THUS FAR, REGRESSION: PREDICT A CONTINUOUS VALUE GIVEN SOME INPUTS
Weather prediction revisited
Reading Your Brain, Simple Example

Pairwise classification accuracy: 85%

Person

Animal
Classification

- **Learn**: $h : X \mapsto Y$
  - $X$ – features
  - $Y$ – target classes

- **Simplest case**: Thresholding
Linear (Hyperplane) Decision Boundaries
Learning a Linear Classifier

- Learn: $h: X \rightarrow Y$
  - $X$ – features
  - $Y$ – target classes

- Decision rule:
Challenge: Data is streaming

Assumption thus far: **Batch data**

But, e.g., in click prediction for ads is a streaming data task:

- User enters query, and ad must be selected:
  - Observe $x_i$, and must predict $y_i$

- User either clicks or doesn’t click on ad:
  - Label $y_i$ is revealed afterwards
    - Google gets a reward if user clicks on ad

- Weights must be updated for next time:
Online Learning Problem

At each time step $t$:

- Observe features of data point:
  - Note: many assumptions are possible, e.g., data is iid, data is adversarially chosen… details beyond scope of course

- Make a prediction:
  - Note: many models are possible, we focus on linear models
  - *For simplicity, use vector notation*

- Observe true label:
  - Note: other observation models are possible, e.g., we don’t observe the label directly, but only a noisy version… Details beyond scope of course

- Update model:
Rosenblatt 1957
The Perceptron Algorithm [Rosenblatt '58, '62]

- Classification setting: $y$ in {-1,+1}
- Linear model
  - Prediction:

Training:
- Initialize weight vector:
- At each time step:
  - Observe features:
  - Make prediction:
  - Observe true class:

- Update model:
  - If prediction is not equal to truth
Fundamental Practical Problem for All Online Learning Methods: **Which weight vector to report?**

- Perceptron prediction:
  - Suppose you run online learning method and want to sell your learned weight vector… Which one do you sell???

- Last one?
Choice can make a huge difference!!

[Freund & Schapire '99]
Mistake Bounds

- Algorithm “pays” every time it makes a mistake:

- How many mistakes is it going to make?
Data linearly separable, if there exists
- a vector
- a margin

Such that
Perceptron Analysis: Linearly Separable Case

- Theorem [Block, Novikoff]:
  - Given a sequence of labeled examples:
  - Each feature vector has bounded norm:
  - If dataset is linearly separable:

- Then the number of mistakes made by the online perceptron on any such sequence is bounded by
Perceptron Proof for Linearly Separable case

- Every time we make a mistake, we get gamma closer to \( w^* \):
  - Mistake at time \( t \): \( w^{(t+1)} = w^{(t)} + y^{(t)} x^{(t)} \)
  - Taking dot product with \( w^* \):
  - Thus after \( m \) mistakes:

- Similarly, norm of \( w^{(t+1)} \) doesn’t grow too fast:
  - \( ||w^{(t+1)}||^2 = ||w^{(t)}||^2 + 2y^{(t)}(w^{(t)} \cdot x^{(t)}) + ||x^{(t)}||^2 \)
  - Thus, after \( m \) mistakes:

- Putting all together:
Beyond Linearly Separable Case

- Perceptron algorithm is super cool!
  - No assumption about data distribution!
    - Could be generated by an oblivious adversary, no need to be iid
  - Makes a fixed number of mistakes, and it’s done for ever!
    - Even if you see infinite data

- However, real world not linearly separable
  - Can’t expect never to make mistakes again
  - Analysis extends to non-linearly separable case
  - Very similar bound, see Freund & Schapire
  - Converges, but ultimately may not give good accuracy (make many many many mistakes)
What you need to know

- Notion of online learning
- Perceptron algorithm
- Mistake bounds and proof
- In online learning, report averaged weights at the end
Summary Thus Far

- **Perceptron algorithm:**
  - Extremely simple classifier, works well in practice
  - If you generalize it slightly by adding regularization ➔ called a support vector machine (more next time)

- Constant number of mistakes in the linearly separable case
  - More general results in the non-linearly separable case

- In general, performance depends on how well we can separate the data
What if the data is not linearly separable?

