Lecture 9: Classification, Part 2
Modeling Social Data, Spring 2017
Columbia University

March 24, 2017

Notes from ads2206

1 Classification Continued

1.1 Naive Bayes

We saw that Naive Bayes fundementally was "Naive" due in part by the naive assumption that each feature was independent from each other conditioned on their class.

We took a look at the classic spam filtering problem. We have an email with D words so each email can be represented by a vector $\mathbf{x} \in \mathbb{R}^D$, where the ith component of $\mathbf{x}$, is denoted as x_i, represents the ith word in the email. Our goal is to classify this email as spam or ham (not spam). Usually D is on the scale of 500k to 100k.

- $\vec{\mathbf{x}} =$ (I am a Nigerian prince and I need your help)

Using Bayes Rule, we have the following:

$$P(A|B)P(B) = P(A \cap B) = P(B|A)P(A) = \quad \Rightarrow \quad P(c|\mathbf{x}) = P(\mathbf{x}|c)\frac{P(c)}{P(\mathbf{x})} \tag{1}$$

Here c denotes a certain class. In this case, $c \in \{\text{spam}, \text{not spam}\}$.

The naive part of Naive Bayes says:

$$P(\mathbf{x}|c) = \prod_{i=1}^{D} P(x_i|c)$$

Consequrently (1) becomes,

$$P(c|\mathbf{x}) = \prod_{i=1}^{D} P(x_i|c)\frac{P(c)}{P(\mathbf{x})} \tag{2}$$

We model each term in the series of products to be a coin flip. In other words, $P(x_j|c)$ will represent the distribution of a Bernoulli random variable X_j, which we will have an associated parameter θ_{jc}. θ_{jc} represents the probablity that the jth word appears under class c.

$$P(x_j|c) = \theta_{jc}^{x_j}(1 - \theta_{jc}^{1-x_j}) \tag{3}$$

With this in mind and denoting $P(c) = \theta_c$, we plug this back into (2) and take the log of the Likelihood to get:

$$\log(p(c|\mathbf{x})) = \log\left(\prod_{i=j}^{D} \theta_{jc}^{x_j}(1 - \theta_{jc}^{1-x_j})\frac{P(c)}{P(\mathbf{x})}\right) = \log\left(\prod_{i=j}^{D} \theta_{jc}^{x_j}(1 - \theta_{jc}^{1-x_j}\right) + \log\left(\frac{\theta_c}{P(\mathbf{x})}\right)$$

$$\Rightarrow \sum_{j=1}^{D} \log[\theta_{jc}^{x_j}(1 - \theta_{jc})^{1-x_j}] + \log\frac{\theta_c}{P(\mathbf{x})} = \sum_{j=1}^{D} x_j \log\theta_{jc} + \sum_{j=1}^{D}(1 - x_j)\log(1 - \theta_{jc}) + \log\frac{\theta_c}{P(\mathbf{x})}$$

$$\sum_{j=1}^{D} x_j \log\theta_{jc} + \sum_{j=1}^{D}\log(1 - \theta_{jc}) - \sum_{j=1}^{D} x_j \log(1 - \theta_{jc}) + \log\frac{\theta_c}{P(\mathbf{x})}$$

$$\log(p(c|\mathbf{x})) = \sum_{j=1}^{D} x_j \log\frac{\theta_{jc}}{1 - \theta_{jc}} + \sum_{j=1}^{D}\log(1 - \theta_{jc}) + \log\frac{\theta_c}{P(\mathbf{x})} \tag{4}$$

To find $P(\mathbf{x})$, we can condition $\mathbf{x}$ on the number of classes and multiply by the respective probabilities of being in that class.

$$P(\mathbf{x}) = \sum_c P(\mathbf{x}, c) = \sum_c P(\mathbf{x}|c)P(c)$$

Restricting ourselves to the 2-class example problem at hand, c can only take values 0 (not spam) or 1 (spam). Taking the difference in probability between an email being in class 1 and that same email being in class 0, we have:

$$P(c = 1|\mathbf{x}) - P(c = 0|\mathbf{x}) \quad \Rightarrow \quad \log \frac{P(c = 1|\mathbf{x})}{P(c = 0|\mathbf{x})} \tag{5}$$

Using equation (4) to calculate $P(c = 1|\mathbf{x})$ and $P(c = 0|\mathbf{x})$:

$$\log P(c = 1|\mathbf{x}) = \sum_{j=1}^{D} x_j \log \frac{\theta_{j1}}{1 - \theta_{j1}} + \sum_{j=1}^{D} \log(1 - \theta_{j1}) + \log \frac{\theta_1}{P(\mathbf{x})} \tag{6}$$

$$\log P(c = 0|\mathbf{x}) = \sum_{j=1}^{D} x_j \log \frac{\theta_{j0}}{1 - \theta_{j0}} + \sum_{j=1}^{D} \log(1 - \theta_{j0}) + \log \frac{\theta_0}{P(\mathbf{x})} \tag{7}$$

Taking the difference between (6) and (7) is equivalent to (5).

$$\sum_{j=1}^{D} x_j \log \frac{\theta_{j1}}{1 - \theta_{j1}} + \sum_{j=1}^{D} \log(1 - \theta_{j1}) + \log \frac{\theta_1}{P(\mathbf{x})} - \left(\sum_{j=1}^{D} x_j \log \frac{\theta_{j0}}{1 - \theta_{j0}} + \sum_{j=1}^{D} \log(1 - \theta_{j0}) + \log \frac{\theta_0}{P(\mathbf{x})} \right)$$

$$\Rightarrow \log \frac{P(c = 1|\mathbf{x})}{P(c = 0|\mathbf{x})} = \sum_{j=1}^{D} x_j \log \frac{\theta_{j1}(1 - \theta_{j0})}{\theta_{j0}(1 - \theta_{j1})} + \sum_{j=1}^{D} \log \frac{1 - \theta_{j1}}{1 - \theta_{j0}} + \log \frac{\theta_1}{\theta_0}$$

Defining $\log \frac{\theta_{j1}(1-\theta_{j0})}{\theta_{j0}(1-\theta_{j1})} = w_j$ where $\mathbf{w} = (w_1, \ldots, w_D) \in \mathbb{R}^D$, then we describe the difference between these log probabilities as:

$$\log \frac{P(c = 1|\mathbf{x})}{P(c = 0|\mathbf{x})} = \mathbf{x} \cdot \mathbf{w} + w_0 \tag{8}$$

As a side note, $w_0 = log \frac{\theta_1}{\theta_)}$ and therefore this term can be computed without knowing x.

How this relate to classification? Intuitively, we want to classify a vector $\mathbf{x}$ to be in a certain class if it has the highest probability of belong in that class. In the 2-class case, we classify $\mathbf{x}$ to be in class 1 only if $P(c = 1|\mathbf{x}) > P(c = 0|\mathbf{x})$ or we classify it to be in class 0 only if $P(c = 1|\mathbf{x}) < P(c = 0|\mathbf{x})$. Because log is a monotonically increasing function then all of these results hold if we log both sides.

$$\log \frac{P(c = 1|\mathbf{x})}{P(c = 0|\mathbf{x})} = \log P(c = 1|\mathbf{x}) - \log P(c = 0|\mathbf{x})$$

Therefore, if $\mathbf{x}$ is in class 1, we have $\log P(c = 1|\mathbf{x}) > \log P(c = 0|\mathbf{x}) \quad \Rightarrow \quad \log P(c = 1|\mathbf{x}) - \log P(c = 0|\mathbf{x}) > 0$.

In other words, if $\log \frac{P(c=1|\mathbf{x})}{P(c=0|\mathbf{x})} = \mathbf{x} \cdot \mathbf{w} + w_0 > 0$, we classify $\mathbf{x}$ to be in 1, otherwise if it is < 0, we classify it as class 0.

All this is left is finding the appropriate parameters θ_{jc} for the jth word that belongs in a certain class. The best value is determined by maximizing the likelihood with respect to the parameters θ.

