Bayesian Learning

- A powerful approach in machine learning
- Combine data seen so far with prior beliefs
  - This is what has allowed us to do machine learning, have good inductive biases, overcome "No free lunch", and obtain good generalization on novel data
- We use it in our own decision making all the time
  - You hear a word which which could equally be “Thanks” or “Hanks”, which would you go with?
    - Combine Data likelihood and your prior knowledge
  - Texting Suggestions on phone
  - Spell checkers, etc.
  - Many application

Bayesian Classification

- $P(c|x)$ - Posterior probability of output class c given the input x
- The discriminative learning algorithms we have learned so far try to approximate this directly
- $P(c|x) = P(x|c)P(c)/P(x)$ Bayes Rule
- Seems like more work but often calculating the right hand side probabilities can be relatively easy
- $P(c)$ - Prior probability of class c – How do we know?
  - Just count up and get the probability for the Training Set – Easy!
- $P(x|c)$ - Probability "likelihood" of data vector x given that the output class is c
  - If x is nominal we can just look at the training set and again count to see the probability of x given the output class c
  - There are many ways to calculate this *likelihood* and we will discuss some
- $P(x)$ - Prior probability of the data vector x
  - This is just a normalizing term to get an actual probability. In practice we drop it because it is the same for each class c (i.e. independent), and we are just interested in which class c maximizes $P(c|x)$.

Bayesian Classification Example

- Assume we have 100 examples in our Training Set with two output classes Good and Bad, and 80 of the examples are of class good.
- Thus our priors are:

Bayesian Classification Example

- Assume we have 100 examples in our Training Set with two output classes Good and Bad, and 80 of the examples are of class good.
- Thus our priors are:
  - $P(Good) = .8$
  - $P(Bad) = .2$
- $P(c|x) = P(x|c)P(c)/P(x)$ Bayes Rule
- Now we are given an input vector x which has the following likelihoods
  - $P(x|Good) = .3$
  - $P(x|Bad) = .4$
- What should our output be?

Bayesian Classification Example

- Assume we have 100 examples in our Training Set with two output classes Good and Bad, and 80 of the examples are of class good.
- Thus our priors are:
  - $P(Good) = .8$
  - $P(Bad) = .2$
- $P(c|x) = P(x|c)P(c)/P(x)$ Bayes Rule
- Now we are given an input vector x which has the following likelihoods
  - $P(x|Good) = .3$
  - $P(x|Bad) = .4$
- What should our output be?
- Try all possible output classes and see which one maximizes the posterior using Bayes Rule: $P(c|x) = P(x|c)P(c)/P(x)$
  - Drop $P(x)$ since it is the same for both
  - $P(c|Good) = P(x|Good)P(Good) = .3 \cdot .8 = .24$
  - $P(c|Bad) = P(x|Bad)P(Bad) = .4 \cdot .2 = .08$

Bayesian Intuition

- Bayesian vs. Frequentist
- Bayesian allows us to talk about probabilities/beliefs even when there is little data, because we can use the prior
  - What is the probability of a nuclear plant meltdown?
  - What is the probability that BYU will win the national championship?
- As the amount of data increases, Bayes shifts confidence from the prior to the likelihood
- Requires reasonable priors in order to be helpful
- We use priors all the time in our decision making
  - Unknown coin: probability of heads? (over time?)

Bayesian Learning of ML Models

- Assume H is the hypothesis space, h a specific hypothesis from H, and D is the training data
- $P(h|D)$ - Posterior probability of h, this is what we usually want to know in a learning algorithm
- $P(h)$ - Prior probability of the hypothesis independent of D - do we usually know?
  - Could assign equal probabilities
  - *Could assign probability based on inductive bias* (e.g. simple hypotheses have higher probability)
- $P(D)$ - Prior probability of the data
- $P(D|h)$ - Probability "likelihood" of data given the hypothesis
  - This is usually just measured by the accuracy of model h on the data
- $P(h|D) = P(D|h)P(h)/P(D)$ Bayes Rule
- $P(h|D)$ increases with $P(D|h)$ *and* $P(h)$. In learning when seeking to discover the best h given a particular D, $P(D)$ is the same and can be dropped.

