Data mining & Machine Learning

CS 373
Purdue University

Dan Goldwasser
dgoldwas@purdue.edu

Today's Lecture

More Supervised Learning!

- *Decision Trees Wrap Up*
- *Evaluation and model selection*
- ***Probabilistic Classification using the Naïve Bayes algorithm***
  - *A **Generative** model (huh?)*
  - *Can be used for classification, ranking and assigns output probabilities*
  - *Naturally deals with binary and multiclass classification*
  - *Really easy to understand and implement. **Too Easy.** (huh?)*
  - *Works annoyingly well!*

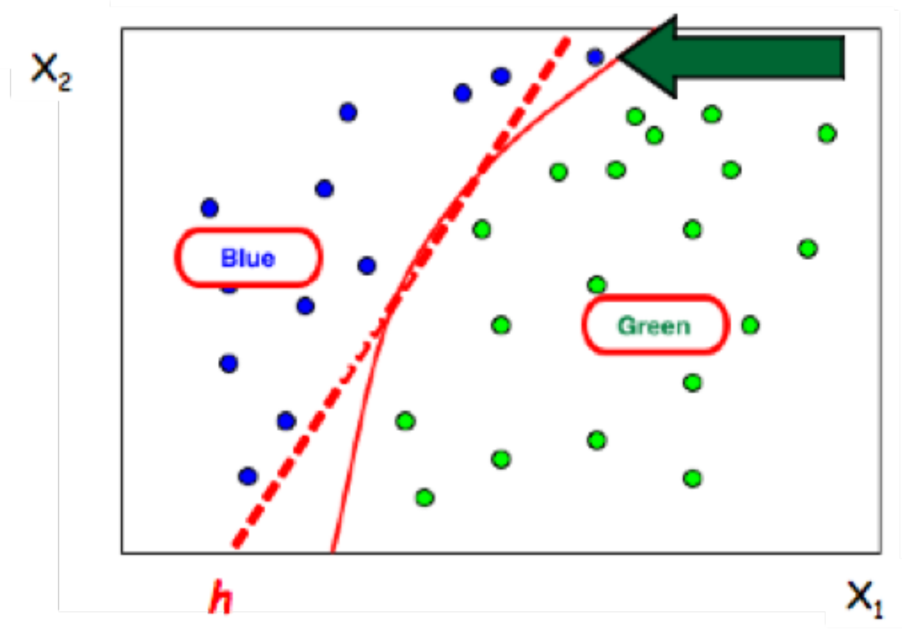
Predictive Modeling

- **Data representation:**
  - **Training set**: Paired attribute vectors and class labels `<y(i), x(i)>`
- **Task**: estimate a predictive function $f(x;\vartheta)=y$
  - Assume that there is a function *y=f(x)* that **maps** data instances (**x**) to class labels (y)
- Construct a model that approximates the mapping
  - **Classification**: if y is categorical
  - **Regression**: if y is real-valued

Classification

- In its simplest form, ***a classification model defines a decision boundary (h) and labels for each side of the boundary***

- Input: $\mathbf{x}=\{x_1,x_2,\ldots,x_n\}$ is a set of attributes, function f assigns a label y to input **x**, where y is a discrete variable with a finite number of values



Classification output

- **Different classification tasks can require different kinds of output**
  - *Each requires progressively more accurate models (e.g., a poor probability estimator can still produce an accurate ranking)*
- **Class labels** — Each instance is assigned a single label
  - *Model only need to decide on crisp class boundaries*
- **Ranking** — Instances are ranked according to their likelihood of belonging to a particular class
  - *Model implicitly explores many potential class boundaries*
- **Probabilities** — Instances are assigned class probabilities $p(y|\mathbf{x})$
  - *Allows for more refined reasoning about sets of instances*