In class, the way we approached this seemingly complex problem is by simply loooking at the following problem: Imagine tossing a coin N times that lands hands with probability θ. Further imagine that of these N tosses, we observed heads n times. Then:

$P(n \text{ heads}|\theta) = C\theta^n(1 - \theta)^{N-n}$ Where C is some aribtrary constant.

$$\ell = \log P(n \text{ heads}|\theta) = \log(C) + n \log(\theta) + (N - n) \log(1 - \theta)$$

$$\frac{d\ell}{d\theta} = \frac{n}{\theta} - \frac{N-n}{1-\theta} = 0 \quad \Rightarrow \quad n - n\theta = \theta N - \theta n \quad \Rightarrow \; \theta = \frac{n}{N} = \frac{\#\text{ head occurs}}{\text{total tosses}}$$

Then, the best estimators in our model are:

$$\theta_{j1} = \frac{n_{j1}}{n_1} = \frac{\#\text{ spam docs with word j}}{\#\text{ spam documents}} \tag{9}$$

$$\theta_{j0} = \frac{n_{j0}}{n_0} = \frac{\#\text{ non-spam docs with word j}}{\#\text{ non-spam documents}} \tag{10}$$

$$\theta_1 = \frac{n_1}{N} = \frac{\#\text{ spam documents}}{\#\text{ total docs}} \qquad \theta_0 = \frac{n_0}{N} = \frac{\#\text{ non-spam documents}}{\#\text{ total docs}} \tag{11}$$

1.1.1 Runtime Complexity

The run time complexity of running Naive Bayes with N documents and K, the number of words in our vocabulary, is $\mathcal{O}(KN)$ if we're not being clever. However, to improve efficeincy all we have to do is run through the data set only once and keep track of words that appear. Therefore, our run time is $\mathcal{O}(\bar{k}N)$, where $\bar{k}$ is the average number of words in a document. This run time is the time complexity. It requires us to run through all the documents, where in each run we append the document to a dictionary and then create a new dictionary associated with this document with all the counts of the words.

The Naive Bayes model can be updated continually as it sees more and more data because the parameters θ would simply get incremented in the appropriate fashion.

1.2 Logistic Regression

Wht if we just start with a predictor of the same form that we derived in Naive Bayes with some unknown weights? What we would find in solving for the probability of being in some class 1. (i.e p is the probability something is classified as class 1), we would get:

$$\log \frac{p}{1-p} = \mathbf{w} \cdot \mathbf{x} \qquad \text{where the intercept has been adding implcitly into the vector w} \tag{12}$$

Solving for p leads us with the following sigmoid function, which looks like the graph below in 2-dimensions:

$$p = \frac{1}{1 + e^{-\mathbf{w}\cdot\mathbf{x}}} \tag{13}$$

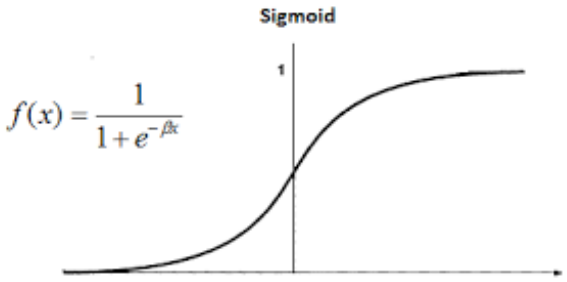

Figure 1: A graph of a sigmoid function in 2 dimensions.

In this case, as $p \to 1$, the higher the probability we have that this vector is a part of class 1.

Like in other Machine Learning models, we need to find the optimal parameters. We turn to maximum likelihood and defining a form of loss that we hope to minimize. Note, each vector $\mathbf{x}_i$ has an associated class label y_i and $p_i = P(\mathbf{x}_i|\mathbf{w})$.

Our likelihood function is:

$$L = \prod_i p_i^{y_i}(1-p_i)^{1-y_i}$$

Our loss function is therefore:

$$\mathcal{L} = -\log P(\text{ data } |\mathbf{w}) = -\sum_i y_i \log p_i + (1-y_i)\log(1-p_i) \tag{14}$$

This simplifies to:

$$\Rightarrow -\sum_i y_i \log p_i + \log(1-p_i) - y_i \log(1-p_i) = -\sum_i y_i \log \frac{p_i}{1-p_i} + \log(1-p_i)$$

$$\mathcal{L} = -\sum_i y_i \mathbf{w}\cdot\mathbf{x}_i - \log(1+e^{\mathbf{w}\cdot\mathbf{x}})$$

We need to find the optimal $\mathbf{w}$, so we'll turn to some basic calculus:

$$\frac{\partial \mathcal{L}}{\partial w_k} = -\sum_i y_i x_{ik} - \frac{e^{\mathbf{w}\cdot\mathbf{x}_i} x_{ik}}{1+e^{\mathbf{w}\cdot\mathbf{x}_i}} = 0 \quad w_k \text{ is the kth component of the vector} \mathbf{w}$$

For each component of the vector w, we have a transcendental equation which has no closed form solution.

We turn to gradient descent, and have the following update equations:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}} \quad \Rightarrow \quad \mathbf{w} \leftarrow \mathbf{w} + \eta \mathbf{x}^T (y - \text{prediction})$$

In Naive Byaes we estimate all the w's as independent from each other but in logisitic regression the w's are dependent. Logistic Regression won't have an over confidence problem like Naive Bayes has.

1.2.1 Regularized Logistic Regresssion

If we want to prevent overfitting and prevent the components in our vector of parameters w from getting to big, we can constrain our loss function by doing the following:

$$\mathcal{L} = -\sum_i y_i \log p_i + (1 - y_i) \log(1 - p_i) + \frac{1}{2}\lambda^2 \tag{15}$$

Again finding the optimal value of this expression gives us the following:

$$\frac{\partial \mathcal{L}}{\partial w_k} = -\sum_i (y_i - p_i) y_{ik} + \lambda w_k = 0$$

We get the following new update equations in our gradent descent algorithm:

$$w_k \leftarrow w_k - \eta \frac{\partial x}{\partial w_k} \Rightarrow w_k \leftarrow w_k + \eta \sum_i (y_i - p_i) x_{ik} - \lambda w_k$$

$$w_k \leftarrow (1 - \eta\lambda) w_k + \eta \sum_i (y_i - p_i) x_{ik} \tag{16}$$

We are essentially shrinknig our weight before we move against the gradient.

2 Model Evaluation

This section focuses on developing new ideas/measurements and evaluation techniques to better understand the performance of our classifier.

Last time we saw the following definitions of accuracy and calibration:
Accuracy: Fraction of times you predict the correct label. $\frac{1}{N}\sum_i \mathbf{1}(y_i = f(x_i))$
Calibration: how often does a event with predicted probability p occur?
The motivation behind moving beyond the accuracy metric is that in general datasets are heavily imbalanced and this can heavily skew percentages.
We were presented with the confusion matrix which looks like the following figure:

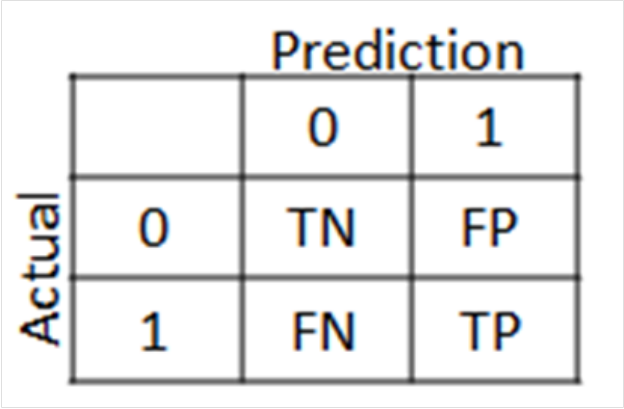

Figure 2: A confusion matrix. *Source: https://computersciencesource.wordpress.com/2010/01/07/year-2-machine-learning-confusion-matrix/*

The variables in the confusion matrix stand for the following:

- **True Positive (TP):** We predict 1 while the class is actually 1 i.e we predict a class accurately to be positive when it is actually 1.
- **False Positive (FP):** We predict 1 while the class is actually 0 i.e we incorrectly predict a class to be positive when it is in fact positive.
- **True Negative (TN):** We predict 0 while the class is actually 0 i.e we correctly predict a class is negative (part of class 0).
- **False Negative (FN):** We predict 0 while the class is actually 1 i.e we incorrectly predict a class that is in fact in class 1 or positve as negative or in class 0.