Bayesian Learning

- Learning (finding) the best model the Bayesian way
- Maximum a posteriori (MAP) hypothesis
- $h_{MAP} = \text{argmax}_{h \in H} P(h|D) = \text{argmax}_{h \in H} P(D|h)P(h)/P(D) = \text{argmax}_{h \in H} P(D|h)P(h)$
- Maximum Likelihood (ML) Hypothesis $h_{ML} = \text{argmax}_{h \in H} P(D|h)$
- MAP = ML if all priors $P(h)$ are equally likely
- Note that the prior can be like an inductive bias (i.e. simpler hypotheses are more probable)
- For Machine Learning $P(D|h)$ is usually measured using the accuracy of the hypothesis on the training data
  - If the hypothesis is very accurate on the data, that implies that the data is more likely given that particular hypothesis
  - For Bayesian learning, don't have to worry as much about h overfitting in $P(D|h)$ (early stopping, etc.) – Why?

Bayesian Learning (cont)

- Brute force approach is to test each $h \in H$ to see which maximizes $P(h|D)$
- Note that the argmax is not the real probability since $P(D)$ is unknown, but not needed if we're just trying to find the best hypothesis
- Can still get the real probability (if desired) by normalization if there is a limited number of hypotheses
  - Assume only two possible hypotheses h_1 and h_2
  - The true posterior probability of h_1 would be

$$P(h_1|D) = \frac{P(D|h_1)P(h_1)}{P(D|h_1) + P(D|h_2)}$$

Example of MAP Hypothesis

- Assume only 3 possible hypotheses in hypothesis space H
- Given a data set D which h do we choose?
- Maximum Likelihood (ML): $\text{argmax}_{h \in H} P(D|h)$
- Maximum a posteriori (MAP): $\text{argmax}_{h \in H} P(D|h)P(h)$

H	Likelihood $P(D\|h)$	Priori $P(h)$	Relative Posterior $P(D\|h)P(h)$
h_1	.6	.3	.18
h_2	.9	.2	.18
h_3	.7	.5	.35

Example of MAP Hypothesis – True Posteriors

- Assume only 3 possible hypotheses in hypothesis space H
- Given a data set D

H	Likelihood $P(D\|h)$	Priori $P(h)$	Relative Posterior $P(D\|h)P(h)$	True Posterior $P(D\|h)P(h)/P(D)$
h_1	.6	.3	.18	.18/(.18+.18+.35) = .18/.71 = .25
h_2	.9	.2	.18	.18/.71 = .25
h_3	.7	.5	.35	.35/.71 = .50

Prior Handles Overfit

- Prior can make it so that less likely hypotheses (those likely to overfit) are less likely to be chosen
- Similar to the regularizer
- Minimize $F(h) = Error(h) + \lambda \cdot Complexity(h)$
- $P(h|D) = P(D|h)P(h)$

Minimum Description Length

- Information theory shows that the number of bits required to encode a message i is $-\log_2 p_i$
- Call the minimum number of bits to encode message i with respect to code C: $L_C(i)$

$$h_{\text{MAP}} = \text{argmax}_{h \in H} P(h)\ P(D|h) =$$
$$\text{argmin}_{h \in H} - \log_2 P(h) - \log_2 (D|h) =$$
$$\text{argmin}_{h \in H} L_{C1}(h) + L_{C2}(D|h)$$

- $L_{C1}(h)$ is a representation of hypothesis
- $L_{C2}(D|h)$ is a representation of the data. Since you already have h all you need is the data instances which differ from h, which are the lists of misclassifications
- The h which minimizes the MDL equation will have a balance of a small representation (simple hypothesis) and a small number of errors

Bayes Optimal Classifier

- *Best* question is what is the most probable classification *c* for a given instance, rather than what is the most probable hypothesis for a data set
- Let all possible hypotheses vote for the instance in question weighted by their posterior (an ensemble approach) - better than the single best MAP hypothesis

$$P(c_j|D,H) = \sum_{h_i \in H} P(c_j|h_i)P(h_i|D) = \sum_{h_i \in H} \frac{P(D|h_i)P(h_i)}{P(D)}$$