Use features of features of features of features....

\[ \Phi(x) : R^m \rightarrow F \]

Feature space can get really large really quickly!
Higher order polynomials

num. terms = \( \binom{d + m - 1}{d} = \frac{(d + m - 1)!}{d!(m - 1)!} \)

- \( m \) – input features
- \( d \) – degree of polynomial

The number of monomial terms grows fast!

- \( d = 6, m = 100 \)
- About 1.6 billion terms
Perceptron Revisited

- Given weight vector $w^{(t)}$, predict point $x$ by:

- Mistake at time $t$: $w^{(t+1)} \leftarrow w^{(t)} + y^{(t)} x^{(t)}$

- Thus, write weight vector in terms of mistaken data points only:
  - Let $M^{(t)}$ be time steps up to $t$ when mistakes were made:

- Prediction rule now:

- When using high dimensional features:
Dot-product of polynomials

$\Phi(u) \cdot \Phi(v) = \text{polynomials of degree exactly d}$
Finally the Kernel Trick!!!
(Kernelized Perceptron)

- Every time you make a mistake, remember \((x^{(t)}, y^{(t)})\)

- Kernelized Perceptron prediction for \(\mathbf{x}\):

\[
sign(w^{(t)} \cdot \phi(\mathbf{x})) = \sum_{j \in M^{(t)}} y^{(j)} \phi(x^{(j)}) \cdot \phi(\mathbf{x}) = \sum_{j \in M^{(t)}} y^{(j)} k(x^{(j)}, \mathbf{x})
\]
Polynomial kernels

- All monomials of degree $d$ in $O(d)$ operations:
  \[ \Phi(u) \cdot \Phi(v) = (u \cdot v)^d = \text{polynomials of degree exactly } d \]

- How about all monomials of degree up to $d$?
  
  □ Solution 0:

  □ Better solution:
Common kernels

- Polynomials of degree exactly $d$
  \[ K(u, v) = (u \cdot v)^d \]

- Polynomials of degree up to $d$
  \[ K(u, v) = (u \cdot v + 1)^d \]

- Gaussian (squared exponential) kernel
  \[ K(u, v) = \exp \left( -\frac{||u - v||^2}{2\sigma^2} \right) \]

- Sigmoid
  \[ K(u, v) = \tanh(\eta u \cdot v + \nu) \]
What you need to know

- Notion of online learning
- Perceptron algorithm
- Mistake bounds and proofs
- The kernel trick
- Kernelized Perceptron
- Derive polynomial kernel
- Common kernels
- In online learning, report averaged weights at the end
Naïve Bayes

Machine Learning – CSEP546
Carlos Guestrin
University of Washington

January 21, 2014
Classification

- Learn: h: X \rightarrow Y
  - X – features
  - Y – target classes

- Thus far: just a decision boundary

- What if you want probability of each class? P(Y|X)
Companies bid on ad prices

Which ad wins? (many simplifications here)

- Naively:
- But:
- Instead:
Key Task: Estimating Click Probabilities

- What is the probability that user \( i \) will click on ad \( j \)

- Not important just for ads:
  - Optimize search results
  - Suggest news articles
  - Recommend products

- Methods much more general, useful for:
  - Classification
  - Regression
  - Density estimation
Learning Problem for Click Prediction

- Prediction task:

- Features:

- Data:
  - Batch:
  - Online:

- Many approaches (e.g., logistic regression, SVMs, naïve Bayes, decision trees, boosting, …)
  - Focus on naïve Bayes and logistic regression; captures main concepts, ideas generalize to other approaches
Bayes Rule

\[ P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \]

Which is shorthand for:

\[ (\forall i, j) P(Y = y_i|X = x_j) = \frac{P(X = x_j|Y = y_i)P(Y = y_i)}{P(X = x_j)} \]
How hard is it to learn the optimal classifier?