With this concept in mind, we defined some more terms:

- **Precision:** Fraction of positive predictions that are true

$$\text{Precision } = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **True Positive Rate (TPR):** also known as sensitivity, hit rate, and recall is defined as

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

  This metric corresponds to the proportion of data points that were correctly classified as positive with respect to all positive points.The higher TPR, the fewer positive points our classifier will miss.

- **False Positive Rate (FPR):** also known as fall-out

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

  The proportion of negative points that were incorrectly classified as positive with respect to all negative points. The higher the FPR, the higher we will misclassify negative points.

A high recall classifier can get all the spam thats there but also will tag a lot of legitimate email as spam. Highprecision means low recall which is opposite from low precision and high recall.

2.0.2 ROC Curve

If we vary our threshold and plot the false positive rate against the true positive rate, we develop the ROC curve, which stands for Receiver Operator Characteristic curve. The AUC or Area Under the Curve is equivalent to accuracy for balanced classification.

A ROC curve for a really really good classifier might look like this:

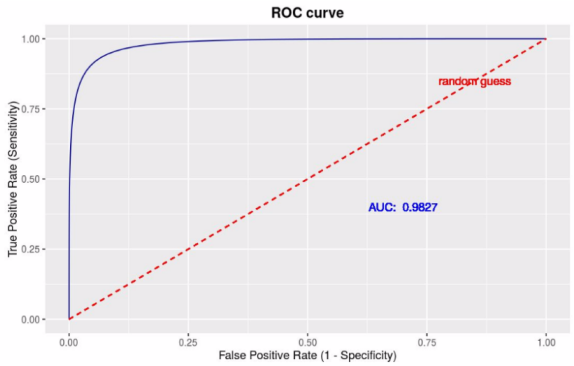

Figure 3: A confusion matrix. *Source: https://github.com/Columbia-Intro-Data-Science/APMAE4990*

For more information please see the github link in the figure description, under the lectures directory.

3 Computational Social SCience: Exciting Progress and Future Challenges

This talk was given by Duncan Watts who is the author of *6 Degrees: Living in a Connected World* as well as other books and currently principal research scientist at Microsoft Research. He was a professor in Sociology in Columbia from 2000 to 2007.

- social phenomena arise when individuals interact to produce collective behavior (agents of these systems can vary: individuals, compaies, nationes, etc.). People comprise these agents at the end of the day and their interactions affect Macro phenomena. There is a micro-macro relationship that is extremely difficult to understand and quantify.
- This has led to an emerging field of Computational Social Science due in large part by the upsurge of big data from web interactions. There has been a dramatic increase in scale, scope, and granularity of data.
  - search, buy, produce, consumption $\Rightarrow$ digital exhaust/bread crumbs
  - the result of all this technological development and digital bread crumbs is computational social science (CSS)
- outline of the talk. a) illustrate CSS with two types of examples, social contagion with big data and small data with virtual labs - can run experiments online (b) emerging difficulties

3.1 Social Contagion using Big Data

a contagion implies large multi-scale, multi-step, peer-to-peer spread and multi-generational branching process (Diffusion, Going Viral, ebola virus)

- usually a S shaped curve implies diffusion but there are lots and lots of models that generate S shaped curves so there are two principal problems here. 1) could result from many processes and 2) typically, our

observation of these processes is only for successful diffusive events. This motivates a strong understanding of unsuccessful attempts of diffusion.

$\Rightarrow$ the web potentially resolves this problem How diverse is going viral? What does viral diffusion look like? Can we predict it?

3.2 Online Diffusion Project

- this project involved created 6 seemingly unrelated projects conducted over the past few years all of which were designed to go viral. They were in different domains which was important and the goal was to see if there were any patterns in the diffusion.

  They ultimately found that diffusion is of course a rare event but that the structure of diffusion is highly similar from one event to another event. There was a lot of consistency in terms of the most common structures.

3.3 Viral or Just Popular - Structural Virality Project

- Big data helps to study rare events. fore each event, one can quantify its structural virality. Shortest paths range from 2 to $\log(N)$

  For popular events, one can ask, what diversity do we see with respect to structure? what is the relationship between size and structural virality?

- they found an incredible diversity of structural virality and that a popular event is not necessarily viral (broadcasting effect e.g Superbowl isnt a viral event)

- popularity is driven mostly by the size of the biggest broadcaster. Indiviudal people can broadcast (celebrity effect)

3.4 How predictable are cascades?

- What features drive cascade size?

- Goal is to predict final number of retweets of tweets with URLs. Restrit to 100 per domain

- features: content, user, content-user interaction $\Rightarrow$ user features are much more predictive than content (interactions add nothing really)

  performance asymptotes around 4 (basis of potential existence of a limit)

  suggests that predictive accuracy may be subject to some fundemental limit

3.5 Virtual Lab

Social media limits the types of questions one can ask. Virtual labs try to tackle more tradiitonal challenges of physical labs such as problems like realism, duration and scale.

Notes from kki2101

1 Naive Bayes

Here we saw Naive Bayes (independent assumption over all words). We predict class c given features.

Bayes Rule:

$$P(c|x) = p(\bar{x}|c) \times p(c)/p(\bar{x}) \tag{17}$$

The naive part says the following:

$$p(\bar{x}|c) = \prod_j p(x_j|c) \tag{18}$$

and then we have:

$$p(c|\bar{x}) = \prod_j p(x_j|c) \times p(c)/p(\bar{x}) \tag{19}$$

$p(x_j|c)$ models a a coinflip (i.e. Bernoulli)

The word occurrences are coinflips:

$$p(x_j|c) = \theta_{jc}^{x_j}(1-\theta_{jc})^{1-x_j} \tag{20}$$

θ_{jc} predicts the jth word in some class c.

$$log(p(c|x)) = \sum_j log[\theta_{jc}^{x_j}(1-\theta_{jc})^{1-x_j}] + log[\frac{\theta_c}{p(\bar{x})}] \tag{21}$$

$$= \sum_j x_j log\frac{\theta_{jc}}{1-\theta_{jc}} + \sum_j log(1-\theta_{jc}) + + log[\frac{\theta_c}{p(\bar{x})}] \tag{22}$$

The leftmost term is the number of words in the document; the middle term is size of the vocab. we are working with.

We have two cases:

$$log\frac{p(c=1|x)}{p(c=0|\bar{x})} = \sum_j log[\frac{\theta_{j1}(1-\theta_{j0})}{\theta_{j0}(1-\theta_{j1})}] + \sum_j log[\frac{(1-\theta_{j1})}{(1-\theta_{j0})}] + log[\frac{\theta_1}{\theta_0}] \tag{23}$$

Let's look at the difference of the log prob. of both of our cases: Lets define:

$$w_j = log[\frac{\theta_{j1}(1-\theta_{j0})}{\theta_{j0}(1-\theta_{j1})}] \tag{24}$$

So we end up with:

$$log\frac{p(c=1|\bar{x})}{p(c=1|\bar{x})} = \bar{x} \cdot \bar{w_j} + \bar{w_0} \tag{25}$$

Calculate θ_j To do this we will take the derivative of the log-likelihood of the probability of seeing n heads:

$$0 = \frac{n}{\theta} + \frac{N-n}{1-\theta} \tag{26}$$

$$\theta_{j1} = \frac{n_{j1}}{n_1} = \frac{\text{num of spam docs w/ word j}}{\text{num of spam docs}} \tag{27}$$

$$\theta_1 = \frac{n_1}{N} = \frac{\text{num of spam docs}}{\text{num of total docs}} \tag{28}$$

2 Logistic Regression

We shall begin here with the predictor we got above:

$$\log \frac{P}{1-p} = w \cdot x \tag{29}$$

$$p = \frac{1}{1+e^{-w \cdot x}} \tag{30}$$

We have a set of documents x_i and a set of labels y_i, we know find the model.

