Probabilistic models

- Decision trees, instance-based learning, and transformation-based learning are called *non-parametric* methods because they don't use an explicit probabilistic model
- *Parametric* machine learning methods assume a particular (typically probabilistic) model
- Parametric methods (usually) search a much more restrictive hypothesis space than non-parametric methods \rightarrow large bias, small variance

- Suppose we have a representation of an instance as feature vector *x* and we want to predict its class *c*
- If we have a way of modeling *P*(*c*|*x*), *Bayes Decision Rule* says our predicted \hat{c} should be:

$$\hat{c} = \operatorname*{argmax}_{c \in C} P(c|x)$$

• This minimizes the expected error:

$$P(\text{error}|x) = 1 - P(\hat{c}|x)$$
$$P(\text{error}) = \sum_{x} P(\text{error}|x) P(x)$$

2

Probabilistic models

- There are two ways of applying the Bayes decision rule
- A discriminative (aka diagnostic) method directly models P(c|x)
- More commonly, a *generative* (aka *sampling*) method is used, which models the joint distribution *P*(*x*,*c*) and uses Bayes rule:

$$\hat{c} = \operatorname{argmax}_{c \in C} P(c|x)$$

$$= \operatorname{argmax}_{c \in C} \frac{P(x|c) P(c)}{P(x)}$$

$$= \operatorname{argmax}_{c \in C} P(x|c) P(c)$$

$$= \operatorname{argmax}_{c \in C} P(x,c)$$

Probabilistic models

- If Bayes decision rule minimizes error, why do we still make mistakes?
- Overlapping classification functions (where *P*(*c*|*x*) ≠ 1) can never be learned perfectly
- The classifier only works as well as our model if our model *P*(*c*|*x*) is inaccurate, then we'll make the wrong decisions
- We need some way of constructing, evaluation, and selecting probability models

Baseline classifier

• We often compute a 'baseline' for a classification task by simply assigning the most frequent class to each instance:

$$\hat{c} = \operatorname*{argmax}_{c \in C} P(c)$$

- Here we assume that P(c|x) = P(c), i.e., X and C are independent
- The extra error a baseline classifi er makes is:

$$\sum_{x} P(x) \left[P(x,c) - P(x) P(c) \right]$$

- Call a particular model *h*, chosen from the hypothesis space *H*.
- The maximum likelihood hypothesis selects:

 $\hat{c} = \underset{c \in C, h \in H}{\operatorname{argmax}} P(c|x, h) P(d|h)$

• The maximum a posteriori hypothesis selects:

 $\hat{c} = \operatorname*{argmax}_{c \in C, h \in H} P(c|x, h) P(d|h) P(h)$

• Both of these commit us to choosing one *h*, which may or may not wind up being the best choice

Bayes Optimal Classifiers

• The Bayes Optimal Classifier selects:

$$\hat{c} = \operatorname*{argmax}_{c \in C} \sum_{h \in H} P(c|x, h) P(d|h) P(h)$$

- We remove the dependence on a particular *h* by averaging over all possible *h*s
- This is almost always impossible to apply in practice, but it can used to establish a lower bound on the error rate
- We can also sometimes approximate it, e.g., by randomly drawing h from the posterior distribution P(d|h)P(h)

Rev. Thomas Bayes (1702–1761)



5

Rev. Thomas Bayes (1702–1761)



- To apply a generative Bayesian classifier, we need P(x,c)
- We can break this down into two parts: the *class prior* P(c), and a likelihood P(x|c)
- The class priors are easy to estimate from training data:

 $\hat{P}(c) = \frac{\text{\# of instances in class } c}{\text{\# of instances}}$

• This won't work for *P*(*x*|*c*), since any particular feature vector *x* is unlikely to turn up in the training data:

$$\hat{P}(x|c) = \frac{\text{\# of instances of } x \text{ in } c}{\text{\# of instances in } c} \approx \frac{0}{\text{\# of instances of } x \text{ in } c}$$

10

Naive Bayes classifiers

• The naive Bayes classifier selects the class \hat{c} such that:

$$\hat{c} = \operatorname*{argmax}_{c \in C} P(c) \prod_{i} P(x_i | c)$$

- Naive Bayes classifiers have been used primarily for classifying texts (Maron 1961)
- We treat a text as a set or bag of words, an unordered collection of all the words that appear in the text
- "We treat a text as a set or bag of words" \equiv { a, a, as, bag, of, or, set, text, treat, we, words }

Naive Bayes classifiers

• To get a better estimate of *P*(*x*|*c*), we can make the simplifying assumption that each of the dimensions *x_i* in *x* are independent, so that:

$$P(x|c) = \prod_{i} P(x_i|c)$$

• Now we only need to get estimates of $P(x_i|c)$ from the data for each x_i , which we can do in the usual way:

$$\hat{P}(x_i|c) = \frac{\text{\# of instances of } x_i \text{ in } c}{\text{\# of instances in } c}$$

