Illustrating Classification Task

Training Set

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Test Set

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>
NEAREST NEIGHBOR CLASSIFICATION
Instance-Based Classifiers

Set of Stored Cases

<table>
<thead>
<tr>
<th>Atr1</th>
<th>........</th>
<th>AtrN</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
</tbody>
</table>

- Store the training records
- Use training records to predict the class label of unseen cases

Unseen Case

<table>
<thead>
<tr>
<th>Atr1</th>
<th>........</th>
<th>AtrN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Instance Based Classifiers

• Examples:
  • Rote-learner
    • Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  
  • Nearest neighbor classifier
    • Uses k “closest” points (nearest neighbors) for performing classification
Nearest Neighbor Classifiers

• Basic idea:
  • “If it walks like a duck, quacks like a duck, then it’s probably a duck”
Nearest-Neighbor Classifiers

- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of $k$, the number of nearest neighbors to retrieve

- To classify an unknown record:
  1. Compute distance to other training records
  2. Identify $k$ nearest neighbors
  3. Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)
Nearest Neighbor Classification

• Compute distance between two points:
  • Euclidean distance

\[ d(p, q) = \sqrt{\sum_i (p_i - q_i)^2} \]

• Determine the class from nearest neighbor list
  • take the majority vote of class labels among the k-nearest neighbors
  • Weigh the vote according to distance
    • weight factor, \( w = 1/d^2 \)
Definition of Nearest Neighbor

K-nearest neighbors of a record $x$ are data points that have the $k$ smallest distance to $x$.
1 nearest-neighbor

Voronoi Diagram defines the classification boundary

The area takes the class of the green point
Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes

The value of k is the **complexity** of the model
Example

**FIGURE 2.3.** The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1), and then predicted by 1-nearest-neighbor classification.

**FIGURE 2.2.** The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1) and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.
FIGURE 2.1. A classification example in two dimensions. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1), and then fit by linear regression. The line is the decision boundary defined by $x^T \beta = 0.5$. The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.

FIGURE 2.4. Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size 200 was used, and a test sample of size 10,000. The orange curves are test and the blue are training error for k-nearest-neighbor classification. The results for linear regression are the bigger orange and blue squares at three degrees of freedom. The purple line is the optimal Bayes error rate.
Nearest Neighbor Classification…

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
- Example:
  - height of a person may vary from 1.5m to 1.8m
  - weight of a person may vary from 90lb to 300lb
  - income of a person may vary from $10K to $1M
Nearest Neighbor Classification…

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results

\[ \begin{array}{cccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array} \quad \text{vs} \quad \begin{array}{cccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array} \]

- Solution: Normalize the vectors to unit length
Nearest neighbor Classification...

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision trees
- Classifying unknown records are relatively expensive
  - Naïve algorithm: \( O(n) \)
  - Need for structures to retrieve nearest neighbors fast.
    - The Nearest Neighbor Search problem.
    - Also, Approximate Nearest Neighbor Search
SUPPORT VECTOR MACHINES
Support Vector Machines

- Find a linear hyperplane (decision boundary) that will separate the data
Support Vector Machines

- One Possible Solution
Support Vector Machines

• Another possible solution
Support Vector Machines

- Other possible solutions
Support Vector Machines

- Which one is better? B1 or B2?
- How do you define better?
Support Vector Machines

- Find hyperplane **maximizes** the margin => B1 is better than B2
Support Vector Machines

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

$$\vec{w} \cdot \vec{x} + b = -1$$

$$f(\vec{x}) = \begin{cases} 
1 & \text{if } \vec{w} \cdot \vec{x} + b \geq 1 \\
-1 & \text{if } \vec{w} \cdot \vec{x} + b \leq -1 
\end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|}$$
Support Vector Machines

• We want to maximize: \( Margin = \frac{2}{\|\mathbf{w}\|} \)

• Which is equivalent to minimizing: \( L(\mathbf{w}) = \frac{\|\mathbf{w}\|}{2} \)

• But subjected to the following constraints:
  \( \mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \) if \( y_i = 1 \)
  \( \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \) if \( y_i = -1 \)

• This is a constrained optimization problem
  • Numerical approaches to solve it (e.g., quadratic programming)
Support Vector Machines

• What if the problem is not linearly separable?
Support Vector Machines

• What if the problem is not linearly separable?

\[ \vec{w} \cdot \vec{x} + b = -1 + \xi_i \]

\[ \frac{\xi_i}{||w||} \]
Support Vector Machines

• What if the problem is not linearly separable?
• Introduce slack variables
  • Minimize:
    \[ L(w) = \frac{||w||^2}{2} + C \sum_{i=1}^{N} \xi_i \]
  • Subject to:
    \[ w \cdot x_i + b \geq 1 - \xi_i \text{ if } y_i = 1 \]
    \[ w \cdot x_i + b \leq -1 + \xi_i \text{ if } y_i = -1 \]
Nonlinear Support Vector Machines

- What if decision boundary is not linear?

\[ y(x_1, x_2) = \begin{cases} 
  1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2 \\
  -1 & \text{otherwise}
\end{cases} \]
Nonlinear Support Vector Machines

- Trick: Transform data into higher dimensional space

\[
wx \cdot \Phi(x) + b = 0
\]

\[
x_1^2 - x_1 + x_2^2 - x_2 = -0.46.
\]

\[
\Phi : (x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1).
\]

\[
w_4x_1^2 + w_3x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.
\]

Decision boundary:
\[
\vec{w} \cdot \Phi(\vec{x}) + b = 0
\]
Learning Nonlinear SVM

- Optimization problem:

\[
\min_w \frac{\|w\|^2}{2} \quad \text{subject to} \quad y_i(w \cdot \Phi(x_i) + b) \geq 1, \ \forall \{(x_i, y_i)\}
\]

- Which leads to the same set of equations (but involve \(\Phi(x)\) instead of \(x\))

\[
L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \Phi(x_i) \cdot \Phi(x_j) \quad \text{w} = \sum_i \lambda_i y_i \Phi(x_i) \quad \lambda_i \{y_i (\sum_j \lambda_j y_j \Phi(x_j) \cdot \Phi(x_i) + b) - 1\} = 0,
\]

\[
f(z) = \text{sign}(w \cdot \Phi(z) + b) = \text{sign}(\sum_{i=1}^{n} \lambda_i y_i \Phi(x_i) \cdot \Phi(z) + b).
\]
Learning NonLinear SVM

