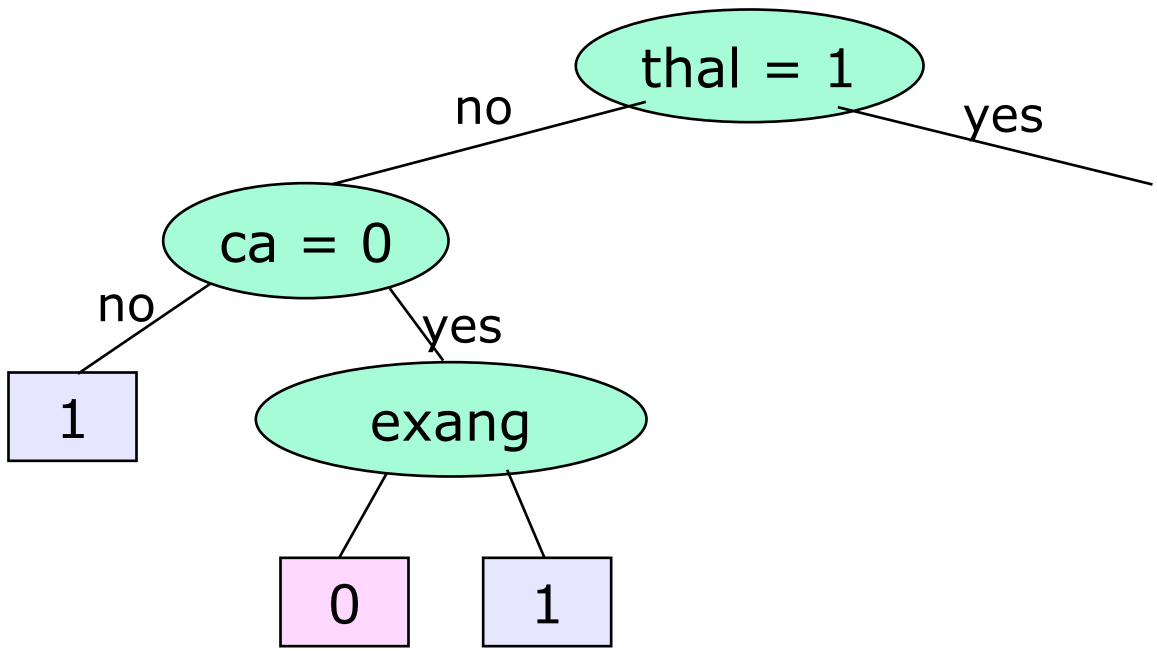
thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy

Heart Disease



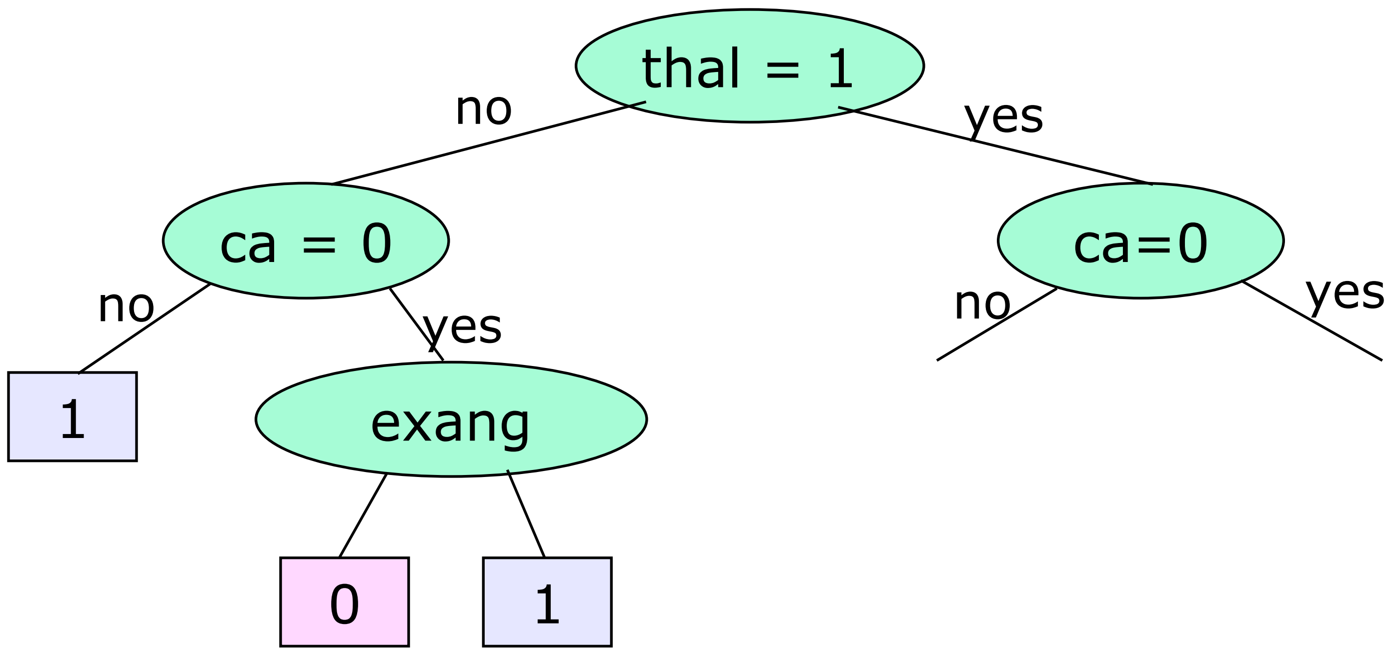
thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy

Heart Disease



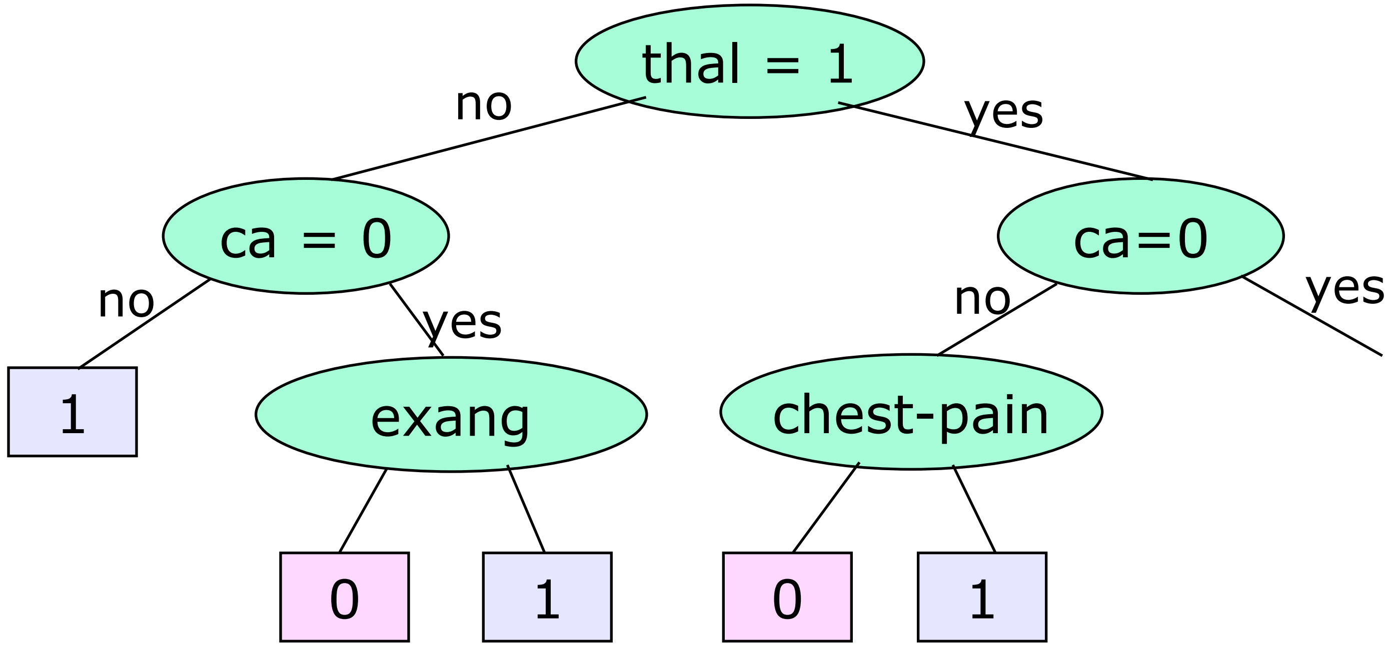
thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy
exang: exercise induced angina

Heart Disease



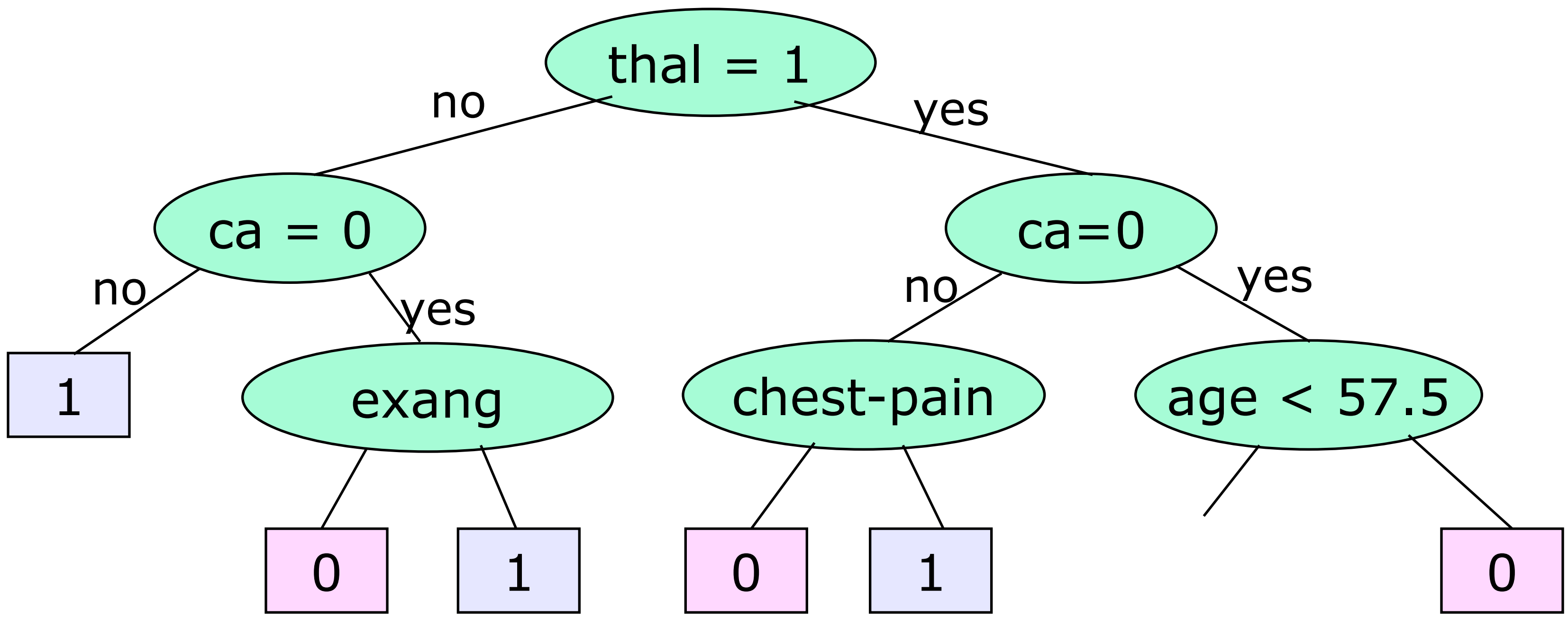
thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy
exang: exercise induced angina