• Both $\hat{P}(c)$ and $\hat{P}(x_i|c)$ can be estimated using whatever tricks we have available

Naive Bayes classifiers

Text classification

 Ignoring word order in the feature representation removes the most obvious syntactic dependencies between words

 $P(\text{the}) P(\text{book}) \neq P(\text{the book})$

• There are still semantic dependencies:

 $P(\text{tackle}) P(\text{touchdown}) \neq P(\text{tackle}, \text{touchdown})$

• And, multiple occurrences of words are probably not independent

- Text classification can be useful for information retrieval and natural language processing tasks
 - ⋆ indexing
 - ★ message routing
 - summarization
- Text classification also plays a role in linguistic research
 - authorship identification
 - ⋆ genre studies
 - * forensic linguistics
 - sociolinguistics
- A combination of the two makes the WWW available as a resource for research

13

Feature selection

- A straight bag-of-words model leads to positing a very large number of features
- Some of those features will not be relevant for the task (stop words)
- Many of the features will appear relevant, but won't be: we can't avoid the Curse of Dimensionality
- So, we want to select a subset of features which appear promising, usually by information gain

Multivariate Bernoulli event model

• If we represent a document as a *set* of words, then each feature *x_i* is a Bernoulli variable, where:

$$P(x_i|c_j) = P(x_i = 1|c_j)^{x_i} (1 - P(x_i = 1|c_j))^{1-x_i}$$

- If there are *v* words in the vocabulary, a document is constructed by flipping *v* coins
- Call $p_{ij} = P(x_i = 1|c_j)$. Substituting this in, we get:

$$P(c_j|x) = \frac{P(c_j)\prod_i P(x_i|c_j)}{P(x)} \\ = \frac{P(c_j)\prod_i p_{ij}^{x_i}(1-p_{ij})^{1-x}}{P(x)}$$

• And taking the log gives us:

$$\begin{split} \log P(c_j|x) &= \log P(c_j) + \sum_{i} x_i \log p_{ij} + \sum_{i} (1 - x_i) \log (1 - p_{ij}) - \log P(x) \\ &= \log P(c_j) + \sum_{i} x_i \log p_{ij} + \sum_{i} \log (1 - p_{ij}) - \sum_{i} x_i \log (1 - p_{ij}) - \log P(x) \\ &= \log P(c_j) + \sum_{i} x_i \log \frac{p_{ij}}{1 - p_{ij}} + \sum_{i} \log (1 - p_{ij}) - \log P(x) \end{split}$$

• Suppose we only have two classes. Then $P(c_1|x) = 1 - P(c_2|x)$, and the posterior log odds are:

$$\log \frac{P(c_1|x)}{1 - P(c_1|x)} = \sum_i x_i \log \frac{p_{i1}(1 - p_{i2})}{(1 - p_{i1})p_{i2}} + \sum_i \log \frac{1 - p_{i1}}{1 - p_{i2}} + \log \frac{P(c_1)}{1 - P(c_1)}$$

17

Multinomial event model

- If instead we represent a document as a *bag* of words, then we can model a document as a sequence of random draws from a multinomial distribution
- The probability of picking word w_i if the document class is c_j once is $P(w_i|c_j)$
- The probability of picking word $w_i x_i$ times in a row is $P(w_i|c_j)^{x_i}$
- The probability of drawing a collection of words in that order is:

$$\prod_i P(w_i|c_i)^{x_i}$$

• Under this *binary independence model,* the parameters *p*_{*ij*} can be estimated via:

$$\hat{p}_{ij} = \frac{\text{\# of documents containing } x_i \text{ in } c_j}{\text{\# of documents in class } c_i}$$

- Note that this doesn't take into account the length of the document
- It also doesn't take into account the number of times a word appears in a document

18

Multinomial event model

- This underestimates *P*(*x*|*c*_{*j*}), since lots of ordered sequences correspond to the same bag of words
- How many different ways are there to draw word $w_1 x_1$ times, word $w_2 x_2$, and so on?
- We can use the *multinomial coefficient*:

$$\begin{pmatrix} n \\ n_1, n_2, \dots \end{pmatrix} = \binom{n}{n_1} \times \binom{n-n_1}{n_2} \times \cdots$$

$$= \frac{n!}{n_1!(n-n_1)!} \times \frac{(n-n_1)!}{n_2!(n-n_1-n_2)!} \times \cdots$$

$$= \frac{n!}{n_1!n_2!\cdots}$$

• So, if we draw $N = \sum_i x_i$ words, we have:

$$P(x|c_j) = \binom{N}{x_1, x_2, \dots} \prod P(w_i|c_j)^{x_i}$$
$$= N! \prod \frac{P(w_i|c_j)^{x_i}}{x_i!}$$

• To be completely correct, we also need to think about the probability of finding a document of a particular length:

$$P(x|c_j) = P(N|c_j)(\sum_i x_i)! \prod \frac{P(w_i|c_j)^{x_i}}{x_i!}$$

but in practice this can be hard to do.