• Issues:
  • What type of mapping function $\Phi$ should be used?
  • How to do the computation in high dimensional space?
    • Most computations involve dot product $\Phi(x_i) \cdot \Phi(x_j)$
    • Curse of dimensionality?
Learning Nonlinear SVM

• **Kernel Trick:**
  
  • $\Phi(x_i) \cdot \Phi(x_j) = K(x_i, x_j)$
  
  • $K(x_i, x_j)$ is a kernel function (expressed in terms of the coordinates in the original space)
    
    • Examples:

\[
K(x, y) = (x \cdot y + 1)^p \\
K(x, y) = e^{-\|x-y\|^2/(2\sigma^2)} \\
K(x, y) = \tanh(kx \cdot y - \delta)
\]
Example of Nonlinear SVM

SVM with polynomial degree 2 kernel

\[ K(x_i, x_j) = (x_i \cdot x_j + 1)^2 \]
Learning Nonlinear SVM

- Advantages of using kernel:
  - Don’t have to know the mapping function $\Phi$
  - Computing dot product $\Phi(x_i) \cdot \Phi(x_j)$ in the original space avoids curse of dimensionality

- Not all functions can be kernels
  - Must make sure there is a corresponding $\Phi$ in some high-dimensional space
  - Mercer’s theorem (see textbook)
LOGISTIC REGRESSION
Classification via regression

• Instead of predicting the class of a record we want to predict the probability of the class given the record
• Transform the classification problem into a regression problem.
• But how do you define the probability that you want to predict?
Linear regression

- Given a dataset of the form \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \) find a linear function that given the vector \( x_i \) predicts the \( y_i \) value as \( y'_i = w^T x_i \)

- Find a vector of weights \( w \) that minimizes the sum of square errors

\[
\sum_i (y'_i - y_i)^2
\]

- Closed form solution:

\[
w = (X^T X)^{-1} X^T y
\]
Linear regression

- A simple approach: use linear regression to learn a linear function that predicts 0/1 values
  - Not good: It may produce negative probabilities, or probabilities that are greater than 1.
Class probabilities

- Assume a linear classification boundary

For the positive class, the bigger the value of $w \cdot x$, the further the point is from the classification boundary, the higher our certainty for the membership to the positive class
- Define $P(C_+|x)$ as an increasing function of $w \cdot x$

For the negative class, the smaller the value of $w \cdot x$, the further the point is from the classification boundary, the higher our certainty for the membership to the negative class
- Define $P(C_-|x)$ as a decreasing function of $w \cdot x$
Logistic Regression

The logistic function

\[ f(t) = \frac{1}{1 + e^{-t}} \]

\[ P(C_+|x) = \frac{1}{1 + e^{-w \cdot x - a}} \]

\[ P(C_-|x) = \frac{e^{-w \cdot x - a}}{1 + e^{-w \cdot x - a}} \]

\[ \log \left( \frac{P(C_+|x)}{P(C_-|x)} \right) = w \cdot x + a \]

Logistic Regression: Find the vector \( w \) that maximizes the probability of the observed data
The logistic function

\( \beta \) controls the slope
\( \alpha \) controls the position of the turning point

\[ \pi(x) = \exp(\alpha + \beta x) / (1 + \exp(\alpha + \beta x)) \]

When \( x = -\alpha / \beta \), \( \alpha + \beta x = 0 \) and hence \( \pi(x) = 1 / (1 + 1) = 0.5 \)
Logistic Regression in one dimension

Data that has a sharp survival cut off point between patients who live or die should have a large value of $\beta$.

Data with a lengthy transition from survival to death should have a low value of $\beta$. 
Logistic Regression in one dimension

Figure 10-3. The solid curved line is called a logistic regression curve. The vertical axis measures the probability that an Old Testament passage is narrative, based on the use of preterite verbs. The probability is zero for poetry and unity or one for narrative. Passages with high preterite verb counts, falling to the right of the vertical dotted line, are likely narrative. The triangle on the upper right represents Genesis 1:1–2:3, which is clearly literal, narrative history.
Logistic regression in 2-d

Coefficients

\[ \beta_1 = -1.9 \]
\[ \beta_2 = -0.4 \]
\[ \alpha = 13.04 \]
Estimating the coefficients

- **Maximum Likelihood Estimation:**
  - We have pairs of the form \((x_i, y_i)\)
- **Log Likelihood function**
  \[
  L(w) = \sum_i \left[ y_i \log P(y_i|x_i, w) + (1 - y_i) \log(1 - P(y_i|x_i, w)) \right]
  \]

- Unfortunately, it does not have a closed form solution
  - Use **gradient descend** to find local minimum
Logistic Regression

- Produces a **probability estimate** for the **class membership** which is often very useful.
- The **weights** can be useful for understanding the **feature importance**.
- Works for relatively large datasets
- Fast to apply.
INTRODUCTION TO NEURAL NETWORKS

(Thanks to Philipp Koehn for the material borrowed from his slides)
Linear Classification

• A simple model for classification is to take a linear combination of the feature values and compute a score.
• Input: Feature vector \( \mathbf{x} = (x_1, \ldots, x_n) \)
• Model: Weights \( \mathbf{w} = (w_1, \ldots, w_n) \)
• Output: \( \text{score}(\mathbf{w}, \mathbf{x}) = \sum_i w_i x_i \)
• Make a decision depending on the output score.
  • E.g.: Decide “Yes” if \( \text{score}(\mathbf{w}, \mathbf{x}) > 0 \) and “No” if \( \text{score}(\mathbf{w}, \mathbf{x}) < 0 \)
• The perceptron classification algorithm
Linear Classification

• We can represent this as a network

Input nodes correspond to features

Edges correspond to weights

"Output" node with incoming edges computes the score

\[ \text{score}(w, x) \]
Linear models

• Linear models partition the space according to a hyperplane

• But they cannot model everything
Multiple layers

- We can add more layers:
  - Each arrow has a weight
  - Nodes compute scores from incoming edges and give input to outgoing edges

Did we gain anything?
Non-linearity

- Instead of computing a linear combination

\[ score(w, x) = \sum_i w_i x_i \]

- Apply a non-linear function on top:

\[ score(w, x) = g \left( \sum_i w_i x_i \right) \]

- Popular functions:

  \[
  \begin{align*}
  \text{tanh}(x) & \quad \text{sigmoid}(x) = \frac{1}{1+e^{-x}} & \quad \text{relu}(x) = \max(0,x)
  \end{align*}
  \]

(sigmoid is also called the "logistic function")

These functions play the role of a soft “switch” (threshold function)
Side note

- Logistic regression classifier:
  - Single layer with a logistic function
Deep learning