Probabilistic classification

- Model the underlying probability distributions
  - Posterior class probabilities: $p(y|\mathbf{x})$
  - Class-conditional and class prior: $p(\mathbf{x}|y)$ and $p(y)$
- Maps from inputs **x** to class label y indirectly through posterior class distribution $p(y|\mathbf{x})$
- Examples:
  - Naive Bayes classifier, logistic regression, probability estimation trees

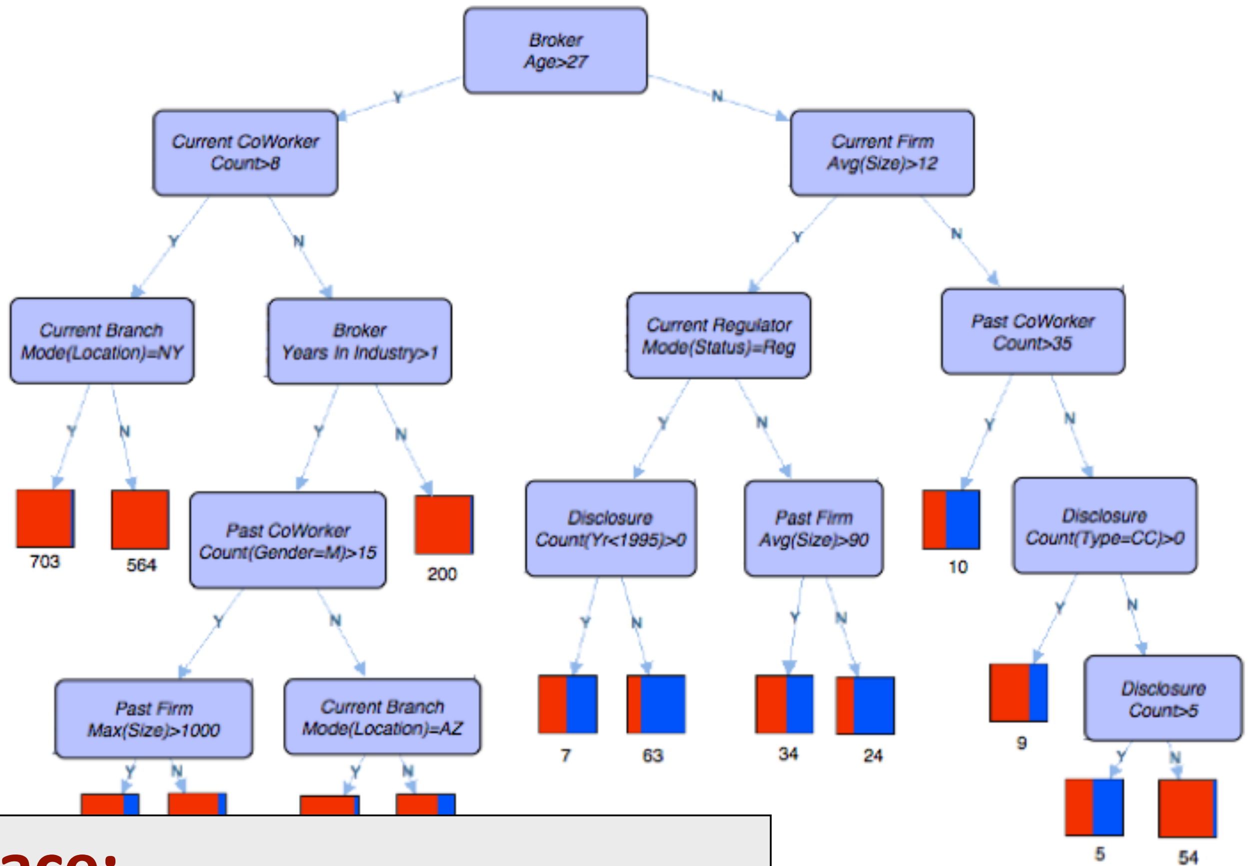
Analyzing Supervised Learning Algorithms

- Similar to our previous discussions, supervised learning algorithms can be analyzed according to:
  - **Model/hypothesis space**(knowledge representation)
  - **Scoring function**
  - **Search procedure**