Heart Disease



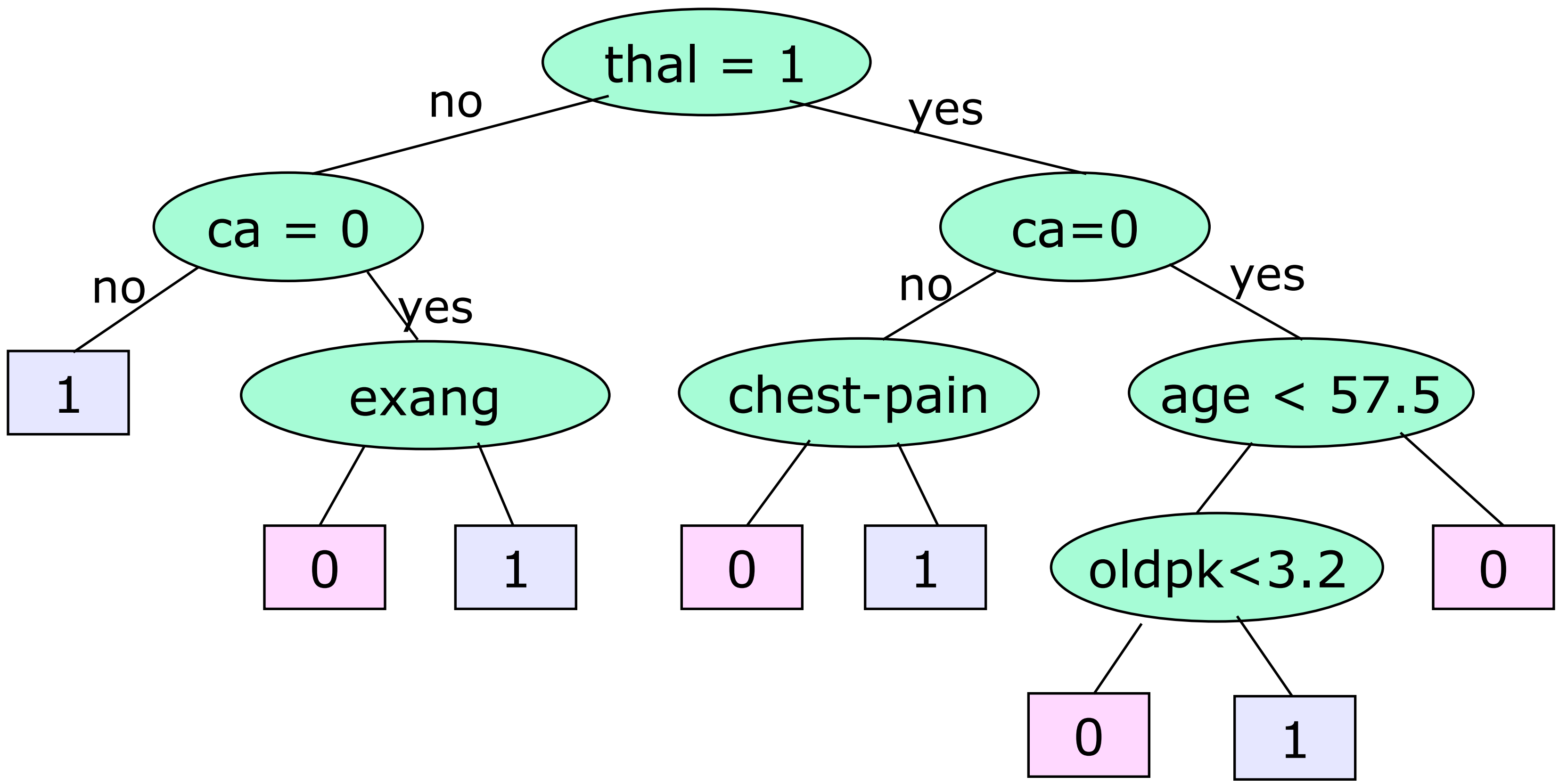
thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy
exang: exercise induced angina

Heart Disease



thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy
exang: exercise induced angina

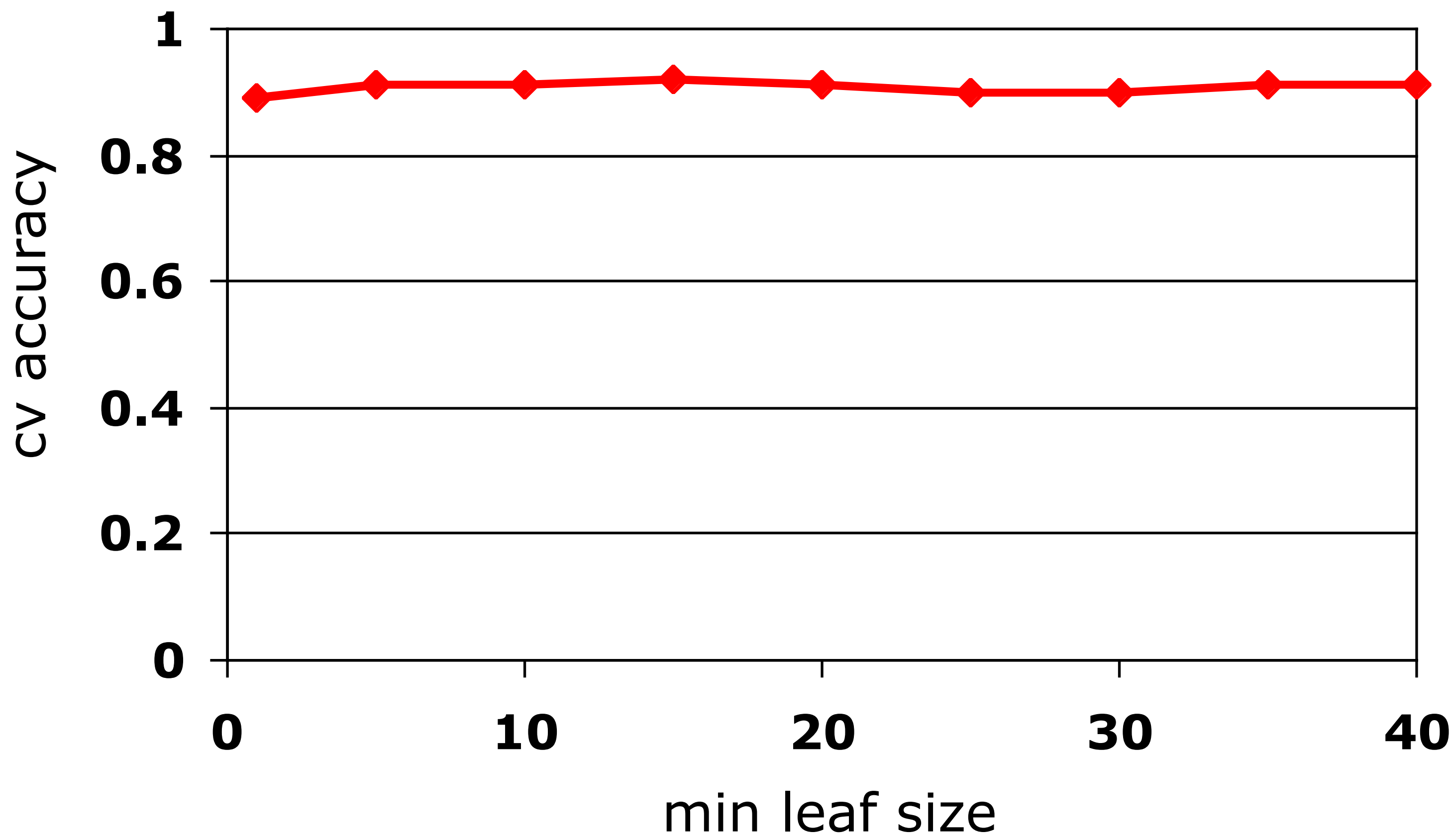
Heart Disease



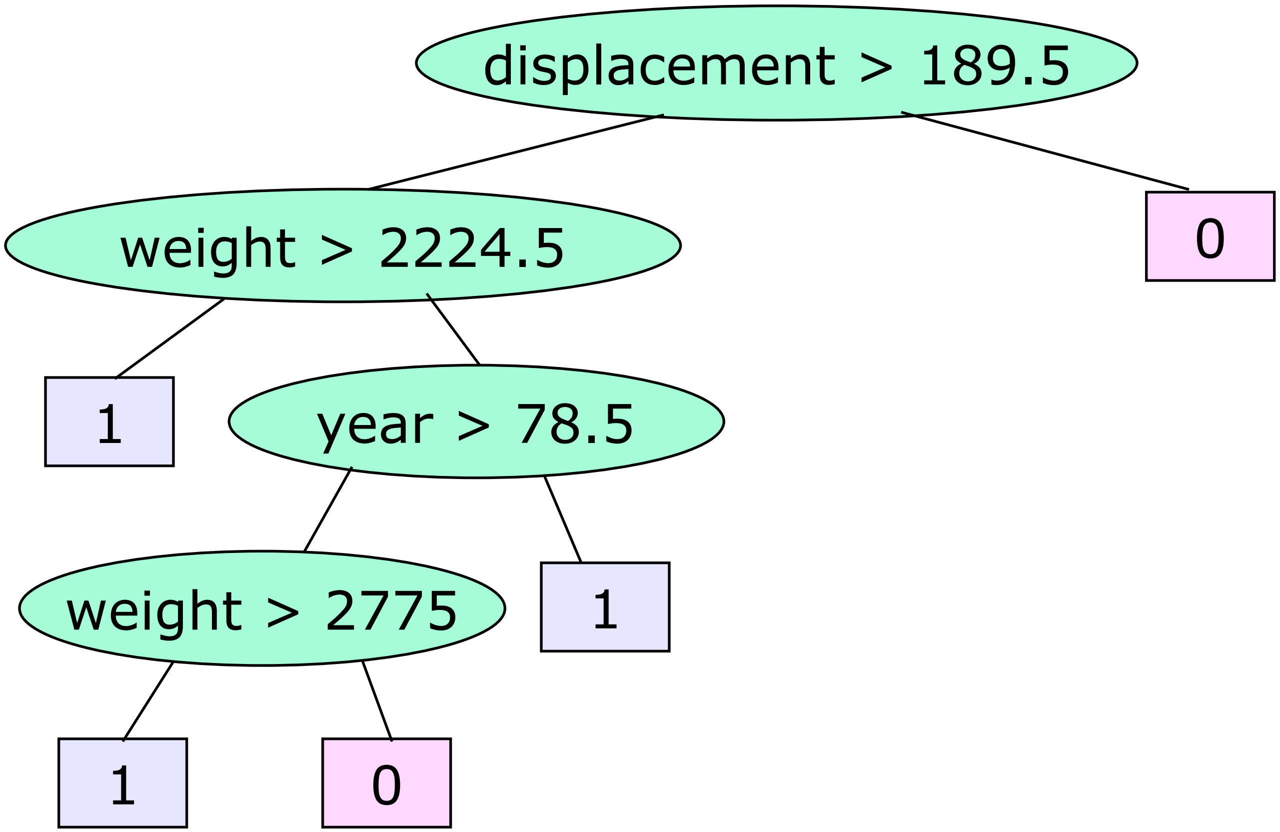
thal = 1: normal exercise thallium scintigraphy test
ca = 0: no vessels colored by fluoroscopy
exang: exercise induced angina
oldpk: feature of cardiogram