- Networks with multiple layers

- Each layer can be thought of as a processing step
- Multiple layers allow for the computation of more complex functions
Example

• A network that implements XOR

<table>
<thead>
<tr>
<th>Input $x_0$</th>
<th>Input $x_1$</th>
<th>Hidden $h_0$</th>
<th>Hidden $h_1$</th>
<th>Output $y_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.12</td>
<td>0.02</td>
<td>0.18 → 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.88</td>
<td>0.27</td>
<td>0.74 → 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.73</td>
<td>0.12</td>
<td>0.74 → 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.99</td>
<td>0.73</td>
<td>0.33 → 0</td>
</tr>
</tbody>
</table>
Error

- The computed value is 0.76 but the correct value is 1
  - There is an error in the computation

- How do we set the weights so as to minimize this error?
Gradient Descent

- The error is a function of the weights
- We want to find the weights that minimize the error
- Compute gradient: gives the direction to the minimum
- Adjust weights, moving at the direction of the gradient.
Gradient Descent
Gradient Descent
Backpropagation

• How can we compute the gradients? **Backpropagation**!

• Main idea:
  • Start from the final layer: compute the gradients for the weights of the final layer.
  • Use these gradients to compute the gradients of previous layers using the chain rule
  • Propagate the error backwards

• Backpropagation essentially is an application of the **chain rule** for differentiation.
Error: \( E = \|y - t\|^2 = (y_1 - t_1)^2 + (y_2 - t_2)^2 \)

\[
\frac{\partial E}{\partial b_{11}} = \frac{\partial E}{\partial s_{y_1}} \frac{\partial s_{y_1}}{\partial b_{11}} = \delta_{y_1} h_1
\]

\[
\frac{\partial E}{\partial b_{21}} = \delta_{y_2} h_1 \quad \delta_{y_2} = \frac{\partial E}{\partial s_{y_2}} = 2(y_2 - t_2)g'(s_{y_2})
\]

\[
\frac{\partial E}{\partial b_{12}} = \delta_{y_1} h_2 \quad \frac{\partial E}{\partial b_{22}} = \delta_{y_2} h_2
\]

\[
\frac{\partial E}{\partial a_{11}} = \frac{\partial E}{\partial s_{h_1}} \frac{\partial s_{h_1}}{\partial a_{11}} = \delta_{h_1} x_1 \quad \frac{\partial E}{\partial a_{22}} = \frac{\partial E}{\partial s_{h_2}} \frac{\partial s_{h_2}}{\partial a_{22}} = \delta_{h_2} x_2 \quad \frac{\partial E}{\partial a_{21}} = \delta_{h_1} x_2 \quad \frac{\partial E}{\partial a_{12}} = \delta_{h_2} x_1
\]

\[
\delta_{h_1} = \frac{\partial E}{\partial s_{h_1}} = \frac{\partial E}{\partial h_1} \frac{\partial h_1}{\partial s_{h_1}} = \left( \frac{\partial E}{\partial s_{y_1}} \frac{\partial s_{y_1}}{\partial h_1} + \frac{\partial E}{\partial s_{y_2}} \frac{\partial s_{y_2}}{\partial h_1} \right) g'(s_{h_1}) = (\delta_{y_1} b_{11} + \delta_{y_2} b_{21})g'(s_{h_1})
\]

\[
\delta_{h_2} = (\delta_{y_1} b_{12} + \delta_{y_2} b_{22})g'(s_{h_2})
\]

Notation:

Activation function: \( g \)

\[
s_{y_1} = b_{11} h_1 + b_{12} h_2, \quad y_1 = g(s_{y_1})
\]

\[
s_{y_2} = b_{21} h_1 + b_{22} h_2, \quad y_2 = g(s_{y_2})
\]

\[
s_{h_1} = a_{11} x_1 + a_{12} x_2, \quad h_1 = g(s_{h_1})
\]

\[
s_{h_2} = a_{21} x_1 + a_{22} x_2, \quad h_2 = g(s_{h_2})
\]
Backpropagation

\[ \delta_{y_1} = \frac{\partial E}{\partial s_{y_1}} \quad \delta_{y_k} = \frac{\partial E}{\partial s_{y_k}} \quad \delta_{y_n} = \frac{\partial E}{\partial s_{y_n}} \]

\[ \frac{\partial E}{\partial a_{ij}} = \sum_{k=1}^{n} \delta_{y_k} b_{ki} g'(s_{hi}) x_j \]

For the sigmoid activation function:

\[ g(x) = \frac{1}{1 + e^{-x}} \]

The derivative is:

\[ g'(x) = g(x)(1 - g(x)) \]

This makes it easy to compute it. We have:

\[ g'(s_{hi}) = h_i(1 - h_i) \]

Therefore

\[ \frac{\partial E}{\partial a_{ij}} = \sum_{k=1}^{n} \delta_{y_k} b_{ki} h_i(1 - h_i) x_j \]
Stochastic gradient descent

• Ideally the loss should be the average loss over all training data.
• We would need to compute the loss for all training data every time we update the gradients.
  • However, this is expensive.
• Stochastic gradient descent: Consider one input point at the time. Each point is considered only once.
• Intermediate solution: Use mini-batches of data points.
WORD EMBEDDINGS

Thanks to Chris Manning for the slides
Basic Idea

• You can get a lot of value by representing a word by means of its neighbors
• “You shall know a word by the company it keeps”
  • (J. R. Firth 1957: 11)
• One of the most successful ideas of modern statistical NLP

These words will represent \textit{banking}
Basic idea

Define a model that aims to predict between a center word $w_c$ and context words in some window of length $m$ in terms of word vectors

$$P(w_c | w_{c-m}, \ldots, w_{c-1}, w_{c+1} \ldots, w_{c+m})$$

Pairwise probabilities

Independence assumption (bigram model)

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=2}^{n} P(w_i | w_{i-1})$$
... turning into banking crises as ...

output context words
m word window

output context words
m word window

center word
position t
Word2Vec

Predict between every word and its context words

Two algorithms

1. Skip-grams (SG)
   Predict context words given the center word

2. Continuous Bag of Words (CBOW)
   Predict center word from a bag-of-words context

*Position independent* (do not account for distance from center)

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, Jeffrey Dean: *Distributed Representations of Words and Phrases and their Compositionality*. NIPS 2013: 3111-3119
CBOW

Use a window of context words to predict the center word.