- Bayes Optimal Classification:

$$c_{BayesOptimal} = \underset{c_j \in C}{\text{argmax}} \sum_{h_i \in H} P(c_j|h_i)P(h_i|D) = \underset{c_j \in C}{\text{argmax}} \sum_{h_i \in H} P(c_j|h_i)P(D|h_i)P(h_i)$$

- Also known as the posterior predictive

Example of Bayes Optimal Classification

$$c_{BayesOptimal} = \underset{c_j \in C}{\text{argmax}} \sum_{h_i \in H} P(c_j|h_i)P(h_i|D) = \underset{c_j \in C}{\text{argmax}} \sum_{h_i \in H} P(c_j|h_i)P(D|h_i)P(h_i)$$

- Assume same 3 hypotheses with priors and posteriors as shown for a data set D with 2 possible output classes (A and B)
- Assume novel input instance x where h_1 and h_2 output B and h_3 outputs A for x – 1/0 output case

H	Likelihood $P(D\|h)$	Prior $P(h)$	Posterior $P(D\|h)P(h)$	P(A)	P(B)
h_1	.6	.3	.18	0·.18 = .054	1·.18 = .126
h_2	.9	.2	.18	0·.18 = .072	1·.18 = .108
h_3	.7	.5	.35	1·.35 = .315	0·.35 = .035
Sum				.35	.36

Example of Bayes Optimal Classification

- Assume probabilistic outputs from the hypotheses

H	$P(\text{A})$	$P(\text{B})$
h_1	.3	.7
h_2	.4	.6
h_3	.9	.1

H	Likelihood $P(D\|h)$	Prior $P(h)$	Posterior $P(D\|h)P(h)$	$P(\text{A})$	$P(\text{B})$
h_1	.6	.3	.18	$.3 \cdot .18 = .054$	$.7 \cdot .18 = .126$
h_2	.9	.2	.18	$.4 \cdot .18 = .072$	$.6 \cdot .18 = .108$
h_3	.7	.5	.35	$.9 \cdot .35 = .315$	$.1 \cdot .35 = .035$
Sum				.441	.269

Bayes Optimal Classifiers (Cont)

- *No other classification method using the same hypothesis space can outperform a Bayes optimal classifier on average, given the available data and prior probabilities over the hypotheses*
- Large or infinite hypothesis spaces make this impractical in general
- Also, it is only as accurate as our knowledge of the priors (background knowledge) for the hypotheses, which we often do not know
  - But if we do have some insights, priors can really help
  - For example, it would automatically handle overfit, with no need for a validation set, early stopping, etc.
  - Note that using accuracy, etc. for likelihood $P(D|h)$ is also an approximation
- If our priors are bad, then Bayes optimal will not be optimal for the actual problem. For example, if we just assumed uniform priors, then you might have a situation where the many lower posterior hypotheses could dominate the fewer high posterior ones.
- However, this is an important theoretical concept, and it leads to many practical algorithms which are simplifications based on the concepts of full Bayes optimality (e.g. ensembles)

Revisit Bayesian Classification

- $P(c|x) = P(x|c)P(c)/P(x)$
- $P(c)$ - Prior probability of class c – How do we know?
  - Just count up and get the probability for the Training Set – Easy!
- $P(x|c)$ - Probability "likelihood" of data vector x given that the output class is c
  - How do we really do this?
  - If x is real valued?
  - If x is nominal we can just look at the training set and again count to see the probability of x given the output class c but how often will they be the same?
    - Which will also be the problem even if we bin real valued inputs

Naïve Bayes Classifier

$$c_{MAP} = \underset{c_j \in C}{\text{argmax}}\, P = (c_j | x_1, \ldots, x_n) \underset{c_j \in C}{\text{argmax}} \frac{P(x_1, \ldots, x_n | c_j) P(c_j)}{P(x_1, \ldots, x_n)} = \underset{c_j \in C}{\text{argmax}}\, P(c_j | x_1, \ldots, x_n) P(c_j)$$