Auto MPG

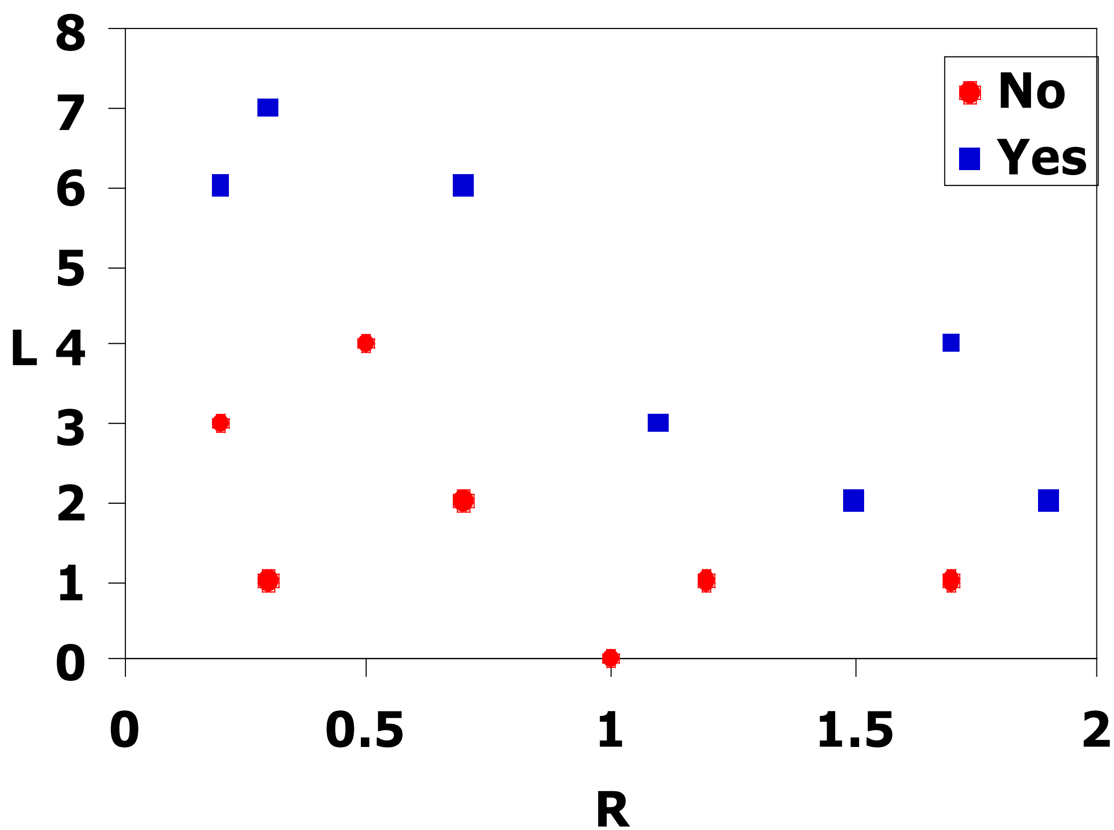
- Performance (.91) essentially the same as nearest neighbor



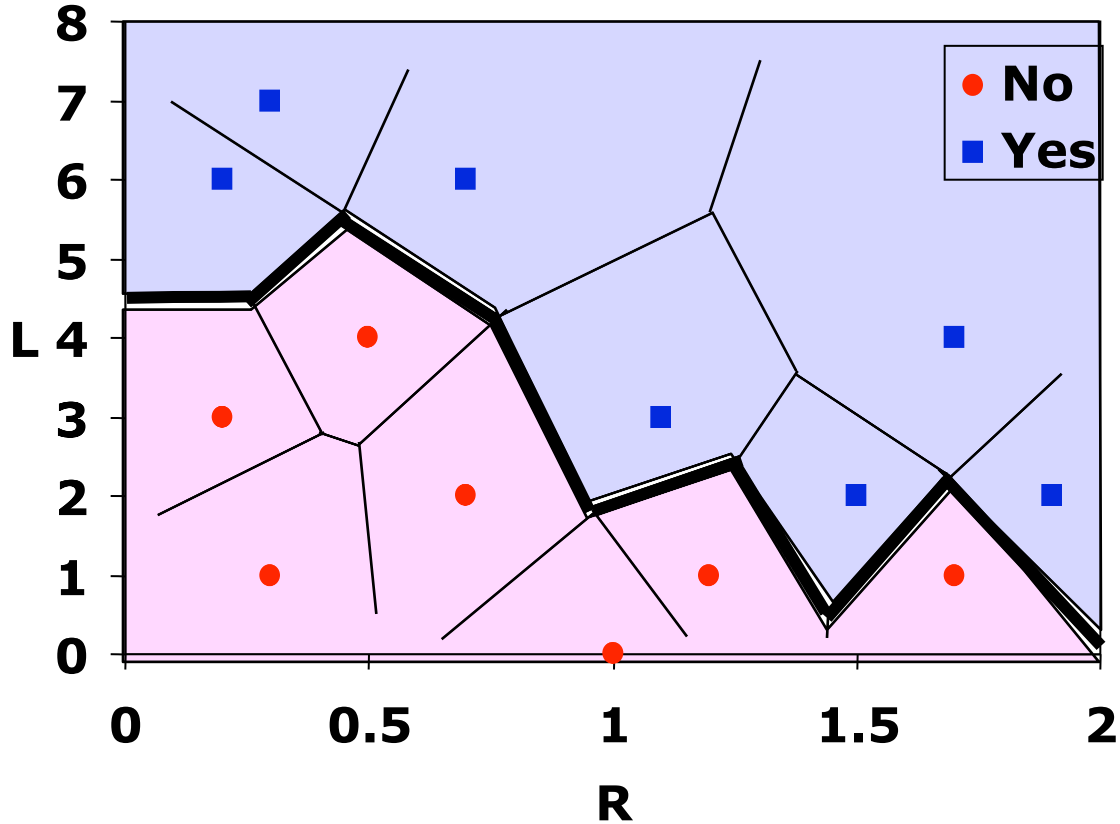
More than 22 MPG?