Learn two matrices (N size of embedding, |V| number of words)

- Embedding of the $i$-th word when center word
- Embedding of the $i$-th word when context word

|V| x N context embeddings when input

N x |V| center embeddings when output
CBOW

Given window size $m$ one hot vector for context words, $y$ one hot vector for the center word

1. Input: the one hot vectors for the $2m$ context words $x^{(c-m)}$, $\ldots$, $x^{(c-1)}$, $x^{(c+1)}$, $\ldots$, $x^{(c+m)}$

2. Compute the embeddings of the context words
   $v_{c-m} = Wx^{(c-m)}$, $\ldots$, $v_{c-1} = Wx^{(c-1)}$, $v_{c+1} = Wx^{(c+1)}$, $\ldots$, $v_{c+m} = Wx^{(c+m)}$

3. Average these vectors
   $\hat{v} = \frac{v_{c-m}+v_{c-m+1}+\cdots+v_{c+m}}{2m}$, $\hat{v} \in R^N$

4. Generate a score vector
   $z = W' \hat{v}$
   dot product, (embedding of center word) similar vectors close to each other

5. Turn the score vector to probabilities
   $\hat{y} = \text{softmax}(z)$
   We want this to be close to 1 for the center word
Exponentiate to make positive

\[ p_i = \frac{e^{u_i}}{\sum_j e^{u_j}} \]

Normalize to give probability

Softmax
• E.g. “The cat sat on floor”
  • Window size = 2
Index of cat in vocabulary

Input layer

Hidden layer

Output layer

one-hot vector

one-hot vector

sat
We must learn $W$ and $W'$

$W_{V \times N}$

$W'_{N \times V}$

N will be the size of word vector
\[ W_{V \times N}^T \times x_{cat} = v_{cat} \]

\[
\begin{bmatrix}
0.1 & 2.4 & 1.6 & 1.8 & 0.5 & 0.9 & \cdots & & \cdots & 3.2 \\
0.5 & 2.6 & 1.4 & 2.9 & 1.5 & 3.6 & \cdots & & \cdots & 6.1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \cdots & \cdots & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \cdots & \cdots & \cdots \\
0.6 & 1.8 & 2.7 & 1.9 & 2.4 & 2.0 & \cdots & & \cdots & 1.2
\end{bmatrix}
\]

\[ v_{cat} = 2.4 \]
\[ v_{cat} = 2.6 \]
\[ v_{cat} = \ldots \]
\[ v_{cat} = 1.8 \]

**Input layer**

**Hidden layer**

**Output layer**

\[ \hat{v} = \frac{v_{cat} + v_{on}}{2} \]

\[ V \times N \times 1 = 1 \]

\[ V \times N \times 1 = 0 \]

\[ V \times N \times 1 = 1 \]
\[ W_{V \times N}^T \times x_{on} = v_{on} \]

\begin{align*}
W_{V \times N}^T & = \\
0.12 & .41 & .61 & .80 & .50 & .9 \ldots \ldots \ldots \ldots 3.2 \\
0.52 & .61 & .42 & .91 & .53 & .6 \ldots \ldots \ldots \ldots 6.1 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
0.61 & .82 & .71 & .92 & .42 & .0 \ldots \ldots 1.2 \\
\end{align*}

\[ x_{on} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad v_{on} = \begin{bmatrix} 1.8 \\ 2.9 \\ \vdots \\ \vdots \\ 1.9 \end{bmatrix} \]

Input layer

\[ x_{cat} \]

\[ v_{cat} = W_{V \times N}^T \times x_{cat} \]

\[ \hat{v} = \frac{v_{cat} + v_{on}}{2} \]

Output layer

\[ \text{sat} \]

Hidden layer

\[ N \text{-dim} \]

\[ V \text{-dim} \]

\[ x_{on} \]
\[ \hat{v} = v \]
\[ \hat{y} = \text{softmax}(z) \]

\( N \) will be the size of word vector
We would prefer $\hat{y}$ close to $\hat{y}_{sat}$.

$W'_{V \times N} \times \hat{v} = z$

$\hat{y} = \text{softmax}(z)$

$N$ will be the size of word vector $V$-dim

$\hat{y}_{sat}$

V-dim

0.01

0.02

0.00

0.02

0.01

0.02

0.01

0.7

0.00
We can consider either $W$ (context) or $W'$ (center) as the word's representation.
Or even take the average.
Skipgram

Given the center word, predict (or, generate) the context words

\( W \): \( N \times |V| \), input matrix, word representation as \textit{center} word
\( W' \): \( |V| \times N \), output matrix, word representation as \textit{context} word

\( y^{(j)} \) one hot vector for context words

1. Get \textit{one hot vector} of the center word \( x \)

2. Get the \textit{embedding} of the center word

\[ v_c = W \cdot x \]

3. Generate a \textit{score vector for each context word}

\[ z = W' \cdot v_c \]

5. Turn the \textit{score vector into probabilities}

\[ \hat{y} = \text{softmax}(z) \]

We want this to be close to 1 for the context words
Skipgram

$W_{\text{context}} = W_{\text{word}}$

$V_{x1}$

$d \times V$

$d \times 1$

$W$

$V_{c}$

Softmax

$p(x|c) = \text{softmax}(u_{x}^{T}v_{c})$

$V_{x1}$

Truth

$W_{x3}$

Softmax

$p_{i} = \frac{e^{x_{i}}}{\sum_{j} e^{x_{j}}}$

Actual context words

One hot word symbol

Word

Looks up column of word embedding matrix as representation of center word

Output word representation
Skipgram

- For each word $t = 1 \ldots T$, predict surrounding words in a window of “radius” $m$ of every word.
- **Objective function**: Maximize the probability of any context word given the current center word:

  \[
  J'(\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} p(w_{t+j} | w_t; \theta)
  \]

  where $\theta$ represents all variables we will optimize.

  \[
  J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log p(w_{t+j} | w_t; \theta)
  \]

  **Likelihood**

  **Negative Log Likelihood**
The basic skipgram utilizes the softmax function:

\[
p(c|w) = \frac{\exp(v'^T_c v_w)}{\sum_{i=1}^{T} \exp(v'^T_i v_w)}
\]

Where:

– \( T \) – # of words in the corpus.
– \( v_w \) - input vector of \( w \).
– \( v'_w \) - output vector of \( w \).
An example
These representations are very good at encoding similarity and dimensions of similarity!