Data =

<table>
<thead>
<tr>
<th>Gender</th>
<th>Age</th>
<th>Location</th>
<th>Income</th>
<th>Referrer</th>
<th>New or Returning</th>
<th>Clicked?</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Young</td>
<td>US</td>
<td>High</td>
<td>Google</td>
<td>New</td>
<td>N</td>
</tr>
<tr>
<td>M</td>
<td>Middle</td>
<td>US</td>
<td>Low</td>
<td>Direct</td>
<td>New</td>
<td>N</td>
</tr>
<tr>
<td>F</td>
<td>Old</td>
<td>BR</td>
<td>Low</td>
<td>Google</td>
<td>Returning</td>
<td>Y</td>
</tr>
<tr>
<td>M</td>
<td>Young</td>
<td>BR</td>
<td>Low</td>
<td>Bing</td>
<td>Returning</td>
<td>N</td>
</tr>
</tbody>
</table>

How do we represent these? How many parameters?

- Prior, $P(Y)$:
  - Suppose $Y$ is composed of $k$ classes

- Likelihood, $P(X|Y)$:
  - Suppose $X$ is composed of $d$ binary features

Complex model! High variance with limited data!!!
Conditional Independence

- **X** is **conditionally independent** of **Y** given **Z**, if the probability distribution governing **X** is independent of the value of **Y**, given the value of **Z**

\[(\forall i, j, k) P(X = i | Y = j, Z = k) = P(X = i | Z = k)\]

- e.g., \[P(\text{Thunder} | \text{Rain}, \text{Lightning}) = P(\text{Thunder} | \text{Lightning})\]

- Equivalent to:

\[P(X, Y | Z) = P(X | Z)P(Y | Z)\]
What if features are independent?

- Predict Thunder
- From two *conditionally Independent* features
  - Lightening
  - Rain
The Naïve Bayes assumption

Naïve Bayes assumption:

- Features are independent given class:

\[ P(X_1, X_2 | Y) = P(X_1 | X_2, Y) P(X_2 | Y) \]
\[ = P(X_1 | Y) P(X_2 | Y) \]

- More generally:

\[ P(X_1 \ldots X_d | Y) = \prod_{i} P(X_i | Y) \]

How many parameters now?

- Suppose \( X \) is composed of \( d \) binary features
The Naïve Bayes Classifier

- Given:
  - Prior $P(Y)$
  - $d$ conditionally independent features $X$ given the class $Y$
  - For each $X_i$, we have likelihood $P(X_i|Y)$

- Decision rule:
  $$y^* = h_{NB}(x) = \arg \max_y P(y)P(x_1, \ldots, x_d | y)$$
  $$= \arg \max_y P(y) \prod_i P(x_i | y)$$

- If assumption holds, NB is optimal classifier!
MLE for the parameters of NB

- Given dataset
  - Count(A=a,B=b) == number of examples where A=a and B=b

- MLE for NB, simply:
  - Prior: P(Y=y) =
  - Likelihood: P(X_i=x_i|Y=y) =
Subtleties of NB classifier 1 – Violating the NB assumption

- Usually, features are not conditionally independent:

\[ P(X_1 \ldots X_d | Y) \neq \prod_i P(X_i | Y) \]

- Actual probabilities \( P(Y|X) \) often biased towards 0 or 1

- Nonetheless, NB is the single most used classifier out there
  - NB often performs well, even when assumption is violated
  - [Domingos & Pazzani ’96] discuss some conditions for good performance
Subtleties of NB classifier 2 – Insufficient training data

- What if you never see a training instance where $X_1=a$ when $Y=b$?
  - e.g., $Y=${SpamEmail}, $X_1=${‘CSEP546’}
  - $P(X_1=a \mid Y=b) = 0$

- Thus, no matter what the values $X_2,\ldots,X_d$ take:
  - $P(Y=b \mid X_1=a,X_2,\ldots,X_d) = 0$

- “Solution”: smoothing
  - Add “fake” counts, usually uniformly distributed
  - Equivalent to “Bayesian Learning”
Text classification

- Classify e-mails
  - $Y = \{\text{Spam, NotSpam}\}$

- Classify news articles
  - $Y = \{\text{what is the topic of the article?}\}$

- Classify webpages
  - $Y = \{\text{student, professor, project, ...}\}$

- What about the features $X$?
  - The text!
Features $X$ are entire document – $X_i$ for $i^{th}$ word in article