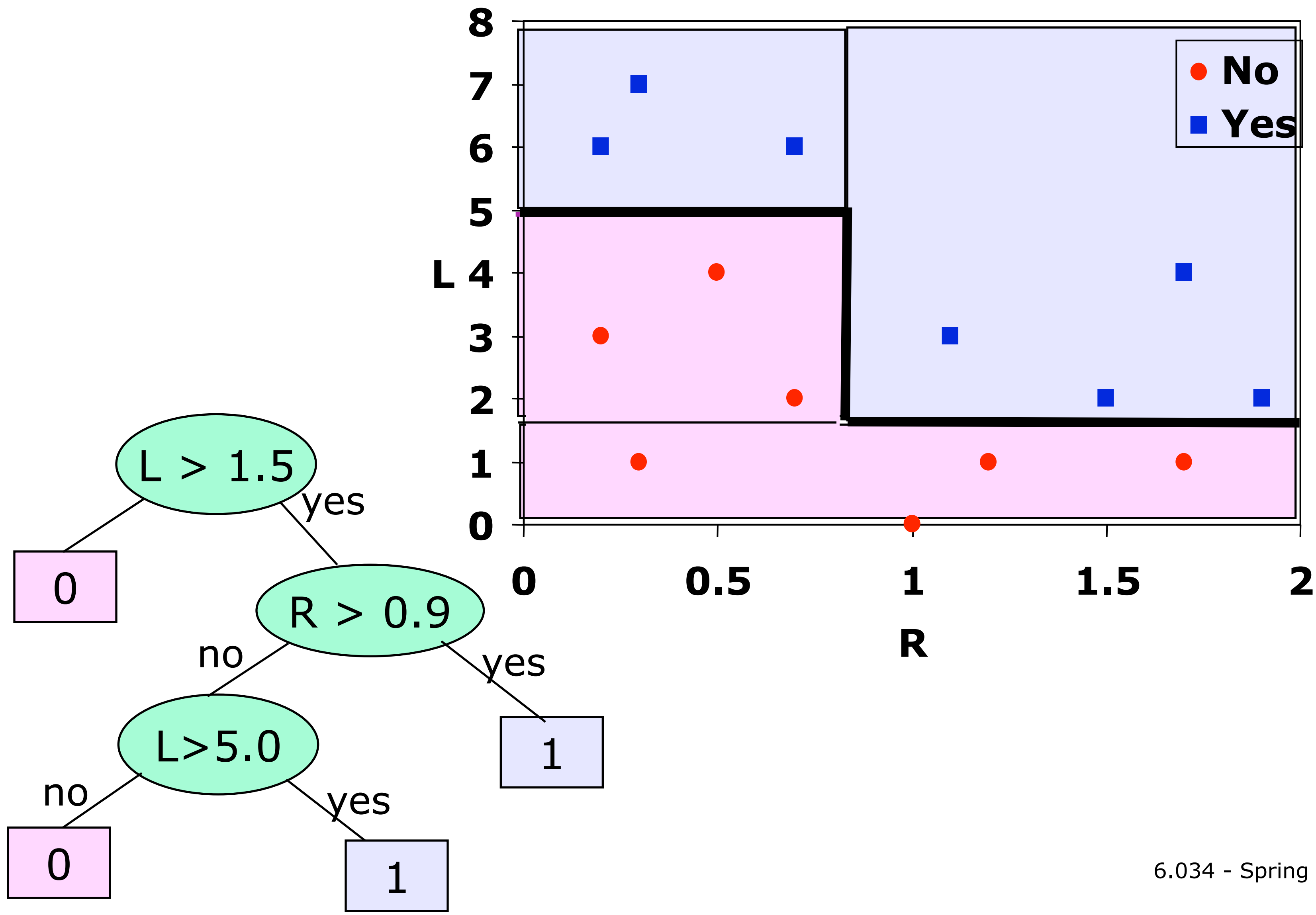
Bankruptcy Example



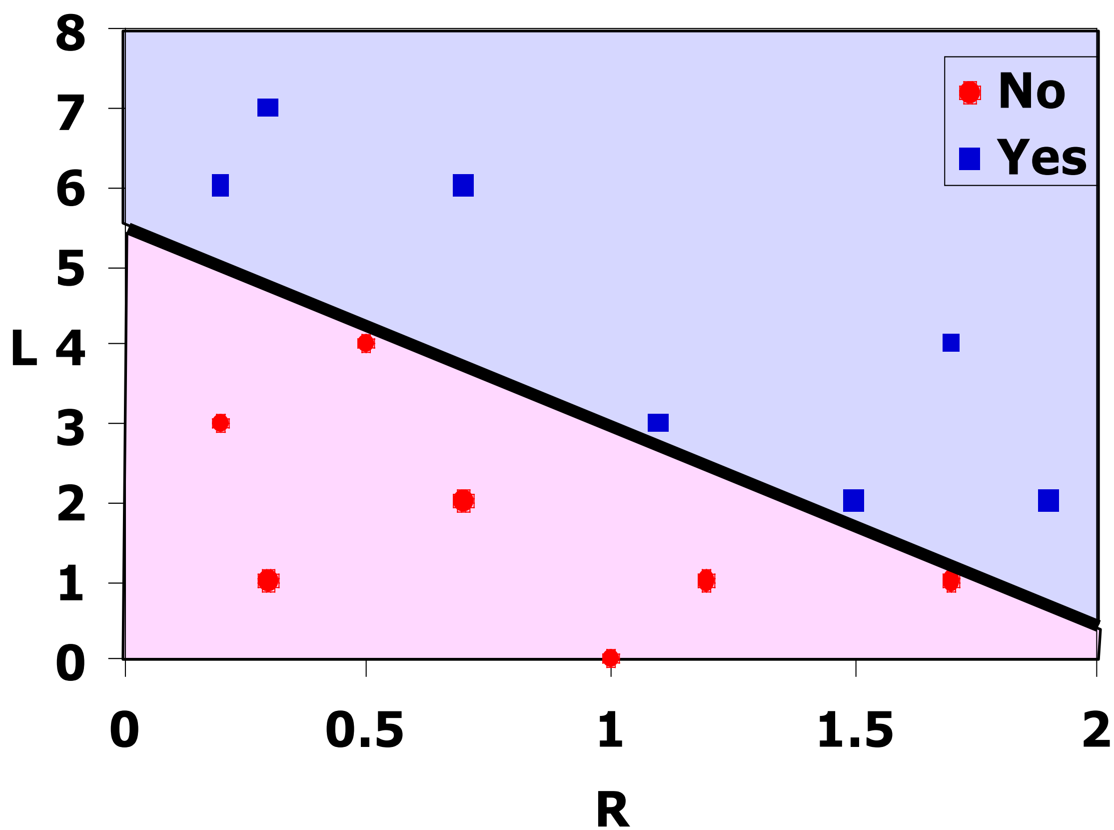
1-Nearest Neighbor Hypothesis



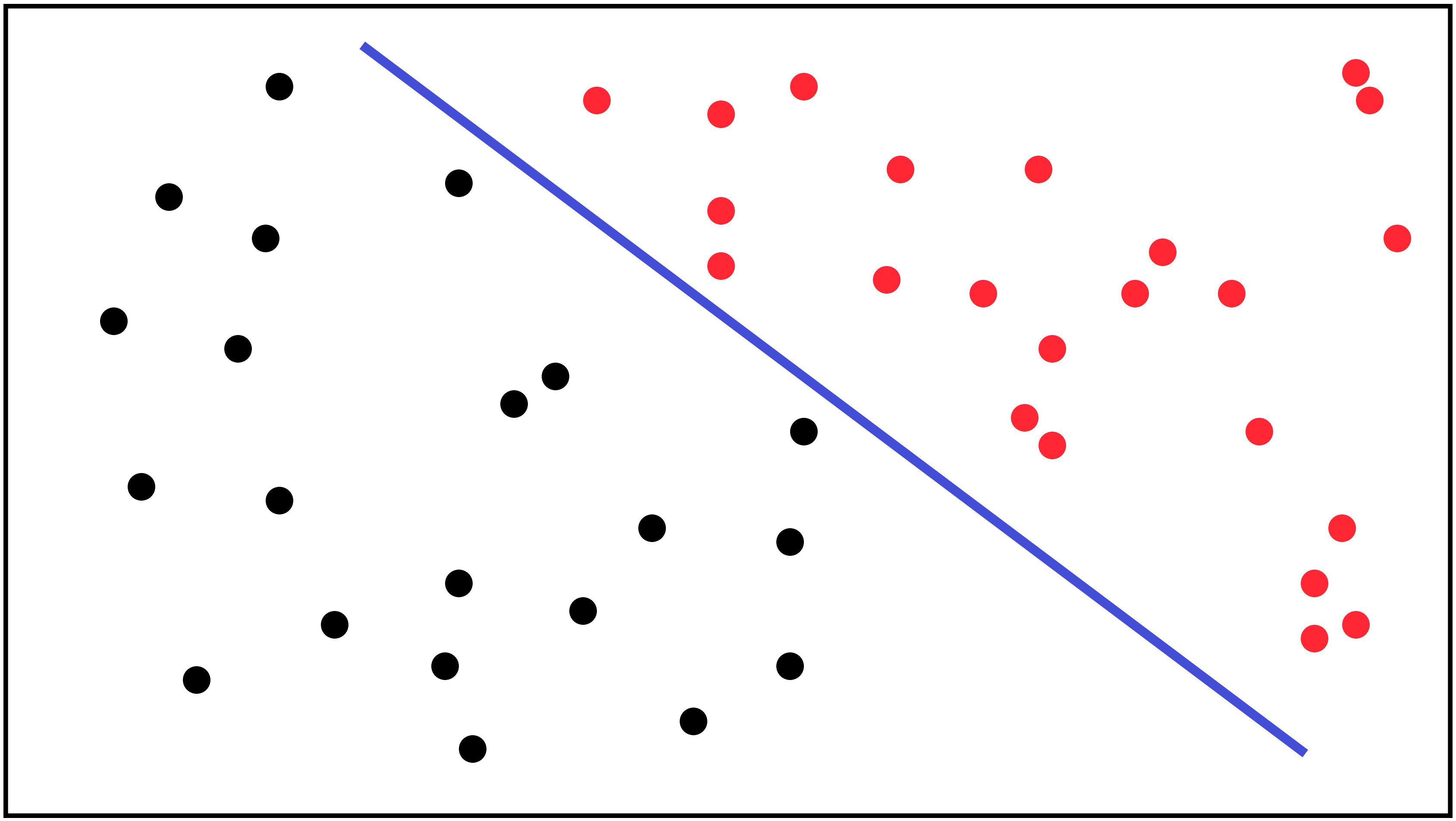
Decision Tree Hypothesis



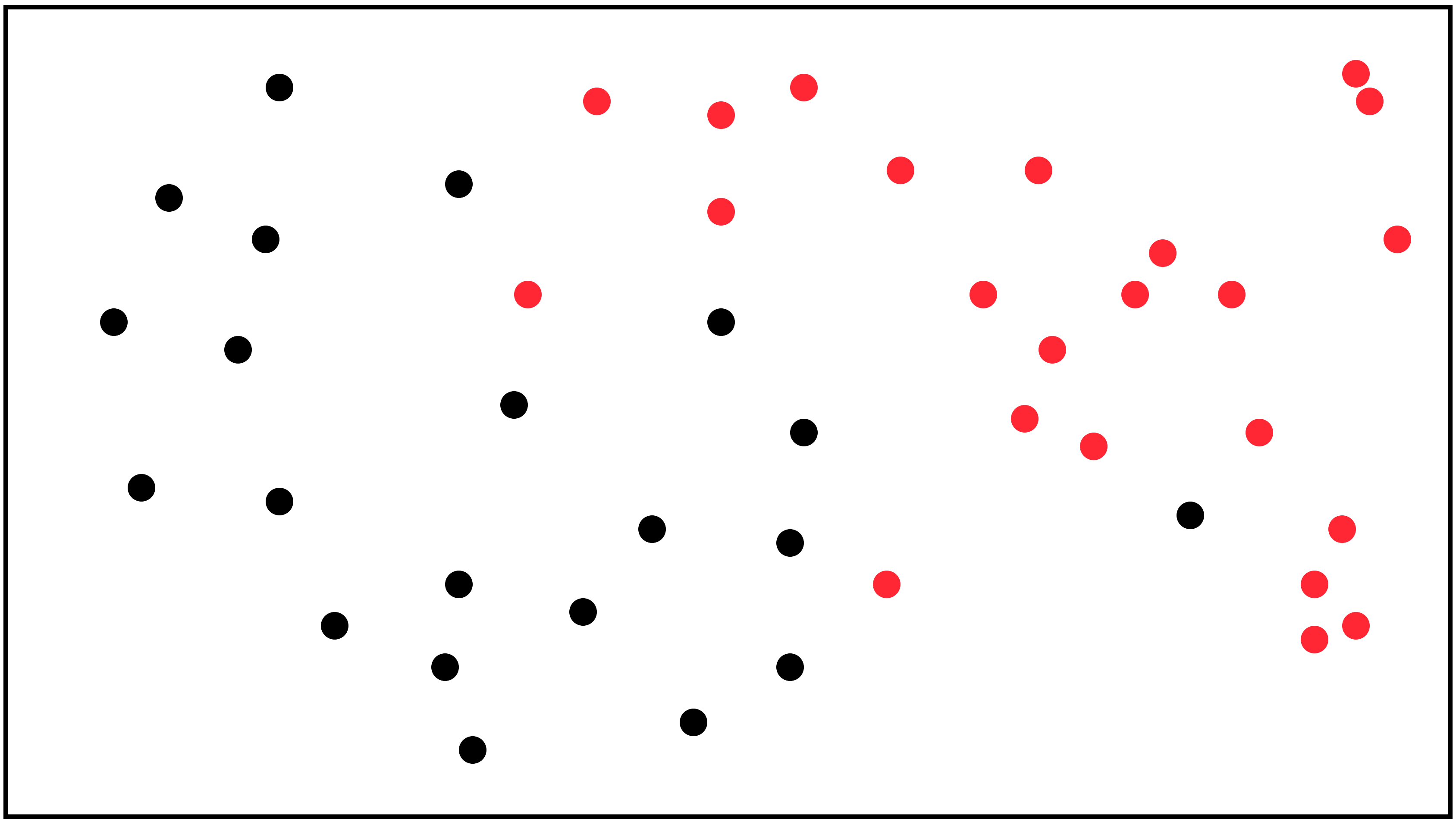
Linear Hypothesis



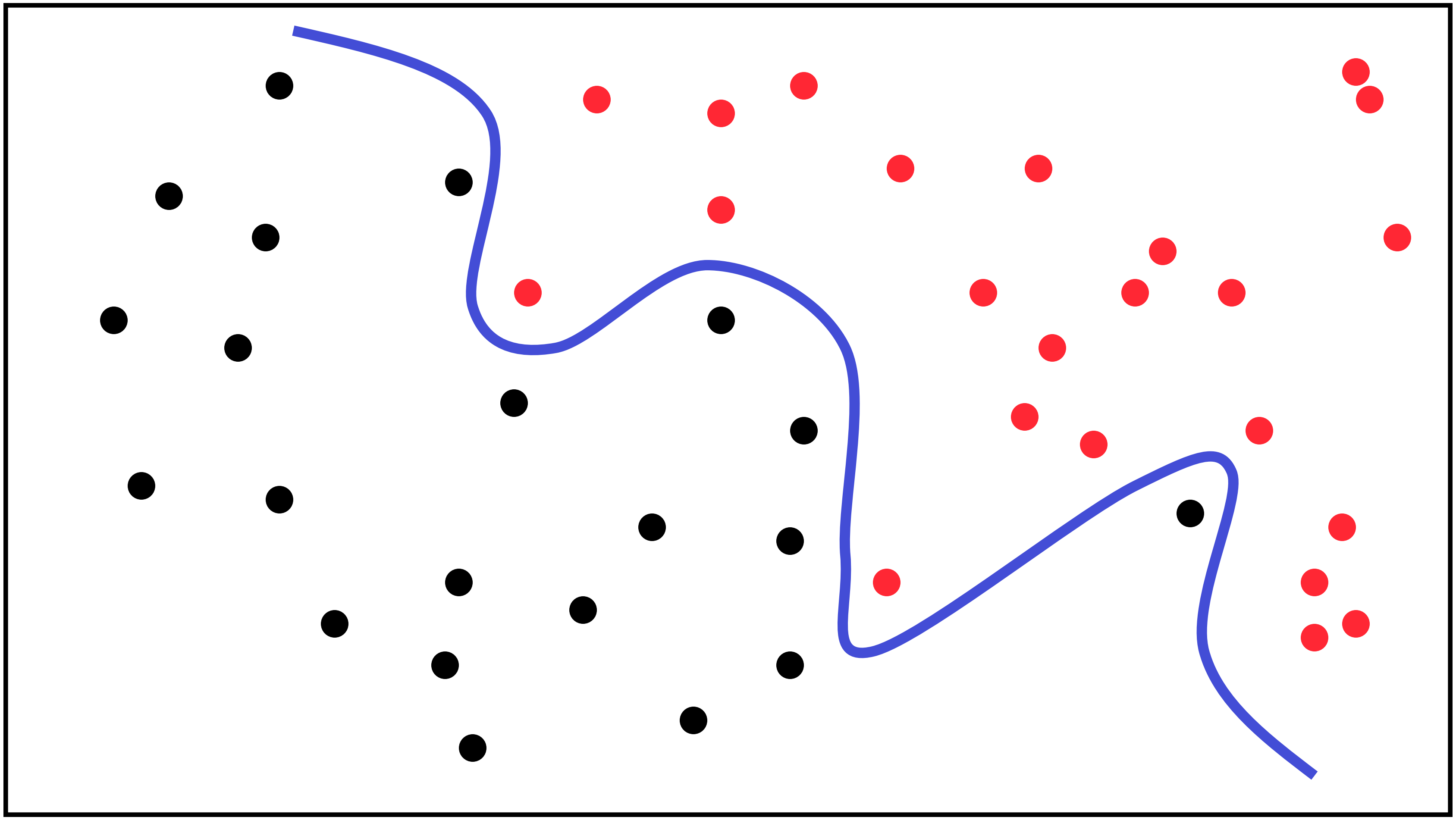
Linearly Separable



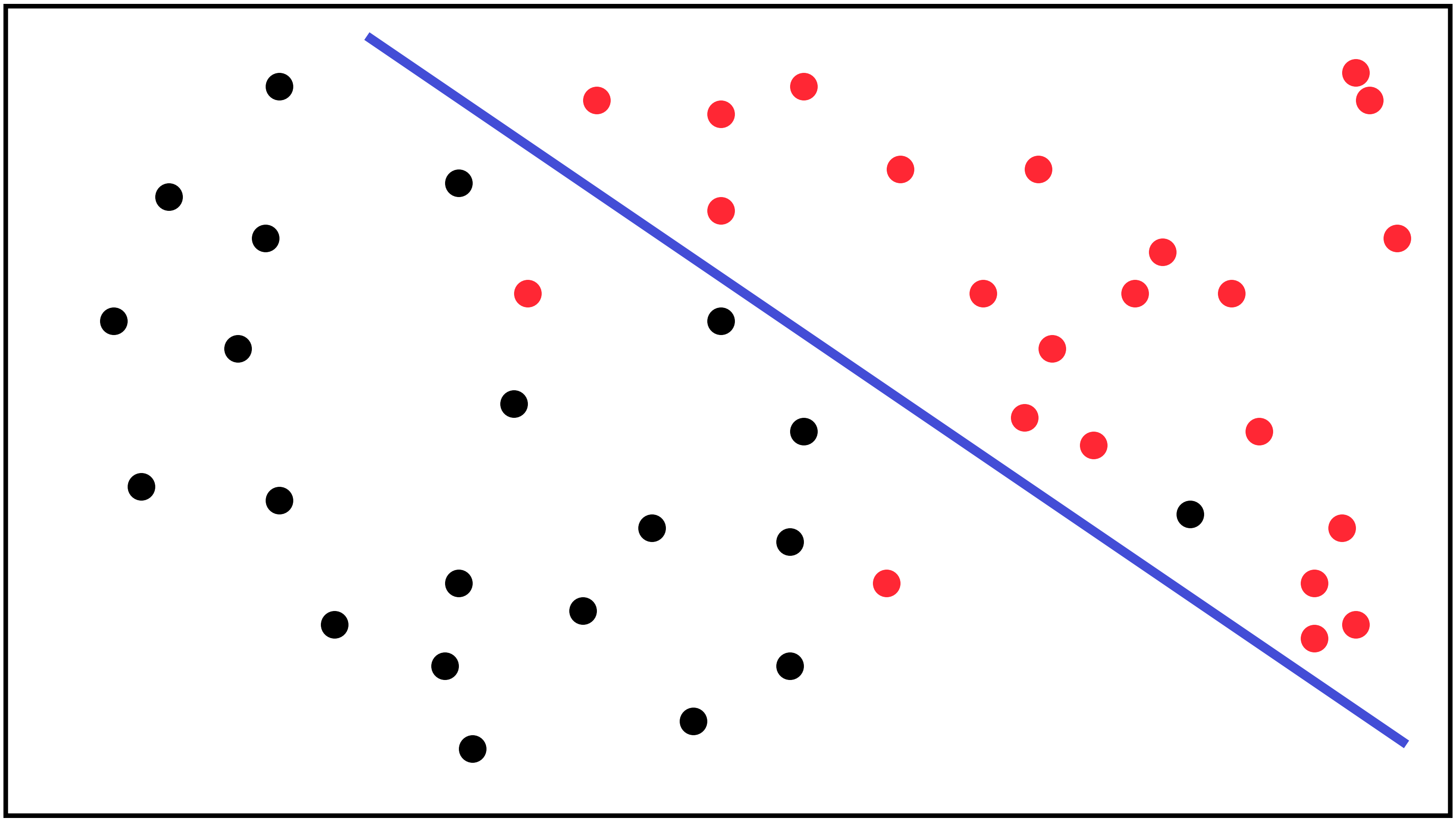
Not Linearly Separable



Not Linearly Separable



Not Linearly Separable



Linear Hypothesis Class

- Equation of a hyperplane in the feature space

