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

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.2</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>0.3</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>0.5</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>0.7</td>
<td>No</td>
</tr>
<tr>
<td>0</td>
<td>1.0</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>1.2</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>1.7</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>0.2</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>0.3</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>0.7</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>1.1</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>1.5</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>1.7</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>1.9</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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)

\[ D(x^i, x^k) = \sqrt{\sum_j (x^i_j - x^k_j)^2} \]
What do we mean by “Nearest”? 

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

\[ D(x^i, x^k) = \sqrt{\sum_j (x^i_j - x^k_j)^2} \]

• 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?
  • $f_1 = \text{weight in pounds}$
  • $f_2 = \text{number of cylinders}$
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?
  • $f_1 = \text{weight in pounds}$
  • $f_2 = \text{number of cylinders}$
• 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:
  $$x' = \frac{x - \bar{x}}{\sigma_x}$$
  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:
  $$x' = \frac{x - \bar{x}}{\sigma_x}$$
  \begin{align*}
  \bar{x} & \quad \text{average} \\
  \sigma_x & \quad \text{standard deviation}
  \end{align*}

• 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:

\[
x' = \frac{x - \bar{x}}{\sigma_x}
\]

  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

\[ D(x^i, x^k) = \sqrt{\sum_j (L^i_j - L^k_j)^2 + (5R^i_j - 5R^k_j)^2} \]
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 $m \times n$ computations
Time and Space

• Learning is fast
• Lookup takes about \( m \times n \) computations
  • storing data in a clever data structure (KD-tree) reduces this, on average, to \( \log(m) \times n \)
Time and Space

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

![Graph showing data points with 'No' and 'Yes' categories.]
• Find the k nearest points
k-Nearest Neighbor

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

![Graph showing cv accuracy vs $k$]
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 $x_j > \text{constant}$
**Numerical Attributes**

- Tests in nodes can be of the form $x_j > \text{constant}$
- Divides the space into axis-aligned rectangles
Numerical Attributes

- Tests in nodes can be of the form $x_j > \text{constant}$
- Divides the space into axis-aligned rectangles
Numerical Attributes

- Tests in nodes can be of the form $x_j > \text{constant}$
- Divides the space into axis-aligned rectangles
Numerical Attributes

• Tests in nodes can be of the form $x_j > \text{constant}$
• 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

<table>
<thead>
<tr>
<th>L&lt;(y)</th>
<th>NL</th>
<th>PL</th>
<th>NR</th>
<th>PR</th>
<th>AE</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5</td>
<td>7</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0.93</td>
</tr>
<tr>
<td>5.0</td>
<td>7</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0.74</td>
</tr>
<tr>
<td>3.5</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>0.85</td>
</tr>
<tr>
<td>2.5</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>0.86</td>
</tr>
<tr>
<td>1.5</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td>0.63</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>7</td>
<td>0.93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># neg to left</th>
<th># pos to left</th>
<th># neg to right</th>
<th># pos to right</th>
<th>AE</th>
<th>R&lt;(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.40</td>
<td>0.60</td>
<td>0.85</td>
<td>1.05</td>
<td>1.15</td>
</tr>
<tr>
<td>0.94</td>
<td>0.98</td>
<td>0.92</td>
<td>0.94</td>
<td>1.35</td>
<td>1.60</td>
</tr>
<tr>
<td>0.92</td>
<td>0.98</td>
<td>0.92</td>
<td>0.98</td>
<td>1.80</td>
<td></td>
</tr>
</tbody>
</table>

The graph illustrates the distribution of Yes and No outcomes based on the values of L<\(y\), AE, and R<\(x\). Points represent the outcomes, with red circles indicating No and blue squares indicating Yes.
Bankruptcy Example

<table>
<thead>
<tr>
<th>L&lt;y</th>
<th>NL</th>
<th>PL</th>
<th>NR</th>
<th>PR</th>
<th>AE</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5</td>
<td>7</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0.93</td>
</tr>
<tr>
<td>5.0</td>
<td>7</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0.74</td>
</tr>
<tr>
<td>3.5</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>0.85</td>
</tr>
<tr>
<td>2.5</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>0.86</td>
</tr>
<tr>
<td>1.5</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td>0.63</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>7</td>
<td>0.93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AE</th>
<th>1.00</th>
<th>1.00</th>
<th>0.98</th>
<th>0.98</th>
<th>0.94</th>
<th>0.98</th>
<th>0.92</th>
<th>0.98</th>
<th>0.92</th>
</tr>
</thead>
<tbody>
<tr>
<td>R&lt;x</td>
<td>0.25</td>
<td>0.40</td>
<td>0.60</td>
<td>0.85</td>
<td>1.05</td>
<td>1.15</td>
<td>1.35</td>
<td>1.60</td>
<td>1.80</td>
</tr>
</tbody>
</table>
# Bankruptcy Example