Parametric vs. non-parametric models

- **Parametric**
  - Particular functional form is assumed (e.g., Binomial)
  - **Number of parameters is fixed in advance**
  - Examples: Naive Bayes, perceptron
- **Non-parametric**
  - Few assumptions are made about the functional form
  - **Model structure is determined from data**
  - Examples: classification tree, nearest neighbor

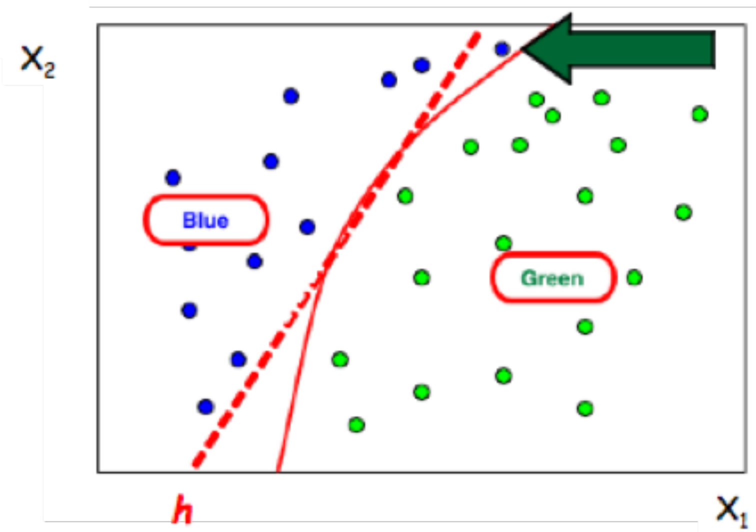
Classification tree



Model space:
all possible decision trees

Perceptron

$$f(x) = \begin{cases} 1 & \sum w_j x_j > 0 \\ 0 & \sum w_j x_j \leq 0 \end{cases}$$



Model space:
weights *w*, for each of j attributes

Example model:

Naïve Bayes classifiers

Classification as probability estimation

- Instead of learning a function f that assigns labels
- Learn a conditional probability distribution over the output of function f
- $P(f(x)|x) = P(f(x) = y \ |x_1, x_2, \ldots, x_p)$
- Can use probabilities for the other two tasks
  - Classification
  - Ranking

Knowledge representation and model space

Bayes rule for probabilistic classifier

$$P(Y|\mathbf{X}) = \frac{P(\mathbf{X}|Y)P(Y)}{P(\mathbf{X})}$$

Bayes rule

$$= \frac{P(\mathbf{X}|Y)P(Y)}{[P(\mathbf{X}|Y=+)P(Y=+)] + [P(\mathbf{X}|Y=-)P(Y=-)]}$$

$$\propto P(\mathbf{X}|Y)P(Y)$$

Denominator: normalizing factor to make probabilities sum to 1 (can be computed from numerators)

Bayes rule for probabilistic classifier

- P(y) - the **prior probability** of a label y
  Reflects *background knowledge*; before data is observed. If no information - uniform distribution.
- P(x) - The probability that this sample of the Data is observed. (*No knowledge of the label*)
- P(x|y): The probability of observing the sample x, given that the label y is the target (*Likelihood*)
- P(y|x): The **posterior probability** of v. The probability that v is the target, given that D has been observed.

$$P(Y|\mathbf{X}) = \frac{P(\mathbf{X}|Y)P(Y)}{P(\mathbf{X})}$$

Bayes rule for probabilistic classifier

$$P(Y|\mathbf{X}) = \frac{P(\mathbf{X}|Y)P(Y)}{P(\mathbf{X})}$$

Check your intuition:

P(y|x) increases with P(y) and with P(x|y)

P(y|x) decreases with P(x)

Bayes rule for probabilistic classifier

- The learner considers a set of candidate labels, and attempts to find the most probable one $y \in Y$, given the observed data.