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$\sum_{j=1}^{n} w_j x_j + b = 0$$

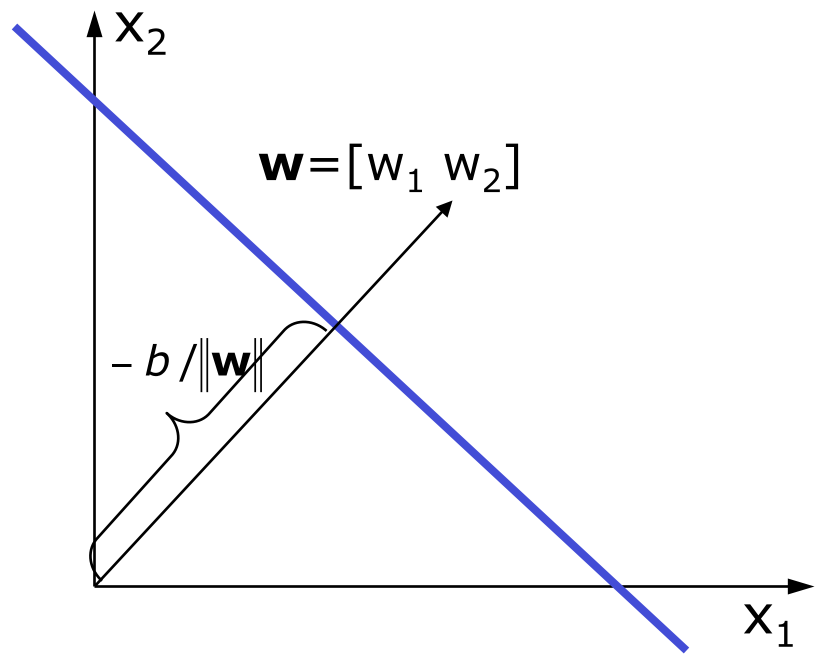
- **w**, b are to be learned

Linear Hypothesis Class

- Equation of a hyperplane in the feature space

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$\sum_{j=1}^{n} w_j x_j + b = 0$$



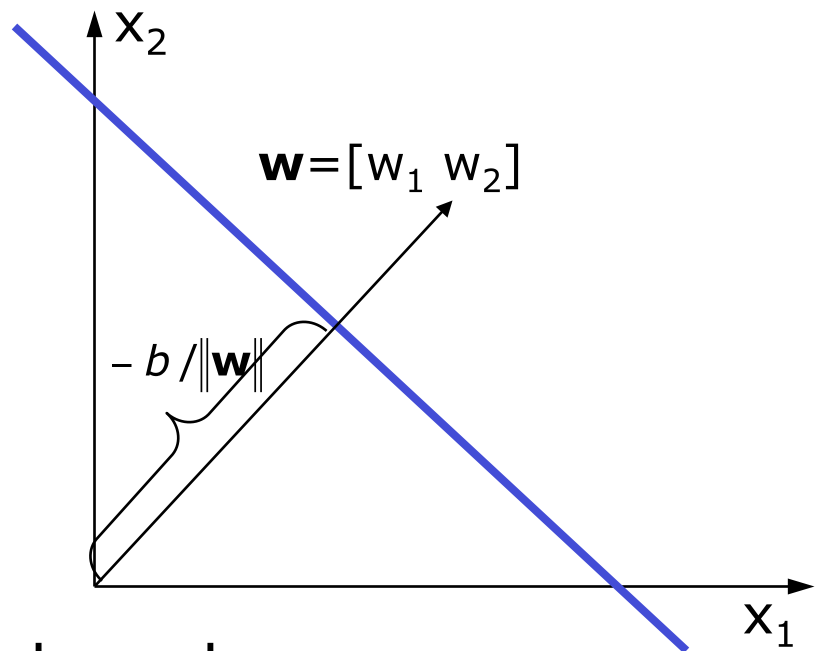
- **w**, b are to be learned

Linear Hypothesis Class

- Equation of a hyperplane in the feature space

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$\sum_{j=1}^{n} w_j x_j + b = 0$$