- Analogies testing dimensions of similarity can be solved quite well just by doing vector subtraction in the embedding space
  
  **Syntactically**

  - $x_{\text{apple}} - x_{\text{apples}} \approx x_{\text{car}} - x_{\text{cars}} \approx x_{\text{family}} - x_{\text{families}}$

  - Similarly for verb and adjective morphological forms

  **Semantically (Semeval 2012 task 2)**

  - $x_{\text{shirt}} - x_{\text{clothing}} \approx x_{\text{chair}} - x_{\text{furniture}}$

  - $x_{\text{king}} - x_{\text{man}} \approx x_{\text{queen}} - x_{\text{woman}}$
Test for linear relationships, examined by Mikolov et al.

\[ d = \arg \max_x \frac{(w_b - w_a + w_c)^T w_x}{\|w_b - w_a + w_c\|} \]

\[
\begin{align*}
\text{man:woman} & \quad \text{:: king:}?
\end{align*}
\]

\[
\begin{align*}
+ \quad \text{king} & \quad [0.30 \ 0.70 ] \\
- \quad \text{man} & \quad [0.20 \ 0.20 ] \\
+ \quad \text{woman} & \quad [0.60 \ 0.30 ] \\
\hline
\text{queen} & \quad [0.70 \ 0.80 ]
\end{align*}
\]
Bayes Classifier

- A probabilistic framework for solving classification problems
- A, C random variables
- Joint probability: \( \Pr(A=a, C=c) \)
- Conditional probability: \( \Pr(C=c \mid A=a) \)
- Relationship between joint and conditional probability distributions:
  \[
  \Pr(C, A) = \Pr(C \mid A) P(A) = P(A \mid C) P(C)
  \]
- Bayes Theorem:
  \[
  P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}
  \]
Bayesian Classifiers

• How to classify the new record $X = ('Yes', 'Single', 80K)"

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Evade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Find the class with the highest probability given the vector values.

Maximum Aposteriori Probability estimate:
• Find the value $c$ for class $C$ that maximizes $P(C=c| X)$

How do we estimate $P(C|X)$ for the different values of $C$?
• We want to estimate $P(C=Yes| X)$
• and $P(C=No| X)$
Bayesian Classifiers

• In order for probabilities to be well defined:
  • Consider each attribute and the class label as random variables
  • Probabilities are determined from the data

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Evade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Evade C
Event space: \{Yes, No\}
\[
P(C) = (0.3, 0.7)
\]

Refund \(A_1\)
Event space: \{Yes, No\}
\[
P(A_1) = (0.3, 0.7)
\]

Martial Status \(A_2\)
Event space: \{Single, Married, Divorced\}
\[
P(A_2) = (0.4, 0.4, 0.2)
\]

Taxable Income \(A_3\)
Event space: \(\mathbb{R}\)
\[
P(A_3) \sim \text{Normal}(\mu, \sigma^2)
\]
\[
\mu = 104: \text{sample mean}, \quad \sigma^2 = 1874: \text{sample var}
\]
Bayesian Classifiers

• Approach:
  • compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ using the Bayes theorem

$$P(C \mid A_1, A_2, \ldots, A_n) = \frac{P(A_1, A_2, \ldots, A_n \mid C)P(C)}{P(A_1, A_2, \ldots, A_n)}$$

• Maximizing

$$P(C \mid A_1, A_2, \ldots, A_n)$$

is equivalent to maximizing

$$P(A_1, A_2, \ldots, A_n \mid C) P(C)$$

• The value $P(A_1, \ldots, A_n)$ is the same for all values of $C$.

• How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?
Naïve Bayes Classifier

• Assume conditional independence among attributes $A_i$ when class $C$ is given:
  
  \[
P(A_1, A_2, ..., A_n | C) = P(A_1 | C) \cdot P(A_2 | C) \cdots P(A_n | C)
  \]

• We can estimate $P(A_i | C)$ from the data.

• New point $X = (A_1 = \alpha_1, ..., A_n = \alpha_n)$ is classified to class $c$ if
  
  \[
P(C = c | X) = P(C = c) \cdot \prod_i P(A_i = \alpha_i | c)
  \]

  is maximum over all possible values of $C$.  

Example

• Record
  \[ X = (\text{Refund} = \text{Yes}, \text{Status} = \text{Single}, \text{Income} = 80K) \]

• For the class \( C = \text{‘Evade’} \), we want to compute:
  \[ P(C = \text{Yes}|X) \text{ and } P(C = \text{No}|X) \]

• We compute:
  • \( P(C = \text{Yes}|X) = P(C = \text{Yes})*P(\text{Refund} = \text{Yes} |C = \text{Yes}) \)
  \*\( P(\text{Status} = \text{Single} |C = \text{Yes}) \)
  \*\( P(\text{Income} = 80K |C = \text{Yes}) \)
  • \( P(C = \text{No}|X) = P(C = \text{No})*P(\text{Refund} = \text{Yes} |C = \text{No}) \)
  \*\( P(\text{Status} = \text{Single} |C = \text{No}) \)
  \*\( P(\text{Income} = 80K |C = \text{No}) \)
How to Estimate Probabilities from Data?

Class Prior Probability:

$$P(C = c) = \frac{N_c}{N}$$

$N_c$: Number of records with class $c$

$N$: Number of records

P(C = No) = 7/10

P(C = Yes) = 3/10
How to Estimate Probabilities from Data?

Discrete attributes:

\[ P(A_i = a | C = c) = \frac{N_{a,c}}{N_c} \]

- \( N_{a,c} \): number of instances having attribute \( A_i = a \) and belong to class \( c \)
- \( N_c \): number of instances of class \( c \)

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Evade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>
How to Estimate Probabilities from Data?

Discrete attributes:

\[ P(A_i = a | C = c) = \frac{N_{a,c}}{N_c} \]

- \( N_{a,c} \): number of instances having attribute \( A_i = a \) and belong to class \( c \)
- \( N_c \): number of instances of class \( c \)

\[ P(\text{Refund} = \text{Yes} | \text{No}) = \frac{3}{7} \]
How to Estimate Probabilities from Data?

Discrete attributes:

\[
P(A_i = a|C = c) = \frac{N_{a,c}}{N_c}
\]

- \(N_{a,c}\): number of instances having attribute \(A_i = a\) and belong to class \(c\)
- \(N_c\): number of instances of class \(c\)

\[
P(\text{Refund} = \text{Yes}|\text{Yes}) = 0
\]
How to Estimate Probabilities from Data?

**Discrete attributes:**

\[ P(A_i = a | C = c) = \frac{N_{a,c}}{N_c} \]

- \( N_{a,c} \): number of instances having attribute \( A_i = a \) and belong to class \( c \)
- \( N_c \): number of instances of class \( c \)

\[ P(\text{Status} = \text{Single} | \text{No}) = \frac{2}{7} \]
How to Estimate Probabilities from Data?