- Note we are not considering $h \in H$, rather just collecting statistics from the data set
- Given a training set, $P(c_j)$ is easy to calculate
- How about $P(x_1, \ldots, x_n | c_j)$? Most cases would be either 0 or 1. Would require a huge training set to get reasonable values.
- Key "Naïve" leap: Assume conditional independence of the attributes

$$P(x_1, \ldots, x_n | c_j) = \prod_i P(x_i | c_j)$$

$$c_{NB} = \underset{c_j \in C}{\text{argmax}} P(c_j) \prod_i P(x_i | c_j)$$

- While conditional independence is not typically a reasonable assumption…
  - Low complexity simple approach, assumes nominal features for the moment - need only store all $P(v_j)$ and $P(a_i | v_j)$ terms, easy to calculate and with only $|attributes| \times |attribute\ values| \times |classes|$ terms there is often enough data to make the terms accurate at a 1st order level
  - Effective for many large applications (Document classification, etc.)

Naïve Bayes Homework

Size (B, S)	Color (R,G,B)	Output (P,N)
B	R	P
S	B	P
S	B	N
B	R	N
B	B	P
B	G	N
S	B	P

For the given training set:

1. Create a table of the statistics needed to do Naïve Bayes
2. What would be the output for a new instance which is Small and Blue?
3. What is the Naïve Bayes value and the normalized probability for each output class (P or N) for this case of Small and Blue?

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_i P(x_i|c_j)$$

What do we need?

Size (B, S)	Color (R,G,B)	Output (P,N)
B	R	P
S	B	P
S	B	N
B	R	N
B	B	P
B	G	N
S	B	P

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_i P(x_i|c_j)$$

P(P)	
P(N)	
P(Size=B|P)	
P(Size=S|P)	
P(Size=B|N)	
P(Size=S|N)	
P(Color=R|P)	
P(Color=G|P)	
P(Color=B|P)	
P(Color=R|N)	
P(Color=G|N)	
P(Color=B|N)	

Naïve Bayes (cont.)

- Again, can normalize to get the actual naïve Bayes probability
- Continuous data? - Can discretize a continuous feature into bins, thus changing it into a nominal feature and then gather statistics normally
  - How many bins? - More bins is good, but need sufficient data to make statistically significant bins. Thus, base it on data available
  - Could also assume data is Gaussian and compute the mean and variance for each feature given the output class, then each $P(a_i|v_j)$ becomes $\mathcal{N}(x_i|\mu_{vj}, \sigma^2{}_{cj})$
  - Not good if data is multi-modal

Infrequent Data Combinations

- Would if there are 0 or very few cases of a particular $x_i=v|c_j$ (n_v/n)? (n_v is the number of instances with output c_j where x_i = attribute value v. n is the total number of instances with output c_j)
- Should usually allow every case at least some finite probability since it could occur in the test set, else the 0 terms will dominate the product (speech example)
- Could replace n_c/n with the Laplacian: $(n_v+1)/(n+1/p)$
- p is a prior probability of the attribute value which is usually set to 1/(# of attribute values) for that attribute (thus $1/p$ is just the number of possible attribute values).
- Thus if n_v/n is 0/10 and x_i has three attribute values, the Laplacian would be 1/13.

Naïve Bayes (cont.)

- No training per se, just gather the statistics from your data set and then apply the Naïve Bayes classification equation to any new instance
- Easier to have many attributes since not building a net, etc. and the amount of statistics gathered grows linearly with the number of attributes (# *attributes* × # attribute values × # *classes*) - Thus natural for applications like text classification which can easily be represented with huge numbers of input attributes.
- Though Naïve Bayes is limited by the first order assumptions, it is still often used in many large real world applications

Text Classification Example

- A text classification approach
  - Want P(*class*|*document*) - Use a "Bag of Words" approach – order independence assumption (valid?)
    - Variable length input of query document is fine
  - Calculate *P*(*word*|*class*) for every word/token in the language and each output class based on the training data. Words that occur in testing but do not occur in the training data are ignored.
  - Good empirical results. Can drop filler words (the, and, etc.) and words found less than *z* times in the training set.