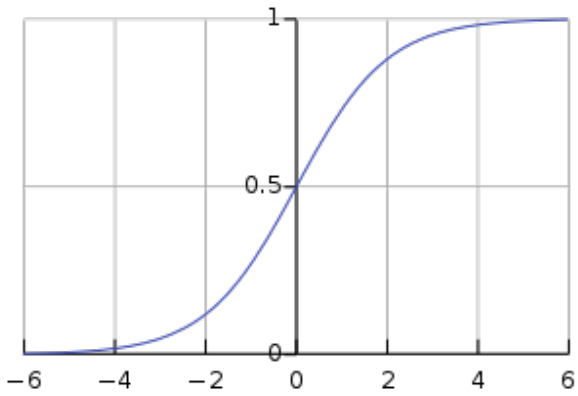

Figure 4: A graph of th Sigmoid Function (from Wikipedia).

$$\text{LL} = P(D|w) = \prod_i p_i^{y_i}(1-p_i)^{1-y_i} \tag{31}$$

Using the above equations and taking the log likelihood we get:

$$L = -\sum_i [y_i w \cdot x - (\log 1 + e^{w \cdot x})] \tag{32}$$

We set the derivative to zero to find the max likelihood and end up with:

$$0 = -\sum_i [y_i \cdot x - (\frac{1}{1+e^{w \cdot x}})] x_{ik} \tag{33}$$

We use Gradient Descent, since no solution exists:

$$w = w - n\frac{\partial L}{\partial w} \tag{34}$$

It is also:

$$w_k = w_k + n\sum_i [(y_i - p_i)x_{ik}] \tag{35}$$

We can regularize the model as well as follows:

$$L = \sum_i [y_i log_i + (1-y_i) log(1-p_i)] + 0.5\lambda ||w||^2 \tag{36}$$

$$\frac{\partial L}{\partial w_k} = \sum_i [(y_i - p_i)x_{ik}] + \lambda w_k \tag{37}$$

We thus end of with:

$$w_k = (1 - n\lambda)w_k + n\sum_i [(y_i - p_i)x_{ik}] \tag{38}$$

3 Evaluation

When evaluating various classifiers we can consider a number of things:

- Accuracy: The fraction of times we predict the correct label.
- Calibration: This is how often an event with predicted probability p occurs.
- Confusion Matrix:

$$p = \frac{1}{1 + e^{-w \cdot x}} \tag{39}$$

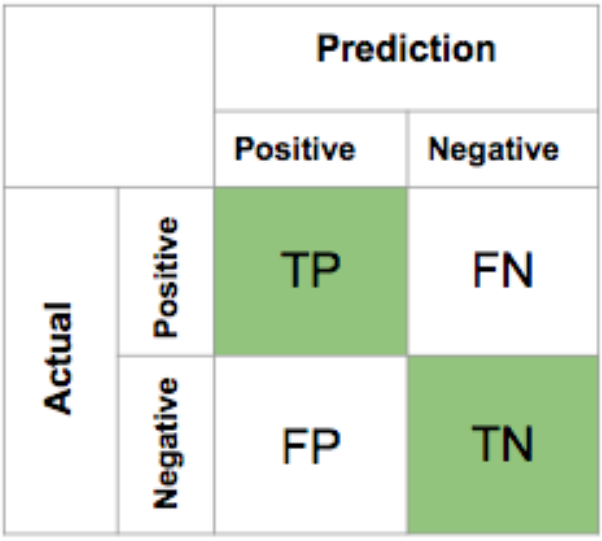

Figure 5: A Confusion Matrix (from WS02).

- Receiver Operating Characteristic (ROC) curve: ROC curve plots the true positive rate and the false positive rate

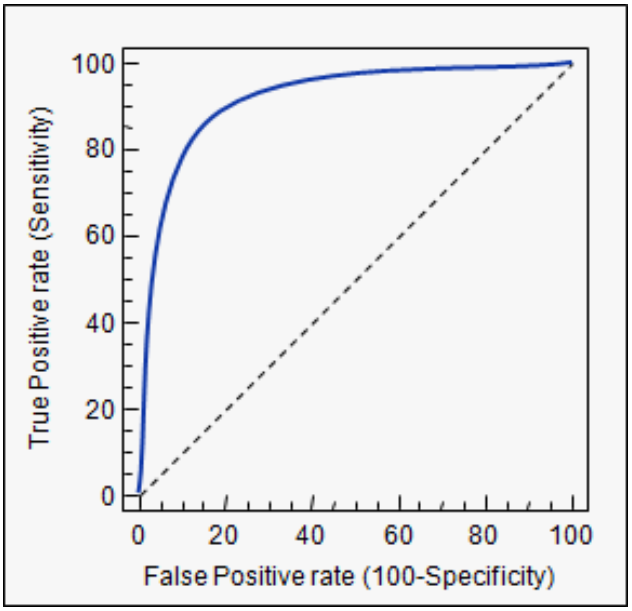

Figure 6: ROC Curve (from Wikipedia).

Notes from np2577

1 Naive Bayes

Given feature $x \in \{0,1\}^K$, predict classes $c \in C$
where:
$x = x_1, x_2, ...x_K$ and $x_j = 1$ if word j is in the document.
K=size of vocabulary: approx 100k-500k

Bayes Rule:

$$p(c|x) = \frac{p(x|c)p(c)}{p(x)} \tag{40}$$

"Naive" Independence Assumption: x_j's are conditionally independent given class c i.e

$$p(x|c) = \prod_{j=1}^{K} p(x_j|c) \tag{41}$$

Furthermore we assume that given class c, word occurrences are coin flips(have Bernouilli distribution) i.e

$$p(x_j|c) = \theta_{jc}^{x_j}(1-\theta_{jc})^{1-x_j} \tag{42}$$

and $p(c) = \theta_c$.

Putting this all together and taking logs we get:

$$log(p(c|x)) = \sum_j log\big(\theta_{jc}^{x_j}(1-\theta_{jc})^{1-x_j}\big) + log\frac{\theta_c}{p(x)} \tag{43}$$

$$log(p(c|x)) = \sum_j x_j log\frac{\theta_{jc}}{1-\theta_{jc}} + \sum_j log(1-\theta_{jc}) + + log\frac{\theta_c}{p(x)} \tag{44}$$

How can we compute $p(x)$? We could use $p(x) = \sum_c p(x, c)$ but this will be cumbersome

Lets restrict ourselves to two classes, 0 and 1.
We want to compare $p(c = 1|x)$ with $p(c = 0|x)$.
Now $p(c = 1|x) > p(c = 0|x)$ iff $log\frac{p(c=1|x)}{p(c=0|x)} > 0$

$$log\frac{p(c = 1|x)}{p(c = 0|x)} = log(p(c = 1|x)) - log(p(c = 0|x)) \tag{45}$$

$$log\frac{p(c = 1|x)}{p(c = 0|x)} = \sum_j x_j log\frac{\theta_{j1}(1 - \theta_{j0})}{\theta_{j0}(1 - \theta_{j1})} + \sum_j log\frac{(1 - \theta_{j1})}{(1 - \theta_{j0})} + log\frac{\theta_1}{\theta_0} \tag{46}$$

$$log\frac{p(c = 1|x)}{p(c = 1|x)} = w \cdot x + w_0 \tag{47}$$

Note: We can precompute w_0 as it doesn't depend on x.
Thus to predict class for any x we simply need to calculate $w \cdot x + w_0$ and predict 1 iff $w \cdot x + w_0 > 0$. This will take time proportionate to the number of unique words in the document.

The next question is, how can we learn parameters θ_{jc} from the training data? As usual we use MLE
Let's consider the general case of tossing a coin with probability of heads equal to θ. If we flip the coin N times and observe n heads then what is the MLE for θ?

$$P(data|\theta) = C\theta^n(1 - \theta)^{N-n} \text{ for some constant C.} \tag{48}$$

$$\text{Log-likelihood} = L = log(P(D|\theta)) = logC + nlog(\theta) + (N - n)log(1 - \theta) \tag{49}$$

$$\frac{\partial L}{\partial \theta} = \frac{n}{\theta} - \frac{N - n}{1 - \theta} = 0 \tag{50}$$

$$n(1 - \theta) = \theta(N - n) \tag{51}$$

$$\theta = \frac{n}{N} = \text{fraction of heads} \tag{52}$$

Therefore the MLE for parameters:

$$\theta_{j1} = \frac{n_{j1}}{n_1} = \frac{\text{No. of spam docs with word j}}{\text{No. of spam docs}} \tag{53}$$

and

$$\theta_1 = \frac{n_1}{N} = \frac{\text{No. of spam docs}}{\text{No. of total docs}} \tag{54}$$

Runtime analysis
N:Number of documents
K: Number words in vocabulary
$\bar{k}$: average number of words in a document
Training time: $O(N\bar{k})$, Run through all the documents twice, once to create the dictionary and once to calculate counts.
Storage: $O(K|C|)$
Naive Bayes can be updated in an online manner. As we get new data we can just update counts. No need to retrain the full model.