Discrete attributes:

$$P(A_i = a | C = c) = \frac{N_{a,c}}{N_c}$$

- $N_{a,c}$: number of instances having attribute $A_i = a$ and belong to class $c$
- $N_c$: number of instances of class $c$

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Evade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

$P(\text{Status}=\text{Single}|\text{Yes}) = \frac{2}{3}$
How to Estimate Probabilities from Data?

- Normal distribution:
  \[ P(A_i = a \mid c_j) = \frac{1}{\sqrt{2\pi\sigma^2_{ij}}} e^{-\frac{(a - \mu_{ij})^2}{2\sigma^2_{ij}}} \]
  
  - One for each \((A_i, c_j)\) pair

- For **Class=No**
  - sample mean \(\mu = 110\)
  - sample variance \(\sigma^2 = 2975\)

- For **Income = 80**
  \[
P(\text{Income} = 80 \mid \text{No}) = \frac{1}{\sqrt{2\pi(54.54)}} e^{-\frac{(80-110)^2}{2(2975)}} = 0.0062
  \]
How to Estimate Probabilities from Data?

- **Normal distribution:**
  \[
P(A_i = a \mid c_j) = \frac{1}{\sqrt{2\pi\sigma^2_{ij}}} e^{-\frac{(a-\mu_{ij})^2}{2\sigma^2_{ij}}}
\]
- One for each \((A_i, c_j)\) pair
- For **Class=Yes**
  - sample mean \(\mu = 90\)
  - sample variance \(\sigma^2 = 2975\)
- For **Income = 80**
  \[
P(\text{Income} = 80 \mid \text{Yes}) = \frac{1}{\sqrt{2\pi (5)}} e^{-\frac{(80-90)^2}{2(25)}} = 0.01
\]

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Evade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Example

• Record
  \[ X = (\text{Refund} = \text{Yes}, \text{Status} = \text{Single}, \text{Income} = 80\text{K}) \]

• We compute:
  
  • \[ P(C = \text{Yes}|X) = P(C = \text{Yes}) \times P(\text{Refund} = \text{Yes}|C = \text{Yes}) \]
  \[ \times P(\text{Status} = \text{Single}|C = \text{Yes}) \]
  \[ \times P(\text{Income} = 80\text{K}|C = \text{Yes}) \]
  \[ = \frac{3}{10} \times 0 \times \frac{2}{3} \times 0.01 = 0 \]

  • \[ P(C = \text{No}|X) = P(C = \text{No}) \times P(\text{Refund} = \text{Yes}|C = \text{No}) \]
  \[ \times P(\text{Status} = \text{Single}|C = \text{No}) \]
  \[ \times P(\text{Income} = 80\text{K}|C = \text{No}) \]
  \[ = \frac{7}{10} \times \frac{3}{7} \times \frac{2}{7} \times 0.0062 = 0.0005 \]
Example of Naïve Bayes Classifier

• Creating a Naïve Bayes Classifier, essentially means to compute counts:

<table>
<thead>
<tr>
<th>Class No:</th>
<th>Number of records: 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Refund:</td>
<td>Yes: 3</td>
</tr>
<tr>
<td></td>
<td>No: 4</td>
</tr>
<tr>
<td>Attribute Marital Status:</td>
<td>Single: 2</td>
</tr>
<tr>
<td></td>
<td>Divorced: 1</td>
</tr>
<tr>
<td></td>
<td>Married: 4</td>
</tr>
<tr>
<td>Attribute Income:</td>
<td>mean: 110</td>
</tr>
<tr>
<td></td>
<td>variance: 2975</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class Yes:</th>
<th>Number of records: 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Refund:</td>
<td>Yes: 0</td>
</tr>
<tr>
<td></td>
<td>No: 3</td>
</tr>
<tr>
<td>Attribute Marital Status:</td>
<td>Single: 2</td>
</tr>
<tr>
<td></td>
<td>Divorced: 1</td>
</tr>
<tr>
<td></td>
<td>Married: 0</td>
</tr>
<tr>
<td>Attribute Income:</td>
<td>mean: 90</td>
</tr>
<tr>
<td></td>
<td>variance: 25</td>
</tr>
</tbody>
</table>

Total number of records: \( N = 10 \)

naive Bayes Classifier:

\[
P(\text{Refund}=\text{Yes}|\text{No}) = \frac{3}{7}
\]

\[
P(\text{Refund}=\text{No}|\text{No}) = \frac{4}{7}
\]

\[
P(\text{Refund}=\text{Yes}|\text{Yes}) = 0
\]

\[
P(\text{Refund}=\text{No}|\text{Yes}) = 1
\]

\[
P(\text{Marital Status}=\text{Single}|\text{No}) = \frac{2}{7}
\]

\[
P(\text{Marital Status}=\text{Divorced}|\text{No}) = \frac{1}{7}
\]

\[
P(\text{Marital Status}=\text{Married}|\text{No}) = \frac{4}{7}
\]

\[
P(\text{Marital Status}=\text{Single}|\text{Yes}) = \frac{2}{7}
\]

\[
P(\text{Marital Status}=\text{Divorced}|\text{Yes}) = \frac{1}{7}
\]

\[
P(\text{Marital Status}=\text{Married}|\text{Yes}) = 0
\]

For taxable income:

If class=No: sample mean=110, sample variance=2975

If class=Yes: sample mean=90, sample variance=25
Example of Naïve Bayes Classifier

Given a Test Record:

\[ X = (\text{Refund} = \text{Yes}, \text{Status} = \text{Single}, \text{Income} = 80K) \]

naive Bayes Classifier:

\[
\begin{align*}
P(\text{Refund}=\text{Yes}|\text{No}) &= \frac{3}{7} \\
P(\text{Refund}=\text{No}|\text{No}) &= \frac{4}{7} \\
P(\text{Refund}=\text{Yes}|\text{Yes}) &= 0 \\
P(\text{Refund}=\text{No}|\text{Yes}) &= 1 \\
\end{align*}
\]

\[
\begin{align*}
P(\text{Marital Status}=\text{Single}|\text{No}) &= \frac{2}{7} \\
P(\text{Marital Status}=\text{Divorced}|\text{No}) &= \frac{1}{7} \\
P(\text{Marital Status}=\text{Married}|\text{No}) &= \frac{4}{7} \\
P(\text{Marital Status}=\text{Single}|\text{Yes}) &= \frac{2}{7} \\
P(\text{Marital Status}=\text{Divorced}|\text{Yes}) &= \frac{1}{7} \\
P(\text{Marital Status}=\text{Married}|\text{Yes}) &= 0 \\
\end{align*}
\]

For taxable income:

- If class=No: sample mean=110, sample variance=2975
- If class=Yes: sample mean=90, sample variance=25

\[ P(\text{No}) = 0.3, \ P(\text{Yes}) = 0.7 \]

Since \( P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes}) \)

Therefore \( P(\text{No}|X) > P(\text{Yes}|X) \)

\[ => \text{Class} = \text{No} \]
Naïve Bayes Classifier

• If one of the conditional probabilities is zero, then the entire expression becomes zero.