Text Classification Example

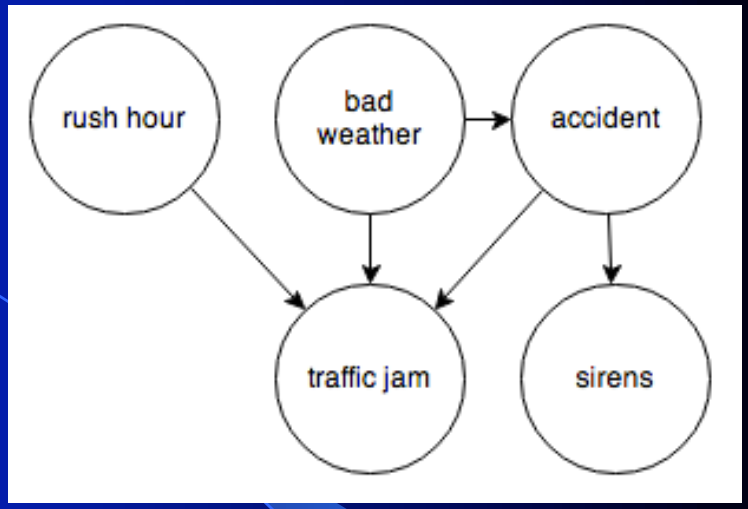
- A text classification approach
  - Want P(*class*|*document*) - Use a "Bag of Words" approach – order independence assumption (valid?)
    - Variable length input of query document is fine
  - Calculate *P*(*word*|*class*) for every word/token in the language and each output class based on the training data. Words that occur in testing but do not occur in the training data are ignored.
  - Good empirical results. Can drop filler words (the, and, etc.) and words found less than *z* times in the training set.
  - $P(class|document) \approx P(class|BagOfWords)$ //assume word order independence
    $= P(BagOfWords|class)*P(class)/P(document)$ //Bayes Rule
    // But *BagOfWords* usually unique
    //and *P*(*document*) same for all classes
    $\approx P(class)*\Pi P(word|class)$ // Thus Naïve Bayes

Less Naïve Bayes

- NB uses just 1st order features - assumes conditional independence
  - calculate statistics for all $P(x_i|c_j)$)
  - $|attributes| \times |attribute\ values| \times |output\ classes|$
- nth order - $P(x_i, \ldots, x_n|c_j)$ - assumes full conditional dependence
  - $|attributes|^n \times |attribute\ values| \times |output\ classes|$
  - Too computationally expensive - exponential
  - Not enough data to get reasonable statistics - most cases occur 0 or 1 time
- 2nd order? - compromise - $P(x_i x_k|c_j)$ - assume only low order dependencies
  - $|attributes|^2 \times |attribute\ values| \times |output\ classes|$
  - More likely to have cases where number of $x_i x_k|c_j$ occurrences are 0 or few, could just use the higher order features which occur often in the data
  - 3rd order, etc.
- How might you test if a problem is conditionally independent?
  - Could compare with nth order but that is difficult because of time complexity and insufficient data
  - Could just compare against 2nd order. How far off on average is our assumption

$$P(x_i x_k|c_j) = P(x_i|c_j)\ P(x_k|c_j)$$

Bayesian Belief Nets



- Can explicitly specify where there is significant conditional dependence - intermediate ground (all dependencies would be too complex and not all are truly dependent). If you can get both of these correct (or close) then it can be a powerful representation. - growing research area
- Specify causality in a DAG and give conditional probabilities from immediate parents (causal)
  - Still can work even if causal links are not that accurate, but more difficult to get accurate conditional probabilities
- Belief networks represent the full joint probability function for a set of random variables in a compact space - Product of recursively derived conditional probabilities
- If given a subset of observable variables, then you can infer probabilities on the unobserved variables - general approach is NP-complete - approximation methods are used
- Gradient descent learning approaches for conditionals. Greedy approaches to find network structure.