2 Logistic Regression

In Naive Bayes we got the following equation

$$log\frac{p(c=1|x)}{p(c=0|x)} = w \cdot x \tag{55}$$

(we can incorporate the intercept w_0 into w by adding a 1 to each x) What if we started off with a predictor that has this form(linear log odds) and unknown weights.

$$log\frac{p}{1-p} = w \cdot x \tag{56}$$

$$\frac{p}{1-p} = e^{w \cdot x} \tag{57}$$

$$p = (1-p)e^{w \cdot x} \tag{58}$$

$$p(1+e^{w \cdot x}) = e^{w \cdot x} \tag{59}$$

$$p = \frac{1}{1+e^{-w \cdot x}} \tag{60}$$

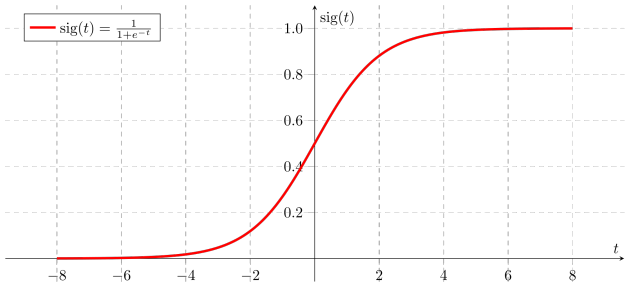

Figure 7: source: https://upload.wikimedia.org/wikipedia/commons/thumb/5/53/Sigmoid-function-2.svg/2000px-Sigmoid-function-2.svg.png

Key equations:

$$p(x|w) = \frac{1}{1+e^{-w \cdot x}} \tag{61}$$

$$1 - p(x|w) = \frac{1}{1+e^{w \cdot x}} \tag{62}$$

$$log\frac{p}{1-p} = w \cdot x \tag{63}$$

We have a set of documents x_i and set of labels y_i.

$$\text{Log-likelihood} = P(data|\theta) = \prod_i p_i^{y_i}(1-p_i)^{1-y_i} \tag{64}$$

$$L = -log(P(D|\theta)) = -\sum_i \{y_i log p_i + (1-y_i)log(1-p_i)\} \tag{65}$$

$$L = -\sum_i \{y_i log \frac{p_i}{1-p_i} + log(1-p_i)\} \tag{66}$$

$$L = -\sum_i \{y_i(w \cdot x_i) - log(1 + e^{w \cdot x_i})\} \tag{67}$$

$$\frac{\partial L}{\partial w_k} = -\sum_i \{y_i x_{ik} - \frac{1}{1+e^{w \cdot x_i}} e^{w \cdot x_i} x_{ik}\} = 0 \tag{68}$$

$$-\sum_i \{(y_i - \frac{1}{1+e^{-w \cdot x_i}}) x_{ik}\} = 0 \tag{69}$$

$$-\sum_i \{(y_i - p_i) x_{ik}\} = 0 \tag{70}$$

Unfortunately this has no closed form solution form for w. Thus we use gradient descent.

$$w = w - \eta \frac{\partial L}{\partial w} \tag{71}$$

$$w = w + \eta X^T (y - p) \tag{72}$$

or

$$w_k = w_k + \eta \sum_i \{(y_i - p_i) x_{ik}\} \tag{73}$$

2.1 Regularization

We can add a regularization term to the loss function.

$$L = -\sum_i \{y_i log p_i + (1-y_i) log(1-p_i)\} + \frac{1}{2}\lambda ||w||^2 \tag{74}$$

$$\frac{\partial L}{\partial w_k} = -\sum_i \{(y_i - p_i) x_{ik}\} + \lambda w_k \tag{75}$$

The gradient descent update will be

$$w_k = (1 - \eta\lambda) w_k + \eta \sum_i \{(y_i - p_i) x_{ik}\} \tag{76}$$

Therefore we shrink the weight before we move against the gradient.

3 Evaluation

Last time
Accuracy: Fraction of time we predict the right label
Calibration: How often does an event with predicted probability p occur

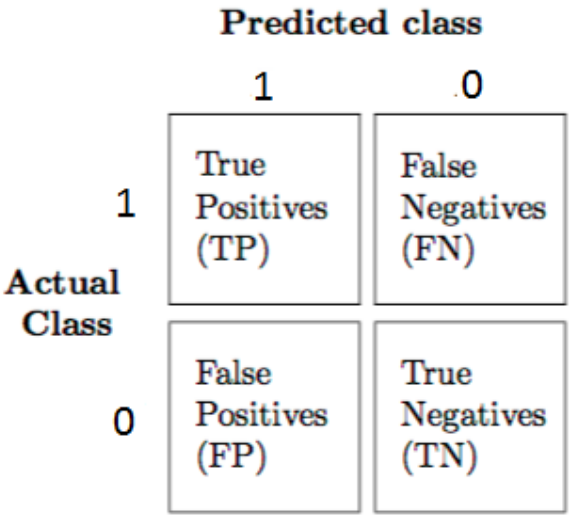

Figure 8: Confusion Matrix

3.1 Confusion Matrix

Thus, Accuracy=$\frac{TP+TN}{TP+TN+FP+FN}$

Some other metrics:
Precison: Fraction of positive predictions that are true. i.e
Precison=$\frac{TP}{TP+FP}$
Recall/True Positive Rate: Fraction of true examples that are predicted to be positive. i.e
Recall=$\frac{TP}{TP+FN}$

In different situations we might want a model with high precision or high recall. In the case where class 1 is marking a document as spam if we want to ensure that we do not mark any email that is not spam as spam then we want a model with high precision. On the other hand if we want to ensure we get all the spam then we want a model with high recall. Another example is a situation where we want a model that classifies a manager as good or bad. If a manager is marked as bad we can take some action. If the action is strong then we want a model with high precision as it is very bad if we take an action against a manager who is in fact good(false positive). If the action is mild such as sending a mail about a management course then we want a model with high recall as it is important to ensure every bad manager gets a mail and not a big deal if a good manager gets the mail.
False Positive Rate: Fraction of false examples that are predicted to be positive. i.e
False Positive Rate=$\frac{FP}{FP+TN}$

Receiver Operator Characteristic(ROC) Curve: Plot of True Positive Rate vs False Positive Rate
Area under ROC Curve (AUC): Can be shown to be equivalent to accuracy for balanced classification

4 Computational Social Science: Exciting Progress and Future Challenges: A talk by Duncan Watts

Micro-macro problem: Things studied in social sciences tend to be about collective phenomena. For example we study and try to understand entities like families, firm, political parties etc. We study them as if they are actors and have certain beliefs and preferences. This is mostly just out of convenience. In reality entities are comprised of large number of individual people who have their own beliefs, intentions and actions. The connection between micro level of individual people and macro level of social actors and facts is central to what drives social sciences.

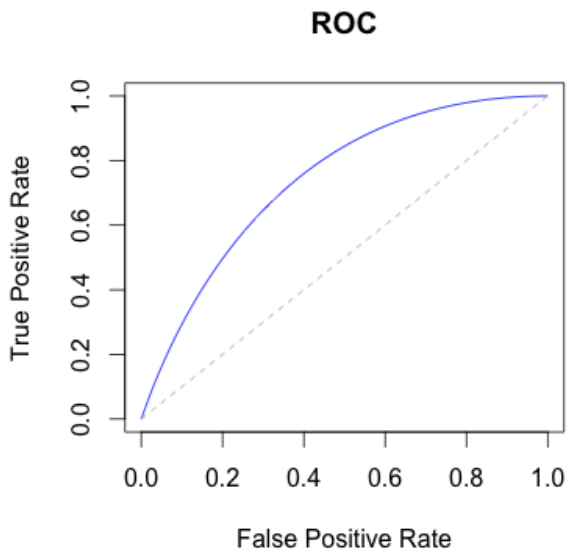

Figure 9: source: http://danielnee.com/2014/08/area-under-the-receiver-operator-curve-auc/

However this is difficult to study empirically as we need to collect data about all individuals, interactions and collective behavior. This is a daunting challenge even if we just want to observe phenomena.To study cause and effect we need to do something that resembles an experiment which is impossible.