## Table

<table>
<thead>
<tr>
<th>L&lt;(y)</th>
<th>NL</th>
<th>PL</th>
<th>NR</th>
<th>PR</th>
<th>AE</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5</td>
<td>7</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0.93</td>
</tr>
<tr>
<td>5.0</td>
<td>7</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0.74</td>
</tr>
<tr>
<td>3.5</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>0.85</td>
</tr>
<tr>
<td>2.5</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>0.86</td>
</tr>
<tr>
<td>1.5</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td>0.63</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>7</td>
<td>0.93</td>
</tr>
</tbody>
</table>

## Diagram

- L > 1.5
  - no
  - yes

## Chart

- Red circles represent No
- Blue squares represent Yes

## Table

<table>
<thead>
<tr>
<th>AE</th>
<th>1.00</th>
<th>1.00</th>
<th>0.98</th>
<th>0.98</th>
<th>0.94</th>
<th>0.98</th>
<th>0.92</th>
<th>0.98</th>
<th>0.92</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>0.25</td>
<td>0.40</td>
<td>0.60</td>
<td>0.85</td>
<td>1.05</td>
<td>1.15</td>
<td>1.35</td>
<td>1.60</td>
<td>1.80</td>
</tr>
</tbody>
</table>
**Bankruptcy Example**

<table>
<thead>
<tr>
<th>L&lt;(y)</th>
<th>NL</th>
<th>PL</th>
<th>NR</th>
<th>PR</th>
<th>AE</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0.83</td>
</tr>
<tr>
<td>5.0</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0.69</td>
</tr>
<tr>
<td>3.5</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>0.85</td>
</tr>
<tr>
<td>2.5</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0.88</td>
</tr>
</tbody>
</table>

0

**R > 0.9**

no

??

yes

<table>
<thead>
<tr>
<th>AD</th>
<th>0.85</th>
<th>0.88</th>
<th>0.79</th>
<th>0.60</th>
<th>0.69</th>
<th>0.76</th>
<th>0.83</th>
</tr>
</thead>
<tbody>
<tr>
<td>R&lt;x</td>
<td>0.25</td>
<td>0.40</td>
<td>0.60</td>
<td>0.90</td>
<td>1.30</td>
<td>1.60</td>
<td>1.80</td>
</tr>
</tbody>
</table>
Bankruptcy Example

<table>
<thead>
<tr>
<th>L&lt;y</th>
<th>NL</th>
<th>PL</th>
<th>NR</th>
<th>PR</th>
<th>AE</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0.81</td>
</tr>
<tr>
<td>5.0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>3.5</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0.54</td>
</tr>
<tr>
<td>2.5</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0.81</td>
</tr>
</tbody>
</table>

If L > 1.5 then yes
If R > 0.9 then yes
If R<x then

<table>
<thead>
<tr>
<th>AE</th>
<th>1.00</th>
<th>0.92</th>
<th>1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>R&lt;x</td>
<td>0.25</td>
<td>0.40</td>
<td>0.60</td>
</tr>
</tbody>
</table>
Bankruptcy Example

<table>
<thead>
<tr>
<th>L&lt;y</th>
<th>NL</th>
<th>PL</th>
<th>NR</th>
<th>PR</th>
<th>AE</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0.81</td>
</tr>
<tr>
<td>5.0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>3.5</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0.54</td>
</tr>
<tr>
<td>2.5</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0.81</td>
</tr>
</tbody>
</table>

R > 0.9

L > 1.5

R < x

AE | 1.00 | 0.92 | 1.00
R<x | 0.25 | 0.40 | 0.60
Bankruptcy Example

- \( R > 0.9 \)
- \( L > 1.5 \)
- \( L > 5.0 \)

- No
- Yes
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 (0.91) essentially the same as nearest neighbor
More than 22 MPG?