• Laplace Smoothing:

\[ P(A_i = a|C = c) = \frac{N_{ac} + 1}{N_c + N_i} \]

• \( N_i \): number of attribute values for attribute \( A_i \)
Example of Naïve Bayes Classifier

- Creating a Naïve Bayes Classifier, essentially means to compute counts:

<table>
<thead>
<tr>
<th>Class No:</th>
<th>Number of records: 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Refund:</td>
<td>Yes: 3</td>
</tr>
<tr>
<td>Attribute Marital Status:</td>
<td>Single: 2</td>
</tr>
<tr>
<td>Attribute Income:</td>
<td>mean: 110</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class Yes:</th>
<th>Number of records: 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Refund:</td>
<td>Yes: 0</td>
</tr>
<tr>
<td>Attribute Marital Status:</td>
<td>Single: 2</td>
</tr>
<tr>
<td>Attribute Income:</td>
<td>mean: 90</td>
</tr>
</tbody>
</table>

Total number of records: \( N = 10 \)

**naive Bayes Classifier:**

\[
\begin{align*}
P(\text{Refund}=\text{Yes}|\text{No}) &= \frac{4}{9} \\
P(\text{Refund}=\text{No}|\text{No}) &= \frac{5}{9} \\
P(\text{Refund}=\text{Yes}|\text{Yes}) &= \frac{1}{5} \\
P(\text{Refund}=\text{No}|\text{Yes}) &= \frac{4}{5} \\
P(\text{Marital Status}=\text{Single}|\text{No}) &= \frac{3}{10} \\
P(\text{Marital Status}=\text{Divorced}|\text{No}) &= \frac{2}{10} \\
P(\text{Marital Status}=\text{Married}|\text{No}) &= \frac{5}{10} \\
P(\text{Marital Status}=\text{Single}|\text{Yes}) &= \frac{3}{6} \\
P(\text{Marital Status}=\text{Divorced}|\text{Yes}) &= \frac{2}{6} \\
P(\text{Marital Status}=\text{Married}|\text{Yes}) &= \frac{1}{6} \\
\end{align*}
\]

For taxable income:

- If class=No: sample mean=110, sample variance=2975
- If class=Yes: sample mean=90, sample variance=25

**With Laplace Smoothing**
Given a Test Record:

\[ X = (\text{Refund} = \text{Yes}, \text{Status} = \text{Single}, \text{Income} = 80K) \]

**Naive Bayes Classifier:**

\[
\begin{align*}
P(\text{Refund}=\text{Yes} | \text{No}) &= 4/9 \\
P(\text{Refund}=\text{No} | \text{No}) &= 5/9 \\
P(\text{Refund}=\text{Yes} | \text{Yes}) &= 1/5 \\
P(\text{Refund}=\text{No} | \text{Yes}) &= 4/5 \\
\end{align*}
\]

\[
\begin{align*}
P(\text{Marital Status}=\text{Single} | \text{No}) &= 3/10 \\
P(\text{Marital Status}=\text{Divorced} | \text{No}) &= 2/10 \\
P(\text{Marital Status}=\text{Married} | \text{No}) &= 5/10 \\
P(\text{Marital Status}=\text{Single} | \text{Yes}) &= 3/6 \\
P(\text{Marital Status}=\text{Divorced} | \text{Yes}) &= 2/6 \\
P(\text{Marital Status}=\text{Married} | \text{Yes}) &= 1/6 \\
\end{align*}
\]

For taxable income:

- If class=No: sample mean=110, sample variance=2975
- If class=Yes: sample mean=90, sample variance=25

\[ P(\text{No}) = 0.7 \quad P(\text{Yes}) = 0.3 \]

\[ P(X|\text{No})P(\text{No}) = 0.0005 \]

\[ P(X|\text{Yes})P(\text{Yes}) = 0.0003 \]

\[ => \text{Class} = \text{No} \]
Implementation details

• Computing the conditional probabilities involves multiplication of many very small numbers
  • Numbers get very close to zero, and there is a danger of numeric instability
• We can deal with this by computing the logarithm of the conditional probability

$$\log P(C|A) \sim \log P(A|C) + \log P(C)$$

$$= \sum_i \log P(A_i|C) + \log P(C)$$
Naïve Bayes for Text Classification

• Naïve Bayes is commonly used for text classification
• For a document with \(k\) terms \(d = (t_1, \ldots, t_k)\)

\[
P(c|d) = P(c)P(d|c) = P(c) \prod_{t_i \in d} P(t_i|c)
\]

\(P(t_i|c) = \text{Fraction of terms from all documents in } c \text{ that are } t_i.\)

• Easy to implement and works relatively well
• Limitation: Hard to incorporate additional features (beyond words).
  • E.g., number of adjectives used.
Multinomial document model

- Probability of document \( d = (t_1, ..., t_k) \) in class \( c \):

\[
P(d|c) = P(c) \prod_{t_i \in d} P(t_i|c)
\]

- This formula assumes a multinomial distribution for the document generation:
  - If we have probabilities \( p_1, ..., p_T \) for events \( t_1, ..., t_T \) the probability of a subset of these is

\[
P(d) = \frac{N}{N_{t_1}! N_{t_2}! \cdots N_{t_T}!} p_1^{N_{t_1}} p_2^{N_{t_2}} \cdots p_T^{N_{t_T}}
\]

- Equivalently: There is an automaton spitting words from the above distribution
**TrainMultinomialNB**($C, \mathcal{D}$)