The digital revolution over the last 20 years has started to lift some of these barriers. We can now see:

- Increase in scope, scale and granularity of data. Can obtain data from ecommerce, search websites, social media, social networks
- Increase in speed and scale of experiments. Can run experiments in "virtual labs" online. Earlier restricted to mostly undergraduate students in a psychology lab for a couple of hours.

4.1 Study of social Contagion using "Big Data"

Contagion implies large multi-step peer to peer spread. If we plot time on x axis and total number of adopters aggregated over population on the y axis we observer a S shaped curve. Traditionally we claim that this curve is an indicator of diffusion: there are a few early adopters, then there is an exponential increase and eventually it slows down as the entire population has adopted. However there are 2 problems with this argument:

- An S shaped curve could be from many processes
- It is only collected for "successful" diffusion events

To verify the claims that there is an underlying diffusion process we need individual data and also study unsuccessful events.

4.1.1 Case Study

Dr Watts and his team studied diffusion phenomena over 6 different forums such as twitter, facebook(at that time one could make a facebook app and they made some game that a user could play and recommend to friends), zync(yahoo video app that allows users to watch videos together and chat). The assumption was that each of these platforms would have different biases. However they observed a huge similarity across all of them.

Looking specifically into the twitter data they did some more analysis. The first observation was that almost everything doesn't go viral. They looked into the differences between popular and viral(please see slides on "The structural virality of online diffusion" from lecture 1). They also tried to determine if the most popular things were more broadcast-y or more viral. It appears that it can be either as they observed very different structures for different things of roughly equal popularity. There also seems to be zero correlation between size and structural virality. They observed that as things get bigger they do not become more viral. They are mostly driven by the largest broadcast. These tend to be retweets by a celebrity.

They designed a model to predict the number of retweets for a given tweet by some user. Some of the features used were number of followers, number of friends, how long they have been on twitter, how active they are etc. They also classified tweets and users into topics and include interaction between topic of person(what field is the user know for) and tweet. They obtained the best results using Random Forest Model with R squared of 0.4. Two main observations:

- Content doesn't matter. They were unable to identify any content related features that had a correlation with output
- User matters alot. Simple regression model with one feature based on average number of retweets in the past gave results which were almost as good.

They tried adding many features but the results seem to asymptote at 0.4 R Squared. It raises the questions if there is some intrinsic limit on how much we can predict. The threshold between something dying out vs becoming viral is a source of great study. However everything is dying out all the time and thus it is hard to model social contagion.

An important takeaway is that for an analysis like this which involves studying rare events at scale enormous data is not just helpful but necessary.

Papers:

- S Goel, D Goldstein, D Watts, *The Structure of Online Diffusion Networks*, http://dl.acm.org/citation.cfm?id=2229058
- S Goel, A Anderson, J Hofman, D Watts, *The Structural Virality of Online Diffusion*, https://www.microsoft.com/en-us/research/publication/the-structural-virality-of-online-diffusion/
- A Anderson, J Hofman, T Martin, A Sharma, D Watts, *Exploring limits to prediction in complex social systems*, https://www.microsoft.com/en-us/research/publication/exploring-limits-to-prediction-in-complex-social-systems-predicting-cascade-size-on-twitter/

4.2 Virtual Labs

Social media data limits the types of questions one can ask. To study questions like "How do people co-operate" one needs to do experiments. In a lab we can ask questions we care about and also have control over all the variables. However they are limited and artificial because

- Number of people and time is limited. Lab experiments typically last one hour. Social processes do not play out in one hour
- The sample is unrepresentative. Mostly consist of undergraduate students in western universities.

Virtual labs tries to tackle these issues of realism, duration and scale.

4.2.1 Scale: Music Lab

Aim: Study cultural markets and understand why some things are more successful than others. Before starting the experiment their conjecture was that success is driven by social influence. They made a website with 48 songs, one each by different unknown bands. They got 14k teenagers to come to the site, listen to song and rate/download if they want. They ran two versions of the experiment. In the first the user only saw the name of the songs. In the second the user could see ratings and how many times a song had been downloaded. They observed that popular things become more popular. It is hard to predict what will become popular as there is

some intrinsic unpredictability in the system. This is a key differentiator in culural markets. They are not just revealing preferences over time. The markets are constructing preferences of people as they reveal them.

It is important to note that they got 14k people to participate in an experiment which is impossible in a traditional lab.

Paper: P Dodds, M. Salganik, D Watts, *Experimental Study of Inequality and Unpredictability in an Artificial Cultural Market*, http://science.sciencemag.org/content/311/5762/854

4.2.2 Realism: Crisis Map

During crisis UN bodies parachute into the affected areas and try and organize relief operations. It is usually difficult to know what is happening on the ground. The Standby Task Force monitors social and regular media online and identifies posts with useful information, classifies and geo-tags them. This is called a crisis map. The idea is that most people have smartphones and are constantly posting things online. Some of the posts include useful information about specific damaged roads, bridges etc.

Dr Watts and his team tried to build a crisis map in a lab. They examined 1600 tweets from Philippians when typhoon Pablo hit in December 2012. They used Mechanical Turks to try and recreate the map created by the Standby Task Force. In one hour they were able to make a map similar to the one created over 24 hours in 2012. The question they were looking to study was "How many people should one recruit for such a task?". The literature claims that as the number of people increase the effort per person decreases. However more people leads to increase in collaboration which can have a positive effect. They observed that coordination does outweigh effort and that larger groups did better even on a per capita basis.

It is important to note that this experiment was a real world task conducted in a lab which allowed them to control and ask questions about the relationship between size and performance.

Paper: A Mao, W Mason, S Suri, D Watts, *An Experimental Study of Team Size and Performance on a Complex Task*, http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0153048

4.2.3 Time: Repeated Prisoner's Dilemma

Two players are paired randomly and play 10 rounds of prisoners dilemma. Traditionally this experiment runs for an hour and people play around 20 games each. In any one game people usually start out by cooperating and defect towards the end of the game. The literature conjectures that when this game is played repeatedly people start defecting in earlier and earlier rounds and eventually it unravels completely and every player defects in round 1. In a lab we see a litle bit of unraveling as people go from defecting on round 9 to 8 and 7. However since there are only 20 games the experiment ends before it completely unravels.

Dr Watts and his team ran this experiment over various days. They got a 100 mechanical turks to play 20 games every day for 20 consecutive working days. Their results contrast the conjecture that there would be no cooperation beyond a point. They observed that for the first few days it did unravel. However after 7 days it stops and asymptotes and remains stable for the remaining 13 days. On further analysis they observed that around 2/3 of the people were rational cooperators. They start out by cooperating but eventually defect as they do not want to get exploited. The remaining 1/3 always cooperate till the other person defects after which they defect as well. They had the same strategy throughout and never defected first in any game. This second group acts like a break on group 1. A person in group 1 doesn't know what person they will run into. They don't want to get locked into a cycle of endless defection but they also don't want to get exploited. Thus group 2 people benefit everyone. However this is at a cost to themselves as their payoff are lower than of a group 1 person. Many of them seemed to be aware of this. The players filled out exit surveys in which they described their strategies. Some of the people in group 2 said that they never defected first as they felt guilty doing so. Some of the others said that they realized that if they started defecting the whole thing would unravel and everyone would be much worse off. Thus they seem to be long term rational players.

It is important to note that this experiment was novel that it involved the same 100 people participating every day for 20 days.

Paper: A Mao, L Dwarkin, S Suri, D Watts, *Resilient Cooperators Stabilize Long-Run Cooperation in the Finitely Repeated Prisoner's Dilemma*, https://papers.ssrn.com/sol3/Papers.cfm?abstract_id=2756249

Notes from syh2115

1 Naive Bayes

1.1 Basic Ideas

Say we want to predict a class c given features x. For example, say we want to classify emails as spam or not spam. Our classes c would be $\{0, 1\}$ where 0 represents not spam and 1 represents spam. Our features x would consist of values $\in \{0, 1\}$ representing whether a word is present or not.