- displacement > 189.5
  - weight > 2224.5
    - year > 78.5
      - weight > 2775
        - 1
        - 0
        - 1
    - 0
  - 1
  - 0
Bankruptcy Example

[Graph showing points with 'No' and 'Yes' markers]
1-Nearest Neighbor Hypothesis
Decision Tree Hypothesis

- If \( R > 0.9 \):
  - If \( L > 1.5 \): Yes
  - If \( L \leq 1.5 \): No
- If \( R \leq 0.9 \):
  - If \( L > 5.0 \): Yes
  - If \( L \leq 5.0 \): No

Graphical representation showing decision points and outcomes.

No
Yes
Linearly Separable
Not Linearly Separable
Not Linearly Separable
Not Linearly Separable
Linear Hypothesis Class

• Equation of a hyperplane in the feature space
  \[ \mathbf{w} \cdot \mathbf{x} + b = 0 \]
  \[ \sum_{j=1}^{n} w_j x_j + b = 0 \]

• \( \mathbf{w}, b \) are to be learned
Linear Hypothesis Class

- Equation of a hyperplane in the feature space

\[ \mathbf{w} \cdot \mathbf{x} + b = 0 \]

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

- \( \mathbf{w}, b \) are to be learned
Linear Hypothesis Class

• Equation of a hyperplane in the feature space

\[ \mathbf{w} \cdot \mathbf{x} + b = 0 \]

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

• \( \mathbf{w} \), \( b \) 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

\[ \hat{w} \]

offset

unit normal

\[ x \]

\[ x_1 \]

\[ x_2 \]
\[ \hat{\mathbf{w}} \cdot \mathbf{x} + b \]

signed perpendicular distance of point \( \mathbf{x} \) 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 \( \mathbf{x} \) to hyperplane.

\( \mathbf{w} \cdot \mathbf{x} \)

perp distance is positive

perp distance is negative

perp distance is zero

recall: \[ \mathbf{a} \cdot \mathbf{b} = \| \mathbf{a} \| \| \mathbf{b} \| \cos \theta \]
Linear Classifier

\[ h(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) = \text{sign}(\mathbf{w} \cdot \overline{\mathbf{x}}) \]

outputs +1 or -1
Linear Classifier

Margin:

\[ \gamma_i = y^i (w \cdot x^i + b) = y^i w \cdot x^i \]

proportional to perpendicular distance of point \( x^i \) 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 \( y_i \langle \mathbf{w} | \mathbf{x}_i \rangle \) for point \( i \)
    – If margin > 0, point is correctly classified
    – Else change weights to increase margin;
      change in weight proportional to \( y_i \langle \mathbf{x}_i \rangle \)
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 ($y^i \cdot w \cdot x^i$) for point i
    - If margin > 0, point is correctly classified
    - Else change weights to increase margin; change in weight proportional to $y^i \cdot x^i$

- 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 \( y^i \overline{wx}^i \) for point \( i \)
    – If margin > 0, point is correctly classified
    – Else change weights to increase margin; change in weight proportional to \( y^i \overline{x}^i \)

• 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

Initial Guess:
\[ w = [0.0 \ 0.0 \ 0.0] \]

Rate \( \eta = 0.1 \)

Final Answer:
\[ w = [-2.2 \ 0.94 \ 0.4] \]
Gradient Ascent

• Why pick $y'x'$ as increment to weights?
• To maximize scalar function of one variable $f(w)$
  • Pick initial $w$
  • Change $w$ to $w + \eta \frac{df}{dw}$ ($\eta > 0$, small)
  • until $f$ stops changing ($\frac{df}{dw} \approx 0$)
Gradient Ascent/Descent

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

$$\nabla_w f = \left[ \frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_n} \right]$$
Gradient Ascent/Descent

- To maximize $f(w)$
  - Pick initial $w$
  - Change $w$ to $w + \eta \nabla_w f$ ($\eta > 0$, small)
  - until $f$ stops changing ($\nabla_w f \approx 0$)
- Finds local maximum; global maximum if function is globally convex
- Rate ($\eta$) has to be chosen carefully.
  - Too small – slow convergence
  - Too big – oscillation
Perceptron Training via Gradient Descent

- Maximize sum of margins of misclassified points

\[ f(w) = \sum_{i \text{ misclassified}} y^i \cdot \text{wx}^i \]

\[ \nabla_w f = \sum_{i \text{ misclassified}} y^i \cdot \text{x}^i \]
Perceptron Training via Gradient Descent

• Maximize sum of margins of misclassified points

\[ f(w) = \sum_{i \text{ misclassified}} y^i w^i \]  

\[ \nabla_w f = \sum_{i \text{ misclassified}} y^i \bar{x}^i \]

• Off-line training: Compute gradient as sum over all training points.