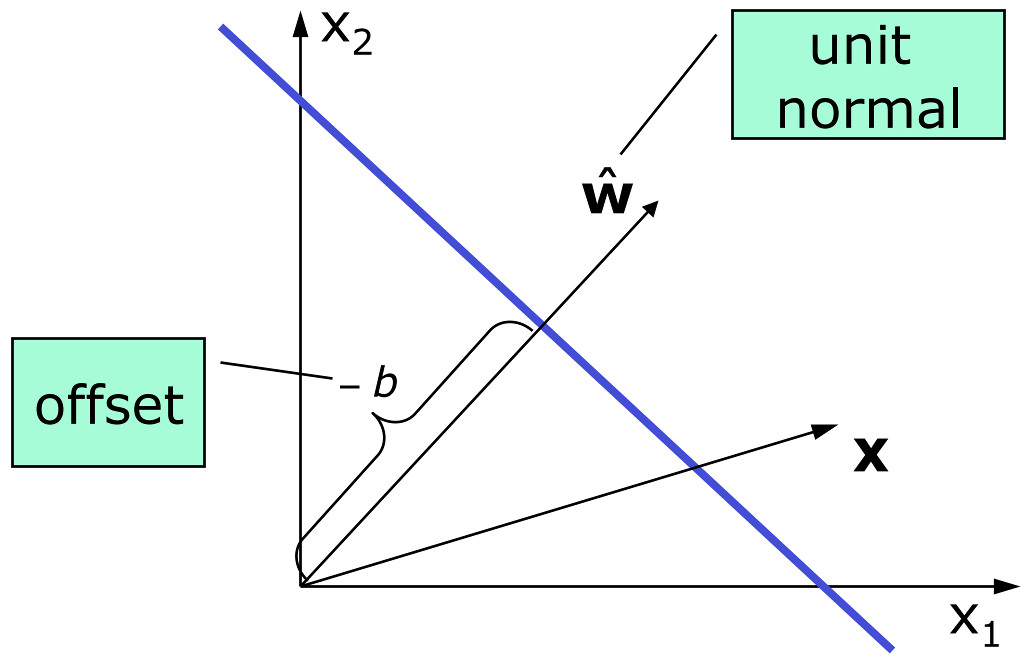
- **w**, b are to be learned

- A useful trick: let x_0=1 and w_0=b

$$\overline{\mathbf{w}} \cdot \overline{\mathbf{x}} = 0$$

$$\sum_{j=0}^{n} w_j x_j = 0$$

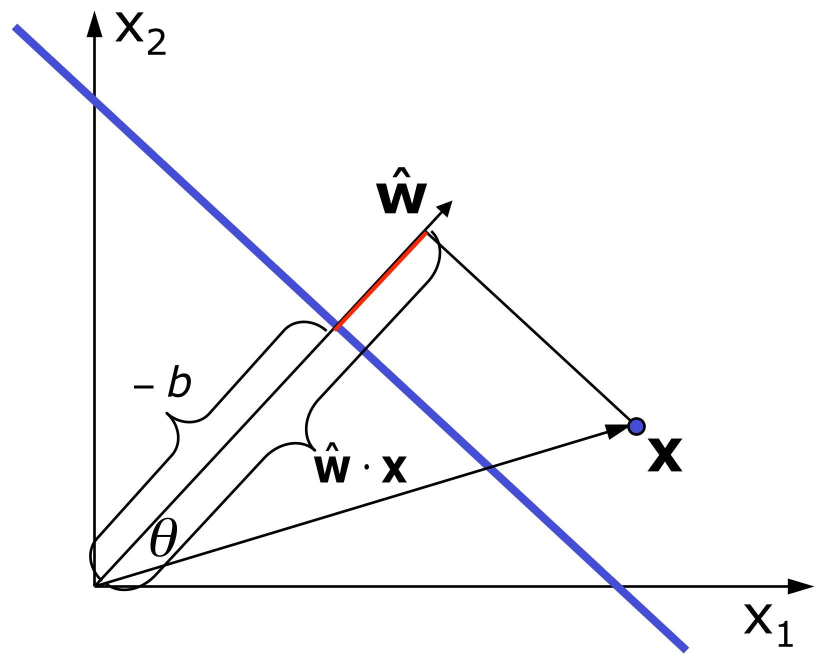
Hyperplane: Geometry



Hyperplane: Geometry

$$\hat{\mathbf{w}} \cdot \mathbf{x} + b$$

signed perpendicular distance of point **x** to hyperplane.

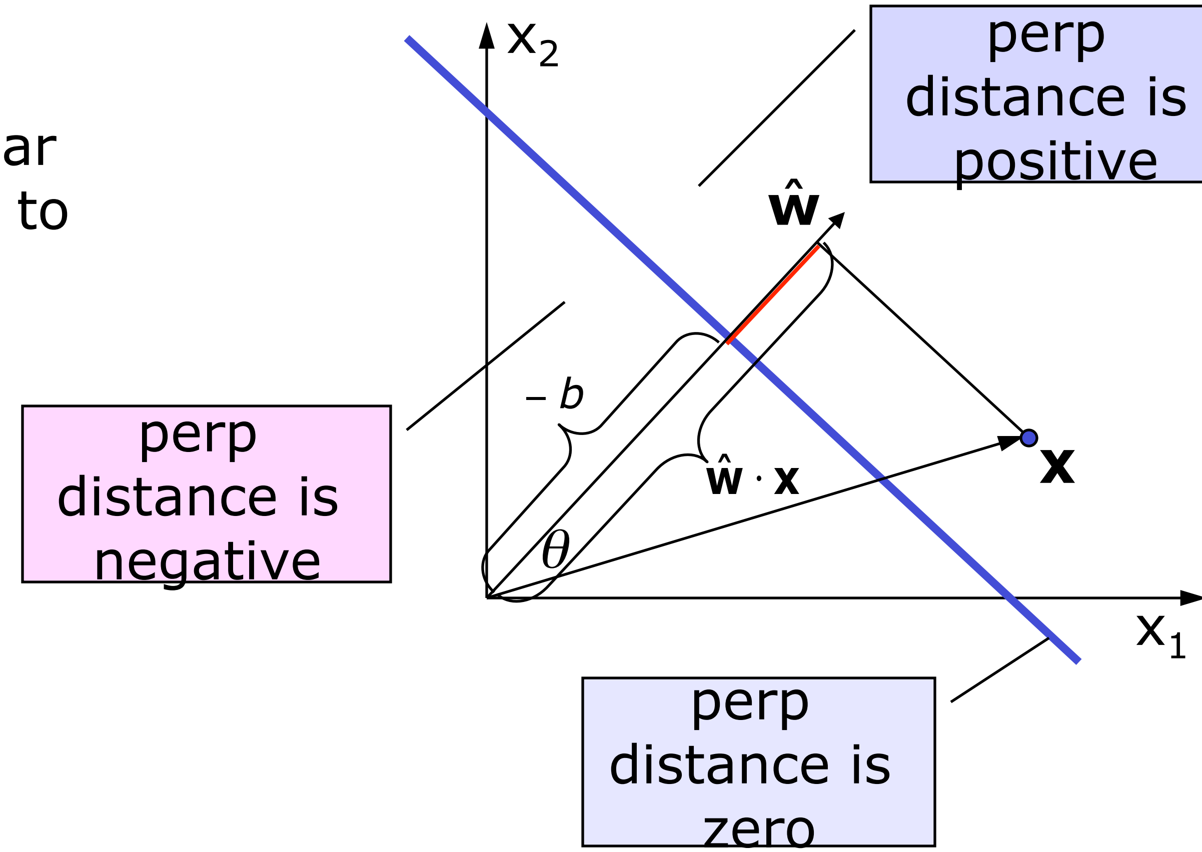


recall: $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|\|\mathbf{b}\| \cos\theta$

Hyperplane: Geometry

$$\hat{\mathbf{w}} \cdot \mathbf{x} + b$$

signed perpendicular distance of point **x** to hyperplane.

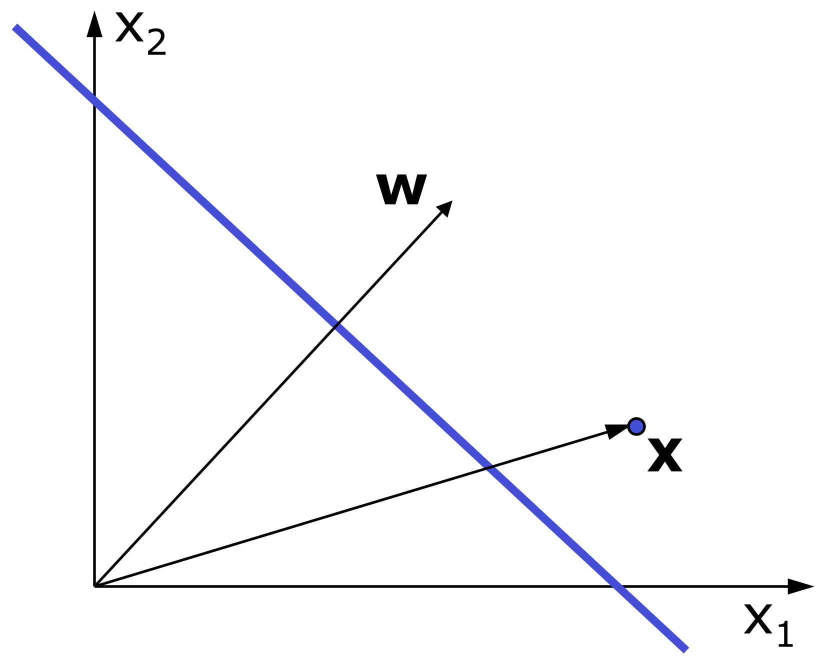


recall: $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos\theta$

Linear Classifier

$$h(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x} + b) \equiv sign(\overline{\mathbf{w}} \cdot \overline{\mathbf{x}})$$

outputs +1 or -1

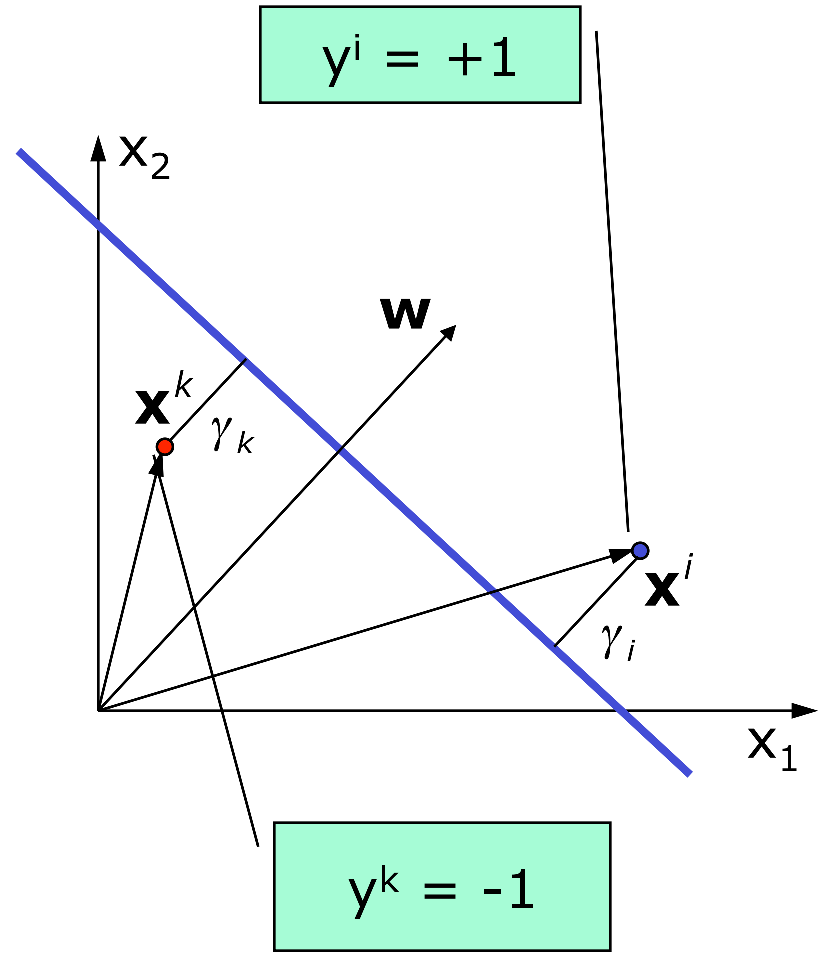


Linear Classifier

Margin:

$$\gamma_i \equiv y^i(\mathbf{w} \cdot \mathbf{x}^i + b) \equiv y^i \overline{\mathbf{w}} \cdot \overline{\mathbf{x}}^i$$

proportional to perpendicular distance of point $\mathbf{x}^i$ to hyperplane.