What we want to predict is $p(c|x)$ or the probability of a class c given the features x.

Using Bayes' Rule we can get:

$$p(c|x) = \frac{p(x|c)p(c)}{p(x)} \tag{77}$$

Here, $p(x|c)$ represents the probability of seeing all words given the class c.

We also make the "naive" assumption of independence and assume that each word occurrence $p(x_i|c)$ is a "coin flip" with a Bernoulli distribution. Thus we get:

$$p(c|x) = \frac{\prod p(x_i|c)p(c)}{p(x)} \tag{78}$$

$$p(x_j|c) = \theta_{jc}^{x_j}(1 - \theta_{jc})^{1-x_j} \tag{79}$$

Here, θ is just shorthand for probabilities s.t. θ_{jc} is the probability of the jth word in class c.

If we take logs then we get:

$$\log p(c|x) = \sum_j \log[\theta_{jc}^{x_j}(1 - \theta_{jc})^{1-x_j}] + \log \frac{\theta_c}{p(x)} \tag{80}$$

$$\log p(c|x) = \sum_j x_j \log \frac{\theta_{jc}}{1 - \theta_{jc}} + \sum_j \log(1 - \theta_{jc}) + \log \frac{\theta_c}{p(x)} \tag{81}$$

Where, using the product rule, we can get:

$$p(x) = \sum_c p(x, c) = \sum_c p(x|c)p(c) \tag{82}$$

Looking quickly at the computational cost for this equation, we know that the first term is dependent on the number of words present in the document we're looking at, and the second term is dependent on the size of the vocabulary.

If there are two classes, with the possible values of c being 1 or 0, then we can represent the Naive Bayes classifier in log odds form as follows:

$$\log \frac{p(c = 1|x)}{p(c = 0|x)} \tag{83}$$

If the log odds is greater than 0, then we predict $c = 1$, and if it's less than 0, we predict $c = 0$.

Using our equation (**??**) we get:

$$\log \frac{p(c = 1|x)}{p(c = 0|x)} = \sum_j x_j \log \frac{\theta_{j1}(1 - \theta_{j0})}{\theta_{j0}(1 - \theta_{j1})} + \sum_j \log \frac{1 - \theta_{j1}}{1 - \theta_{j0}} + \log \theta_1\theta_0 \tag{84}$$

We can assign the following values to simplify our classifier:

$$w_j = \log \frac{\theta_{j1}(1 - \theta_{j0})}{\theta_{j0}(1 - \theta_{j1})} \tag{85}$$

$$w_0 = \sum_j \log \frac{1-\theta_{j1}}{1-\theta_{j0}} + \log \theta_1 \theta_0 \tag{86}$$

$$\log \frac{p(c=1|x)}{p(c=0|x)} = \mathbf{w} \cdot \mathbf{x} + w_0 \tag{87}$$

If we precompute w_0 then the computational cost is just the number of nonzero words.

1.2 Finding θ_j

To explain how to find individual θ values for our classifier, we did a derivation of the probability of seeing n heads for N coin flips, since a similar concept is applied to the probabilities of seeing words in documents, or other applications of the classifier.

We have the the probability of seeing n heads given θ is:

$$p(n|\theta) = C\theta^n (1-\theta)^{N-n} \tag{88}$$

We can take the log for likelihood, then take the derivative and set to zero to find the max likelihood.

$$\mathcal{L} = \log p(n|\theta) = \log C + n \log \theta + (N-n) \log(1-\theta) \tag{89}$$

$$0 = \frac{\partial \mathcal{L}}{\partial \theta} = \frac{n}{\theta} + \frac{N-n}{1-\theta} \tag{90}$$

$$\frac{n}{\theta} = \frac{N-n}{1-\theta} \tag{91}$$

$$n - n\theta = N\theta - n\theta \tag{92}$$

$$\theta = \frac{n}{N} \tag{93}$$

Using this, then we can get:

$$\theta_{j1} = \frac{n_{j1}}{n_1} = \frac{\#\text{ of spam docs w/ word j}}{\text{total }\#\text{ of spam docs}} \tag{94}$$

$$\theta_1 = \frac{n_1}{N} = \frac{\#\text{ of spam docs}}{\text{total }\#\text{ docs}} \tag{95}$$

1.3 Time and Space

With N documents, c classes, and k different words, the training time for a Naive Bayes classifier is $O(N \cdot \overline{k})$ where $\overline{k}$ is the average number of words in a document. The required space *seems* like it would be $O(k \cdot c)$ but storing using a sparse data structure would save space.

2 Logistic Regression

2.1 The Logistic Function

Say we start with a predictor that has the same form as the Naive Bayes classifier that we ended with:

$$\log \frac{p}{1-p} = \mathbf{w} \cdot \mathbf{x} \tag{96}$$

(Note: There is an implicit intercept in this equation that we are ignoring.)

Using this predictor, we can get some useful values for $p(x|w)$ and $1 - p(x|w)$:

$$\frac{p}{1-p} = e^{\mathbf{w}\cdot\mathbf{x}} \tag{97}$$

$$p = e^{w\cdot x} - pe^{w\cdot x} \tag{98}$$

$$p(1 + e^{w\cdot x}) = e^{w\cdot x} \tag{99}$$

$$p = \frac{e^{w\cdot x}}{1 + e^{w\cdot x}} \tag{100}$$

$$p(x|w) = \frac{1}{1 + e^{-w\cdot x}} \tag{101}$$

$$1 - p(x|w) = 1 - \frac{1}{1 + e^{-w\cdot x}} = \frac{1}{1 + e^{w\cdot x}} \tag{102}$$

Equation **??** is a sigmoid function that when plotted gives us the S-shaped logistic curve, as seen below:

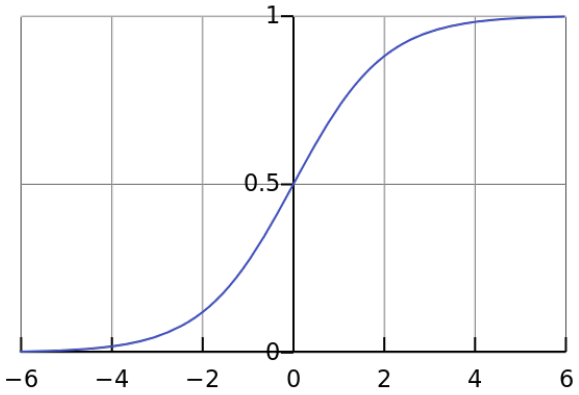

Figure 10: https://en.wikipedia.org/wiki/Sigmoid_function

2.2 Determining the Model

Say we have a set of labels $y_i \in 0, 1$ where 1 represents spam and 0 represents ham, and a set of documents x_i. Then we get likelihood as follows, where $p_i = p(x_i|w)$:

$$p(D|w) = \prod_i p_i^{y_i}(1 - p_i)^{1-y_i} \tag{103}$$

Taking the log likelihood, we get:

$$\mathcal{L} = -\log p(D|w) \tag{104}$$

$$\mathcal{L} = -\sum_i \{y_i \log p_i + (1 - y_i) \log(1 - p_i)\} \tag{105}$$

$$\mathcal{L} = -\sum_i \{y_i \log \frac{p_i}{1 - p_i} + \log(1 - p_i)\} \tag{106}$$

From **??** and **??** above, we can get:

$$\mathcal{L} = -\sum_i \{y_i w \cdot x - \log(1 + e^{w \cdot x})\} \tag{107}$$

Then finding maximum likelihood:

$$0 = \frac{\partial \mathcal{L}}{\partial w_k} = -\sum_i \{y_i x_{ik} - \frac{1}{1 + e^{w \cdot x_i}} e^{w \cdot x_i} x_{ik}\} \tag{108}$$

Using equation **??** we get:

$$0 = -\sum_i \{y_i - p_i\} x_{ik} \tag{109}$$

$$0 = -\sum_i \{y_i - \frac{1}{1 + e^{-w \cdot x}}\} x_{ik} \tag{110}$$

There is no closed form solution to this equation, so we use gradient descent:

$$w \leftarrow w - \eta \frac{\partial \mathcal{L}}{\partial w} \tag{111}$$

In matrix form we get:

$$w \leftarrow w + \eta X^T (y - p) \tag{112}$$

Which can also be represented in component form as:

$$w_k \leftarrow w_k + \eta \sum_i (y_i - p_i) x_{ik} \tag{113}$$

2.3 Side Note: Regularized Logistic Regression

If we don't want the weights in our logistic regression model to get too big, we can use regularization. Then our likelihood equation becomes:

$$\mathcal{L} = -\sum_i \{y_i \log p_i + (1 - y_i) \log(1 - p_i)\} + \frac{1}{2} \lambda ||w||^2 \tag{114}$$

$$\frac{\partial \mathcal{L}}{\partial w} = -\sum_i (y_i - p_i) x_{ik} + \lambda w_k \tag{115}$$

The last term in (**??**) is our regularization term.
Then our update rule becomes:

$$w_k \leftarrow w_k - \eta \frac{\partial \mathcal{L}}{w_k} = w_k + \eta \sum_i (y_i - p_i) x_{ik} - \lambda w_k \tag{116}$$

$$w_k \leftarrow (1 - \eta\lambda) w_k + \eta \sum_i (y_i - p_i) x_{ik} \tag{117}$$

Essentially what we're doing in the update is shrinking the weights first before moving against the gradient the updating our weights.