Article from rec.sport.hockey

Path: cantaloupe.srv.cs.cmu.edu!das-news.harvard.edu
From: xxx@yyy.zzz.edu (John Doe)
Subject: Re: This year’s biggest and worst (opinic
Date: 5 Apr 93 09:53:39 GMT

I can only comment on the Kings, but the most obvious candidate for pleasant surprise is Alex Zhitnik. He came highly touted as a defensive defenseman, but he’s clearly much more than that. Great skater and hard shot (though wish he were more accurate). In fact, he pretty much allowed the Kings to trade away that huge defensive liability Paul Coffey. Kelly Hrudey is only the biggest disappointment if you thought he was any good to begin with. But, at best, he’s only a mediocre goaltender. A better choice would be Tomas Sandstrom, though not through any fault of his own, but because some thugs in Toronto decided
NB for Text classification

- $P(X|Y)$ is huge!!!
  - Article at least 1000 words, $X=\{X_1, \ldots, X_{1000}\}$
  - $X_i$ represents $i^{th}$ word in document, i.e., the domain of $X_i$ is entire vocabulary, e.g., Webster Dictionary (or more), 10,000 words, etc.

- NB assumption helps a lot!!!
  - $P(X_i=x_i|Y=y)$ is just the probability of observing word $x_i$ in a document on topic $y$

\[
h_{NB}(x) = \arg \max_y P(y) \prod_{i=1}^{\text{LengthDoc}} P(x_i|y)
\]
Bag of words model

- Typical additional assumption – **Position in document doesn’t matter**: $P(X_i=x_i|Y=y) = P(X_k=x_i|Y=y)$
  - “Bag of words” model – order of words on the page ignored
  - Sounds really silly, but often works very well!

\[
P(y) \prod_{i=1}^{\text{LengthDoc}} P(x_i|y)
\]

When the lecture is over, remember to wake up the person sitting next to you in the lecture room.
Bag of words model

- Typical additional assumption – **Position in document doesn’t matter**: \(P(X_i=x_i|Y=y) = P(X_k=x_i|Y=y)\)
  - “Bag of words” model – order of words on the page ignored
  - Sounds really silly, but often works very well!

\[
P(y) \prod_{i=1}^{\text{LengthDoc}} P(x_i|y)
\]

```
in is lecture lecture next over person remember room sitting the the the to to up wake when you
```
Bag of Words Approach

Our energy exploration, production, and distribution operations span the globe, with activities in more than 100 countries.

At TOTAL, we draw our greatest strength from our fast-growing oil and gas reserves. Our strategic emphasis on natural gas provides a strong position in a rapidly expanding market.

Our expanding refining and marketing operations in Asia and the Mediterranean Rim complement already solid positions in Europe, Africa, and the U.S.

Our growing specialty chemicals sector adds balance and profit to the core energy business.
NB with Bag of Words for text classification

- **Learning phase:**
  - Prior $P(Y)$
    - Count how many documents you have from each topic (+ prior)
  - $P(X_i|Y)$
    - For each topic, count how many times you saw word in documents of this topic (+ prior)

- **Test phase:**
  - For each document
    - Use naïve Bayes decision rule

\[
h_{NB}(x) = \arg \max_y P(y) \prod_{i=1}^{\text{LengthDoc}} P(x_i|y)
\]
Twenty News Groups results

Given 1000 training documents from each group
Learn to classify new documents into which newsgroup it came from

comp.graphics      misc.forsale
comp.os.ms-windows.misc  rec.autos
comp.sys.ibm.pc.hardware   rec.motorcycles
comp.sys.mac.hardware     rec.sport.baseball
comp.windows.x          rec.sport.hockey
alt.atheism             sci.space
soc.religion.christian   sci.crypt
talk.religion.misc      sci.electronics
talk.politics.mideast   sci.med
talk.politics.misc      talk.politics.guns

Naive Bayes: 89% classification accuracy
Learning curve for Twenty News Groups

Accuracy vs. Training set size
What you need to know

- Click prediction problem
- Probabilities rather than classification
- Naïve Bayes model
  - Assumption
  - Formulation
- Application to text data
  - Bag of words model