$\gamma_i > 0$: point is correctly classified (sign of distance = y^i)

$\gamma_i < 0$: point is incorrectly classified (sign of distance ≠ y^i)

Perceptron Algorithm

Rosenblatt, 1956

- Pick initial weight vector (including b), e.g. [0 … 0]
- Repeat until all points correctly classified
  - Repeat for each point
    - Calculate margin ($y^i \overline{\mathbf{w}\mathbf{x}}^i$) for point i
    - If margin > 0, point is correctly classified
    - Else change weights to increase margin; change in weight proportional to $y^i \overline{\mathbf{x}}^i$

Perceptron Algorithm
Rosenblatt, 1956

- Pick initial weight vector (including b), e.g. [0 … 0]
- Repeat until all points correctly classified
  - Repeat for each point
    - Calculate margin ($y^i \overline{\mathbf{w}\mathbf{x}^i}$) for point i
    - If margin > 0, point is correctly classified
    - Else change weights to increase margin; change in weight proportional to $y^i \overline{\mathbf{x}}^i$

- Note that, if $y^i = 1$
  if $x_j^i > 0$ then w_j increased (increases margin)
  if $x_j^i < 0$ then w_j decreased (increases margin)
- And, similarly for $y^i = -1$

Perceptron Algorithm

Rosenblatt, 1956

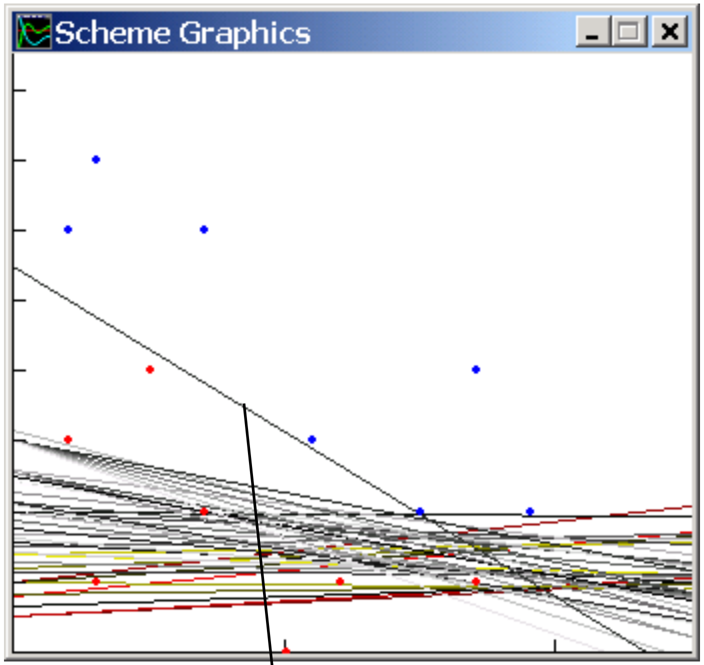
- **Pick initial weight vector (including b), e.g. [0 … 0]**
- **Repeat until all points correctly classified**
  - **Repeat for each point**
    - **Calculate margin ($y^i \overline{\mathbf{w}\mathbf{x}}^i$) for point i**
    - **If margin > 0, point is correctly classified**
    - **Else change weights to increase margin; change in weight proportional to $y^i \overline{\mathbf{x}}^i$**

- Note that, if $y^i = 1$
  if $x_j^i > 0$ then w_j increased (increases margin)
  if $x_j^i < 0$ then w_j decreased (increases margin)
- And, similarly for $y^i = -1$
- Guaranteed to find separating hyperplane if one exists
- Otherwise, data are not linearly separable, loops forever

Perceptron Algorithm
Bankruptcy Data

rate η = 0.1

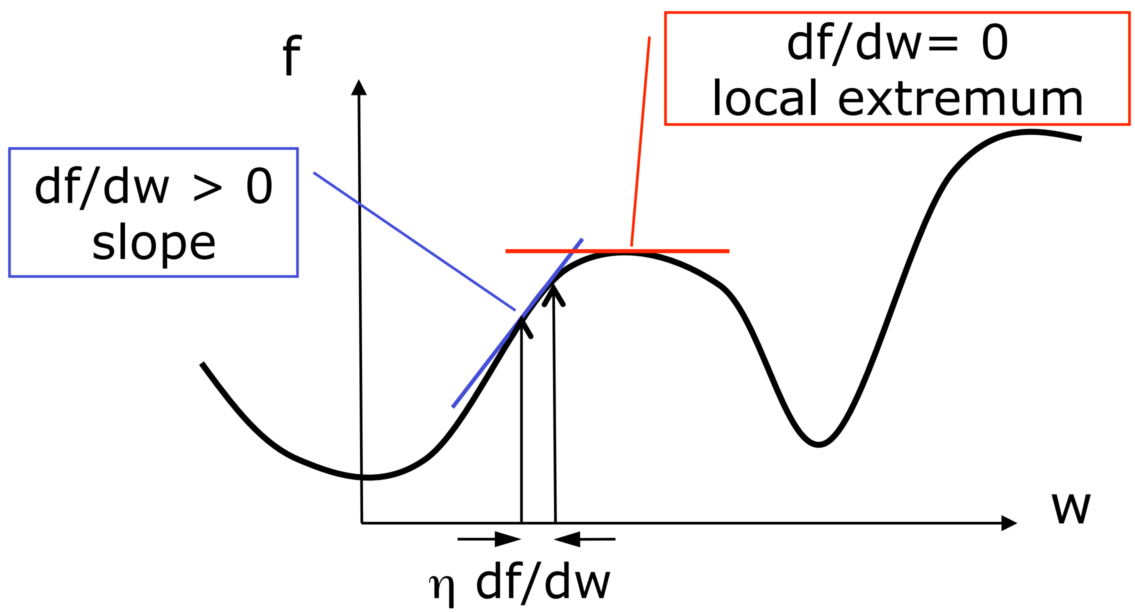
Initial Guess:
w=[0.0 0.0 0.0]



Final Answer:
w=[-2.2 0.94 0.4]

Gradient Ascent

- Why pick $y^i \overline{\mathbf{x}}^i$ as increment to weights?
- To maximize scalar function of one variable f(w)
  - Pick initial w
  - Change w to w + η df/dw   (η > 0, small)
  - until f stops changing (df/dw $\approx$ 0)



Gradient Ascent/Descent

$$\nabla_{\mathbf{w}} f = \left[\frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_n} \right]$$

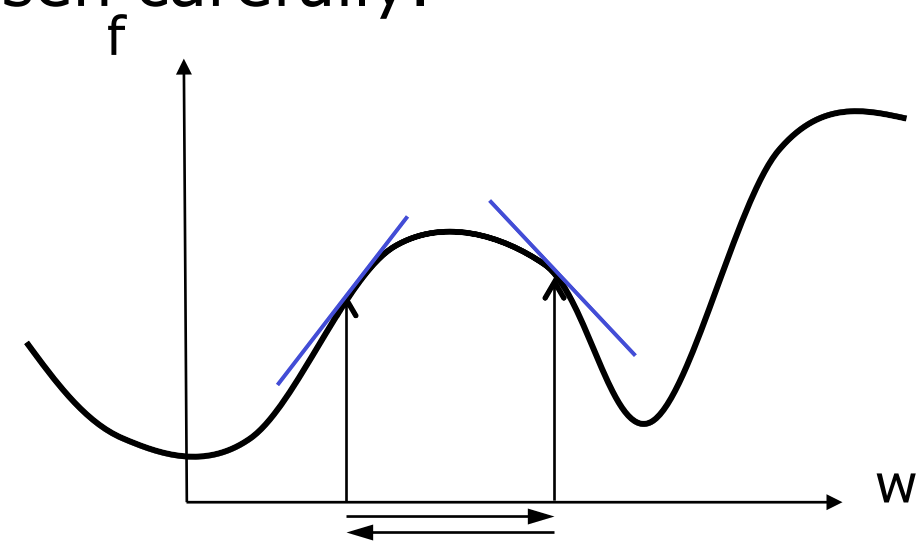
- To maximize f(**w**)
  - Pick initial **w**
  - Change **w** to **w** + $\eta \nabla_{\mathbf{w}} f$ ($\eta > 0$, small)
  - until f stops changing ($\nabla_{\mathbf{w}} f \approx 0$)
- Finds local maximum; global maximum if function is globally convex.