3 Evaluating Classifiers

There are a couple of different metrics and terms that are often used to evaluate classifiers:

- **accuracy**: The fraction of times you predict the correct label.
- **calibration**: Measures how often an event with predicted probability p occurs.
- **confusion matrix**: a way of representing predicted and actual values.

	predicted	
actual	1	0
1	true positive (TP)	false negative (FN)
0	false positive (FP)	true negative (TN)

- **precision**: The fraction of positive predictions that are true. Can also be represented as $\frac{\text{TP}}{\text{TP+FP}}$
- **recall** (aka true positive rate): fraction of true examples that we predict to be positive. $\frac{\text{TP}}{\text{TP+FN}}$
- **false positive rate**: fraction of false examples that are predicted positive. $\frac{\text{TP}}{\text{TP+FN}}$
- **receiver operator characteristic curve (ROC)**: we can plot the TPR (probability of true detection) vs. FPR (probability of a false alarm) and by changing our probability threshold, we can get a curve, as shown below:

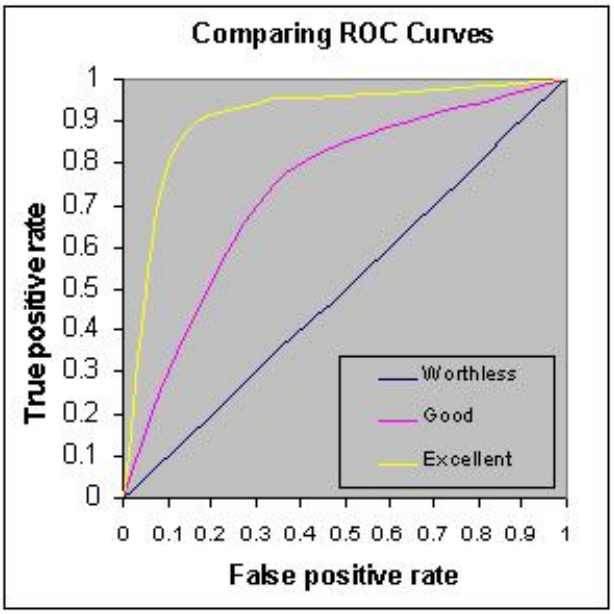

Figure 11: http://gim.unmc.edu/dxtests/roc3.htm

- **area under curve (AUC)**: the area under the ROC curve is equivalent to the accuracy for a balanced classification

4 "Guest Lecture": Computational Social Science: Exciting Progress and Future Challenges

Speaker: Duncan Watts

4.1 Introduction

A problem of particular interest in social science is the **"micro-macro" or emergence problem**. This is the phenomena of collective social behavior that spawns from an individual level. Unfortunately, it's hard to study this problem empirically, especially on a "macro" scale. However, the advent of the Internet means that there is a dramatic increase in the scale, scope, and granularity of data available. Web platforms also allow for an increase in the speed and scale of experiments.

In this talk, we looked at computational social science from two main examples: "big data" social contagion and "small data" virtual lab experiments.

4.2 Social Contagion using Big Data

In this section of the talk, we look at how things go viral. When we talk about "viral" or "social contagion," our concept of this term comes from biological/epidemiological ideas of a multigenerational branching spread. But historically, S-shaped curves have always been used to represent social data and infer "diffusion" events and virality.

There are two main problems with this historical approach:

1. An S-shaped curve can result from many different processes (e.g. marketing efforts, population heterogeneity)
2. We usually only see and collect data for successful diffusion events, so we have few examples of unsuccessful ones.

Exploring social contagion requires the use of individual-level data and data from unsuccessful attempts. The web helps solve some of these problems. The following are some examples of projects that involve the web, big data, and social contagion.

4.2.1 Online Diffusion Project (2012: Goel, Watts, Goldstein)

Associated Paper: S. Goel, D. J. Watts, and D. G. Goldstein, The structure of online diffusion networks. In Proceedings of the 13th ACM Conference on Electronic Commerce (EC '12). 2012.

This project involved 6 unrelated projects that were all designed in an attempt to go viral. There were consistent diffusion patterns across all the projects. They found that 99% of adoptions are ≤ 1 "hop" from the seed, or origin/source, suggesting that there isn't much virality in spreading.

4.2.2 "Structural Virality" Project (2015: Goel, Anderson, Hofman, Watts)

Associated Paper: S. Goel, A. Anderson, J. Hofman, and D. J. Watts, The Structural Virality of Online Diffusion. Management Science 62(1):180-196. 2015.

This project looked at every video, news story, image, and petition on Twitter over a timespan of 12 months. This was about 1.4 billion different "events". They restricted this dataset to only "popular" things by taking only events that had over 100 retweets, resulting in about 350,000 items to look at.

They looked at the average shortest path length for nodes in the viral structure to classify the structure. Structures with an average shortest path length of 2 were "broadcast" and those with an average shortest path length of log(n) were "viral".

What they found was that there are all sorts of patterns and structures in the "virality" of a viral event's spread that is unrelated to time scale. They found that popularity $\neq$ virality. Popularity is usually driven by celebrities, or the individuals that are the largest source of broadcasting.

4.2.3 How Predictable are Cascades? (2016: Martin, Hofman, Sharma, Anderson, Watts)

Associated Paper: T. Martin, J. Hofman, A. Sharma, A. Anderson, and D. J. Watts, Exploring Limits to Prediction in Complex Social Systems. Proceedings of the 25th ACM International World Wide Web Conference (WWW). 2016.

This paper looked at whether popularity/cascade size is predictable. What they found was that using only past success as a predictor could get the same result as a more complex model with more features. This suggests that there's an asymptotical limit to how well and how much popularity/cascade size can be predicted, and that perhaps sociological theories are not enough to predict behavior in these cases because there is too much randomness.

4.3 Virtual Labs

Traditional physical behavioral lab experiments have a lot of limitations (validity in external environments, expensive, slow.) Virtual labs allow for more realism, scale, and duration in the experimental design space.

We went over 3 examples of how virtual labs allow for expansion in scale, realism, and time.

4.3.1 Scale: Music Lab (2006: Salganik, Dodds, Watts)

Associated Paper: M. J. Salganik, P. S. Dodds, and D. J. Watts, "Experimental study of inequality and unpredictability in an artificial cultural market." Science, 311:854-856, 2006.

The Music Lab was an experiment that looked at social influence and market dynamics, and involved getting thousands of volunteers recruited from social media to participate in a web app that was designed expressly for the purposes of the experiment. This experiment is an example of how virtual labs allow for massive increases in the scale of an experiment.

4.3.2 Realism: Crisis Mapping in the Lab (2016: Mao, Mason, Suri, Watts)

Associated Paper: A. Mao, W. Mason, S. Suri, and D. J. Watts, An Experimental Study of Team Size and Performance on a Complex Task. PLoS ONE 11(4). 2016.

4.3.3 Time: Month-long Prisoner's Dilemma Experiment

Associated Paper: A. Mao, L. Dworkin, S. Suri, D. J. Watts, Resilient cooperators stabilize long-run cooperation in the finitely repeated Prisoner's Dilemma. 2017.