- Multinomial event model
- The parameters of the multinomial model are the individual word probabilities $P(w_i|c_j)$
- Since these are the parameters of a multinomial distribution, we need to maintain:

$$\sum_{i} P(w_i | c_j) = 1$$

• We can estimate those from training data as:

$$\hat{P}(w_i|c_j) = \frac{\text{\# of times } w_i \text{ occurs in documents in } c_j}{\text{\# of words in documents in class } c_j}$$

• As always, smoothing is important

21

Text classification

- The multinomial model takes word frequencies and document length into account, but treats multiple occurrences of a word as independent events
- McCallum and Nigam (1998) compare the two event models
- Multinominal model almost always outperforms multivariate Bernoulli model, by 25% or so
- The multinominal model handles large vocabulary sizes much better
- It's easier to see how to add non-text features and to account for limited inter-dependencies using a multivariate Bernoulli model

Naive Bayes classifiers

- Despite its obvious limitations, naive Bayes text classifiers work quite well
- Lewis and Ringuette 1994 'breakeven point' for naive Bayes very close to decision trees
- In other work, naive Bayes scores close to, but consistently worse than, more sophisticated methods
- Since naive Bayes is pretty good, and it's easy to implement, it is very widely used

Naive Bayes classifiers

Naive Bayes classifiers

• Paul Graham wrote an article on naive Bayes classifiers for filter ing junk mail, which has become a standard method

Free CableTV!No more pay!%RND_SYB

requisite silt administer orphanage teach hypothalamus diatomic conflict atlas moser cofactor electret coffin diversionary solicitous becalm absent satiable blurb mackerel sibilant tehran delivery germicidal barometer falmouth capricorn • Maron (1961):

It is feasible to have a computing machine read a document and to decide automatically the subject category to which the item in question belongs. No real intellectual breakthroughs are required before a machine will be able to index rather well. Just as in the case of machine translation of natural language, the road is gradual but, by and large, straightfoward.

25

Zero-one loss

- Given its obvious deficiencies, why does naive Bayes work as well as it does?
- Its probability estimates are only as good as the independence assumptions are valid (i.e., not very)
- But, we don't evaluate a naive Bayes classifier on its probability estimates
- Instead, we measure its misclassification error, or zero-one loss
- The two measures need not be closely related

Zero-one loss



Zero-one loss

Zero-one loss

- If the features are independent, then naive Bayes is optimal under zero-one loss
- Domingos and Pazzani (1997) evaluate naive Bayes on problems from the UCI repository, and find it often performs very well, but sometimes it performs badly
- They then used mutual information to measure the pairwise dependencies between features
- There was no clear relationship between the validity of independence assumptions and the performance of naive Bayes

- Suppose there are two classes, and let $p = P(c_1|x)$, $r = P(c_1) \prod_i P(x_i|c_1)$ and $s = r = P(c_2) \prod_i P(x_i|c_2)$
- For any instance x, naive Bayes is optimal under zero-one loss if and only $(p \ge \frac{1}{2} \land r \ge s) \lor (p \le \frac{1}{2} \land r \le s)$
- That means that naive Bayes is optimal under zero-one loss for half the volume of the space of possible values of (*p*,*r*,*s*)!
- The naive Bayes probabilities are optimal only along the line where the planes *r* = *p* and *s* = 1 *p* intersect

29

Zero-one loss

- A necessary condition: naive Bayes can only be optimal (for discrete features) for concepts that are linearly separable
- For discrete features, combinations of variables by $\land,\lor,$ and \neg are linearly separable
- This isn't a sufficient condition, since there are linearly separable concepts which naive Bayes performs poorly on (*m*-of-*n* concepts)
- Naive Bayes is optimal for conjunctions of features and for disjunctions of features
- This points to one way to improve naive Bayes: introduce new features which are disjunctions (or conjunctions) of other features

Zero-one loss

- Even when naive Bayes is not optimal, it may outperform other methods with greater representational power (e.g., C4.5)
- Zero-one loss is relatively insensitive to *bias,* but can be highly sensitive to *variance*
- When there isn't enough training data, a high bias, low variance learner will give a lower zero-one loss than a low bias, high variance learner
- We've seen this before: a simple model can outperform a more complex one, even when the assumptions of the simple model are false