Gradient Ascent/Descent

- To maximize f(**w**)

$$\nabla_{\mathbf{w}} f = \left[\frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_n} \right]$$

  - Pick initial **w**
  - Change **w** to **w** + $\eta \nabla_{\mathbf{w}} f$ ($\eta > 0$, small)
  - until f stops changing ($\nabla_{\mathbf{w}} f \approx 0$)
- Finds local maximum; global maximum if function is globally convex
- Rate (η) has to be chosen carefully.
  - Too small – slow convergence
  - Too big – oscillation



Perceptron Training via Gradient Descent

- Maximize sum of margins of misclassified points

$$f(\mathbf{w}) = \sum_{i \text{ misclassified}} y^i \overline{\mathbf{wx}}^i$$

$$\nabla_{\mathbf{w}} f = \sum_{i \text{ misclassified}} y^i \overline{\mathbf{x}}^i$$

Perceptron Training via Gradient Descent

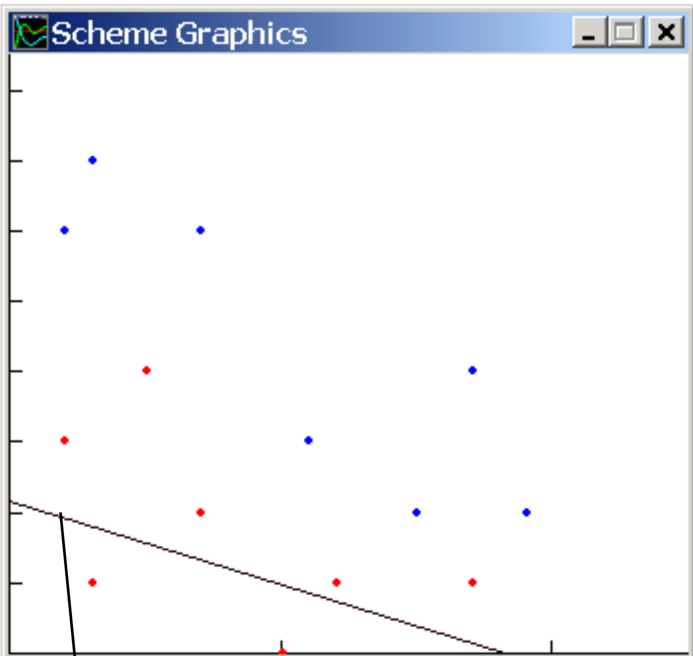
- Maximize sum of margins of misclassified points

$$f(\mathbf{w}) = \sum_{i \text{ misclassified}} y^i \overline{\mathbf{w}\mathbf{x}}^i$$

$$\nabla_{\mathbf{w}} f = \sum_{i \text{ misclassified}} y^i \overline{\mathbf{x}}^i$$

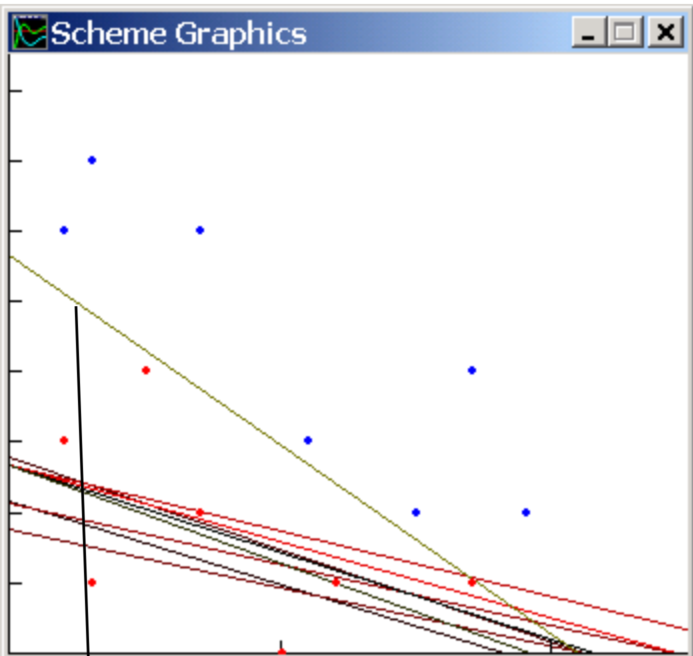
- Off-line training: Compute gradient as sum over all training points.
- On-line training: Approximate gradient by one of the terms in the sum: $y^i \overline{\mathbf{x}}^i$

Perceptron Algorithm
Bankruptcy Data



rate η = 0.1

w_0	w_1	w_2
-1.0	1.00	1.0
-1.3	0.71	0.6
-1.4	0.66	0.5
-1.4	0.66	0.8
-1.5	0.61	0.7
-1.6	0.56	0.6
-1.6	0.65	0.6
-1.6	0.74	0.6
-1.6	0.83	0.6
-1.7	0.81	0.3



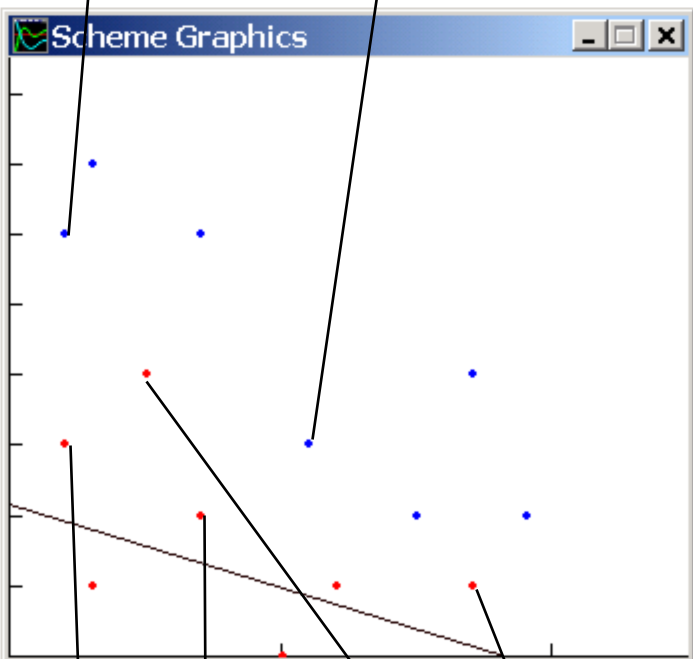
Initial Guess:
w=[-1.0 1.0 1.0]

Final Answer:
w=[-1.7 0.81 0.3]

Perceptron Algorithm
Bankruptcy Data

5x(0.2 6)

3x(1.1 3)



-9x(0.2 3)

-4x(0.5 4)

-1x(0.7 2)

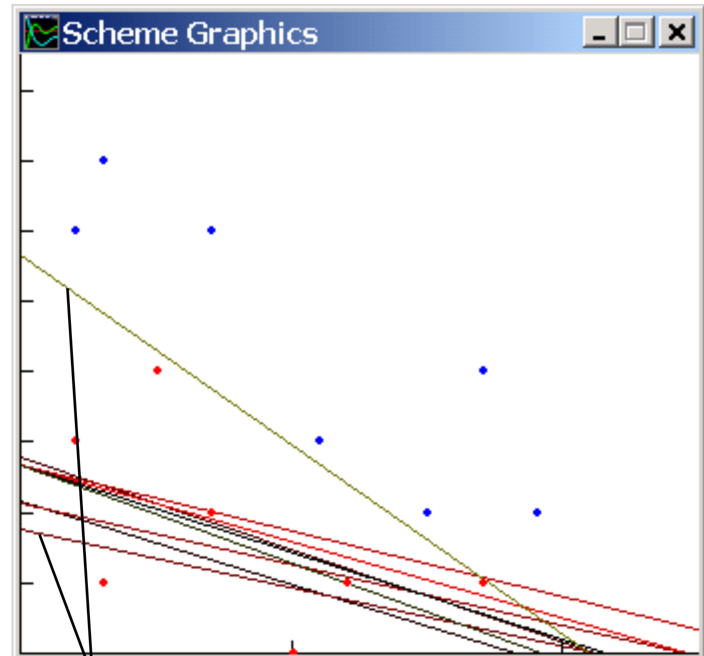
-1x(1.7 1)

rate η = 0.1

$$
\begin{array}{r}
5 \ \mathbf{x} \ (1.0 \ 0.2 \ \ 6.0) \\
3 \ \mathbf{x} \ (1.0 \ 1.1 \ \ 3.0) \\
-9 \ \mathbf{x} \ (1.0 \ 0.2 \ \ 3.0) \\
-1 \ \mathbf{x} \ (1.0 \ 0.7 \ \ 2.0) \\
-4 \ \mathbf{x} \ (1.0 \ 0.5 \ \ 4.0) \\
-1 \ \mathbf{x} \ \underline{(1.0 \ 1.7 \ \ 1.0)} \\
\end{array}
$$

(-7.0 -1.9 -7.0)

x 0.1 =
(-0.7 -0.19 -0.7)



Initial Guess:
w=[-1.0 1.0 1.0]

Final Answer:
w=[-1.7 0.81 0.3]

Dual Form

Assume initial weights are 0; rate=η>0

5 **x** (1.0 0.2 6.0)
3 **x** (1.0 1.1 3.0)
-9 **x** (1.0 0.2 3.0)
-1 **x** (1.0 0.7 2.0)
-4 **x** (1.0 0.5 4.0)
-1 **x** (1.0 1.7 1.0)

(-7.0 -1.9 -7.0) **x** 0.1
=
(-0.7 -0.19 -0.7)

$\Rightarrow$

$\alpha_1 y_1 \bar{\mathbf{x}}_1$
$\alpha_3 y_3 \bar{\mathbf{x}}_3$
$\alpha_4 y_4 \bar{\mathbf{x}}_4$
$\alpha_7 y_7 \bar{\mathbf{x}}_7$
$\alpha_8 y_8 \bar{\mathbf{x}}_8$
$\alpha_{11} y_{11} \bar{\mathbf{x}}_{11}$

$$\bar{\mathbf{w}} = \eta \sum_{i=1}^{m} \alpha_i y_i \bar{\mathbf{x}}_i$$

α_i is count of mistakes on point i during training

Dual Form

Assume initial weights are 0;   rate=η>0

$$\begin{array}{rl}
5 \,\mathbf{x} & (1.0\ 0.2\ \ 6.0) \\
3 \,\mathbf{x} & (1.0\ 1.1\ \ 3.0) \\
-9 \,\mathbf{x} & (1.0\ 0.2\ \ 3.0) \\
-1 \,\mathbf{x} & (1.0\ 0.7\ \ 2.0) \\
-4 \,\mathbf{x} & (1.0\ 0.5\ \ 4.0) \\
-1 \,\mathbf{x} & (1.0\ 1.7\ \ 1.0) \\
\hline
\end{array}$$

$$(-7.0\ -1.9\ -7.0)\ \mathbf{x}\ \ 0.1$$
$$=$$
$$(-0.7\ -0.19\ -0.7)$$

$$\begin{array}{c}
\alpha_1 y^1 \overline{\mathbf{x}}^1 \\
\alpha_3 y^3 \overline{\mathbf{x}}^3 \\
\alpha_4 y^4 \overline{\mathbf{x}}^4 \\
\alpha_7 y^7 \overline{\mathbf{x}}^7 \\
\alpha_8 y^8 \overline{\mathbf{x}}^8 \\
\alpha_{11} y^{11} \overline{\mathbf{x}}^{11} \\
\hline
\end{array}$$

$$\overline{\mathbf{w}} = \eta \sum_{i=1}^{m} \alpha_i y_i \overline{\mathbf{x}}_i$$

α_i is count of mistakes on point i during training

η just scales answer, set to 1

$$h(\mathbf{x}) = sign(\overline{\mathbf{w}} \cdot \overline{\mathbf{x}}) = sign(\sum_{i=1}^{m} \alpha_i y^i \overline{\mathbf{x}}^i \cdot \overline{\mathbf{x}})$$

Perceptron Training
Dual Form

- $\alpha = 0$
- Repeat until all points correctly classified
  - Repeat for each point i
    - Calculate margin $\sum_{j=1}^{m} \alpha_j y^j \overline{\mathbf{x}}^j \cdot \overline{\mathbf{x}}^i$
    - If margin > 0, point is correctly classified
    - Else increment α_i
- Return $\overline{\mathbf{w}} = \sum_{j=1}^{m} \alpha_j y^j \overline{\mathbf{x}}^j$
- If data is not linearly separable, the α_i grow without bound