1. \(V \leftarrow \text{ExtractVocabulary}(\mathcal{D})\)
2. \(N \leftarrow \text{CountDocs}(\mathcal{D})\)
3. for each \(c \in C\)
4. \(N_c \leftarrow \text{CountDocsInClass}(\mathcal{D}, c)\)
5. \(\text{prior}[c] \leftarrow N_c / N\)
6. \(text_c \leftarrow \text{ConcatenateTextOfAllDocsInClass}(\mathcal{D}, c)\)
7. for each \(t \in V\)
8. \(T_{ct} \leftarrow \text{CountTokensOfTerm}(text_c, t)\)
9. for each \(t \in V\)
10. \(\text{condprob}[t][c] \leftarrow \frac{T_{ct} + 1}{\sum_{t'}(T_{ct'} + 1)}\)
11. return \(V, \text{prior}, \text{condprob}\)

**ApplyMultinomialNB**($C, V, \text{prior}, \text{condprob}, d$)

1. \(W \leftarrow \text{ExtractTokensFromDoc}(V, d)\)
2. for each \(c \in C\)
3. \(\text{score}[c] \leftarrow \log \text{prior}[c]\)
4. for each \(t \in W\)
5. \(\text{score}[c] += \log \text{condprob}[t][c]\)
6. return \(\arg\max_{c \in C} \text{score}[c]\)

*Figure 13.2* Naive Bayes algorithm (multinomial model): Training and testing.
Example

News titles for Politics and Sports

**Politics**
- “Obama meets Merkel”
- “Obama elected again”
- “Merkel visits Greece again”

**Sports**
- “OSFP European basketball champion”
- “Miami NBA basketball champion”
- “Greece basketball coach?”

P(p) = 0.5

P(s) = 0.5

**Terms**
- obama:2, meets:1, merkel:2, elected:1, again:2, visits:1, greece:1
- OSFP:1, european:1, basketball:3, champion:2, miami:1, nba:1, greece:1, coach:1

**Vocabulary size:** 14

**Total terms:** 10

**Total terms:** 11

**New title:** X = “Obama likes basketball”

\[
P(\text{Politics}|X) \sim P(p) \times P(\text{obama}|p) \times P(\text{likes}|p) \times P(\text{basketball}|p)
\]
\[
= 0.5 \times \frac{3}{10 + 14} \times \frac{1}{10 + 14} \times \frac{1}{10 + 14} = 0.000108
\]

\[
P(\text{Sports}|X) \sim P(s) \times P(\text{obama}|s) \times P(\text{likes}|s) \times P(\text{basketball}|s)
\]
\[
= 0.5 \times \frac{1}{11 + 14} \times \frac{4}{11 + 14} = 0.000128
\]
Naïve Bayes (Summary)

• Robust to isolated noise points

• Handle missing values by ignoring the instance during probability estimate calculations

• Robust to irrelevant attributes

• Independence assumption may not hold for some attributes
  • Use other techniques such as Bayesian Belief Networks (BBN)

• Naïve Bayes can produce a probability estimate, but it is usually a very biased one
  • Logistic Regression is better for obtaining probabilities.
Generative vs Discriminative models

- Naïve Bayes is a type of a generative model
  - Generative process:
    - First pick the category of the record
    - Then given the category, generate the attribute values from the distribution of the category
  - Conditional independence given C

- We use the training data to learn the distribution of the values in a class
Generative vs Discriminative models

- Logistic Regression and SVM are **discriminative models**
  - The goal is to find the boundary that discriminates between the two classes from the training data

- In order to classify the language of a document, you can
  - Either learn the two languages and find which is more likely to have generated the words you see
  - Or learn what differentiates the two languages.
SUPERVISED LEARNING
Learning

• **Supervised Learning**: learn a model from the data using *labeled data*.
  • *Classification* and *Regression* are the prototypical examples of supervised learning tasks. Other are possible (e.g., ranking)

• **Unsupervised Learning**: learn a model – extract structure from *unlabeled data*.
  • *Clustering* and *Association Rules* are prototypical examples of unsupervised learning tasks.

• **Semi-supervised Learning**: learn a model for the data using both *labeled and unlabeled data*. 
Supervised Learning Steps

• **Model** the problem
  • What is you are trying to predict? What kind of optimization function do you need? Do you need classes or probabilities?

• **Extract Features**
  • How do you find the right features that help to discriminate between the classes?

• **Obtain training data**
  • Obtain a collection of labeled data. Make sure it is large enough, accurate and representative. Ensure that classes are well represented.

• **Decide on the technique**
  • What is the right technique for your problem?

• **Apply in practice**
  • Can the model be trained for very large data? How do you test how you do in practice? How do you improve?
Modeling the problem

- Sometimes it is not obvious. Consider the following three problems
  - Detecting if an email is spam
  - Categorizing the queries in a search engine
  - Ranking the results of a web search
  - Predicting the reply to a question.
Feature extraction

- Feature extraction, or feature engineering is the most tedious but also the most important step
  - How do you separate the players of the Greek national team from those of the Swedish national team?

- One line of thought: throw features to the classifier and the classifier will figure out which ones are important
  - More features, means that you need more training data
- Another line of thought: Feature Selection: Select carefully the features using various functions and techniques
  - Computationally intensive

- Deep Neural Networks
  - They use raw data for classification
  - They learn a representation from the data
Training data

• An overlooked problem: How do you get labeled data for training your model?
  • E.g., how do you get training data for ranking?
  • Chicken and egg problem

• Usually requires a lot of manual effort and domain expertise and carefully planned labeling
  • Results are not always of high quality (lack of expertise)
  • And they are not sufficient (low coverage of the space)

• Recent trends:
  • Find a source that generates the labeled data for you, or use the data themselves for the prediction task
  • Crowd-sourcing techniques
Dealing with small amounts of labeled data

• **Semi-supervised learning** techniques have been developed for this purpose.

• **Self-training**: Train a classifier on the data, and then feed back the high-confidence output of the classifier as input.

• **Co-training**: train two “independent” classifiers and feed the output of one classifier as input to the other.

• **Regularization**: Treat learning as an optimization problem where you define relationships between the objects you want to classify, and you exploit these relationships
  • Example: Image restoration
Technique

• The choice of technique depends on the problem requirements (do we need a probability estimate?) and the problem specifics (does independence assumption hold? do we think classes are linearly separable?)
• For many cases finding the right technique may be trial and error
• For many cases the exact technique does not matter.
Big Data Trumps Better Algorithms

- If you have enough data then the algorithms are not so important.
- The web has made this possible.
  - Especially for text-related tasks
  - Search engine uses the collective human intelligence.

Google lecture: Theorizing from the Data
Apply-Test

- How do you scale to very large datasets?
  - Distributed computing – map-reduce implementations of machine learning algorithms (Mahaut, over Hadoop)

- How do you test something that is running online?
  - You cannot get labeled data in this case
  - A/B testing

- How do you deal with changes in data?
  - Active learning