• On-line training: Approximate gradient by one of the terms in the sum: \( y^i \bar{x}^i \)
Perceptron Algorithm
Bankruptcy Data

Initial Guess:
\[ w = [-1.0 \ 1.0 \ 1.0] \]

Final Answer:
\[ w = [-1.7 \ 0.81 \ 0.3] \]

rate \( \eta = 0.1 \)

<table>
<thead>
<tr>
<th>( w_0 )</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.0</td>
<td>1.00</td>
<td>1.0</td>
</tr>
<tr>
<td>-1.3</td>
<td>0.71</td>
<td>0.6</td>
</tr>
<tr>
<td>-1.4</td>
<td>0.66</td>
<td>0.5</td>
</tr>
<tr>
<td>-1.4</td>
<td>0.66</td>
<td>0.8</td>
</tr>
<tr>
<td>-1.5</td>
<td>0.61</td>
<td>0.7</td>
</tr>
<tr>
<td>-1.6</td>
<td>0.56</td>
<td>0.6</td>
</tr>
<tr>
<td>-1.6</td>
<td>0.65</td>
<td>0.6</td>
</tr>
<tr>
<td>-1.6</td>
<td>0.74</td>
<td>0.6</td>
</tr>
<tr>
<td>-1.7</td>
<td>0.83</td>
<td>0.6</td>
</tr>
</tbody>
</table>
Perceptron Algorithm
Bankruptcy Data

rate $\eta = 0.1$

Initial Guess: $w=[-1.0 \ 1.0 \ 1.0]$

Final Answer: $w=[-1.7 \ 0.81 \ 0.3]$
Assume initial weights are 0; rate=$\eta>0$

\[
\begin{align*}
5 \mathbf{x} & (1.0 0.2 6.0) \\
3 \mathbf{x} & (1.0 1.1 3.0) \\
-9 \mathbf{x} & (1.0 0.2 3.0) \\
-1 \mathbf{x} & (1.0 0.7 2.0) \\
-4 \mathbf{x} & (1.0 0.5 4.0) \\
-1 \mathbf{x} & (1.0 1.7 1.0) \\
\end{align*}
\]

\[
\begin{align*}
\alpha_1 y_1 \mathbf{x}_1 \\
\alpha_3 y_3 \mathbf{x}_3 \\
\alpha_4 y_4 \mathbf{x}_4 \\
\alpha_7 y_7 \mathbf{x}_7 \\
\alpha_8 y_8 \mathbf{x}_8 \\
\alpha_{11} y_{11} \mathbf{x}_{11} \\
\end{align*}
\]

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

$\alpha_i$ is count of mistakes on point $i$ during training
Dual Form

Assume initial weights are 0; \( \text{rate} = \eta > 0 \)

\[
\begin{align*}
5 \mathbf{x} &= (1.0 \ 0.2 \ 6.0) \\
3 \mathbf{x} &= (1.0 \ 1.1 \ 3.0) \\
-9 \mathbf{x} &= (1.0 \ 0.2 \ 3.0) \\
-1 \mathbf{x} &= (1.0 \ 0.7 \ 2.0) \\
-4 \mathbf{x} &= (1.0 \ 0.5 \ 4.0) \\
-1 \mathbf{x} &= (1.0 \ 1.7 \ 1.0) \\
(-7.0 \ -1.9 \ -7.0) \mathbf{x} &= 0.1
\end{align*}
\]

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

\( \alpha_i \) is count of mistakes on point \( i \) during training

\( \eta \) just scales answer, set to 1

\[
h(\mathbf{x}) = \text{sign}(\overline{\mathbf{w}} \cdot \overline{\mathbf{x}}) = \text{sign}(\sum_{i=1}^{m} \alpha_i y^i \overline{\mathbf{x}}^i \cdot \overline{\mathbf{x}})
\]
Perceptron Training
Dual Form

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