- Such maximally probable assignment is called maximum a posteriori assignment (**MAP**); Bayes theorem is used to compute it:

$$y_{MAP} = \text{argmax}_{y \in Y} P(y|x) = \text{argmax}_{y \in Y} P(x|y)\,P(y)/P(x)$$

$$= \text{argmax}_{y \in Y} P(x|y)\,P(y)$$

Since P(x) is the same for all $y \in Y$

Bayes rule for probabilistic classifier

- How can we compute P(v |D)?
    - Basic idea: represent input as a set of features (e.g., BoW features)

$$y_{MAP} = \text{argmax}_{y \in Y} P(y|x) = \text{argmax}_{y \in Y} P(y|x_1, x_2, \ldots, x_n)$$

$$y_{MAP} = \text{argmax}_{y \in Y} P(x_1, x_2, \ldots, x_n|y) P(y) / P(x_1, x_2, \ldots, x_n) =$$

$$= \text{argmax}_{y \in Y} P(x_1, x_2, \ldots, x_n|y) P(y)$$

Bayes rule for probabilistic classifier

$$y_{MAP} = \text{argmax}_{y \in Y} P(x_1, x_2, \ldots, x_n|y)\, P(y)$$

- Given training data we can estimate the two terms
  - Estimating $P(y)$ is **easy**. For each value v count how many times it appears in the training data.

Question: Assume binary x_i's. How many parameters does the model require?

- However, it is not feasible to estimate $P(x_1,\ldots, x_n \mid y)$
  - In this case we have to estimate, for each target value, the probability of each instance (most of which will not occur)
- In order to use a Bayesian classifiers in practice, we *need to make assumptions* that will allow us to estimate these quantities.

NB: Independence Assumptions

Conditional Independence:

Assume feature probabilities are independent given the label

$P(x_i|y_j) = P(x_i|x_{i-1}; y_j)$

$$\boxed{P(Y|\mathbf{X}) \propto P(\mathbf{X}|Y)P(Y)}$$

Bayes rule

$$\propto \boxed{\prod_{i=1}^{m} P(X_i|Y)} P(Y)$$

Naive assumption

Question: How many parameters do we need to estimate now?

Is assuming independence a problem?

$Y = XOR(X_1, X_2)$

X_1	X_2	$P(Y=0 \mid X_1, X_2)$	$P(Y=1 \mid X_1, X_2)$
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	0

Is assuming independence a problem?

- Let's consider the spam classification problem
- Is NB an appropriate model to use?
  - Does the conditional independence assumption hold for this problem?
- However NB is frequently (and successfully) used for spam detection!
- **Why does it succeed?**

Inductive bias – Naïve Bayes version

- An acute version of overfitting occurs when we try to estimate $P(Y|\mathbf{X}) = P(Y)\, P(\mathbf{X}|Y)$ directly
- It requires learning 2^n parameters, **essentially one parameter for each input instance.**
    - This is overfitting at its worst – just memorizing the data
- We encountered this problem before in decision trees – Trees that have n intermediate nodes only memorize the data.
    - How did we solve it for decision trees?
- Similarly – making independence assumptions is a way to control the complexity of the model space and prevent overfitting.

NBC learning

$$P(BC|A,I,S,CR) = \frac{P(A,I,S,CR|BC)P(BC)}{P(A,I,S,CR)}$$
$$= \frac{P(A|BC)P(I|BC)P(S|BC)P(CR|BC)P(BC)}{P(A,I,S,CR)}$$
$$\propto \underline{P(A|BC)P(I|BC)P(S|BC)P(CR|BC)P(BC)}$$

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

NBC parameters = CPDs+prior

```
CPDs: P(A|BC)
      P(I|BC)
      P(S|BC)
      P(CR|BC)
Prior:P(BC)
```

Score function

Likelihood

- Let $D = \{x(1), ..., x(n)\}$
- Assume the data *D* are independently sampled from the same distribution: $p(X|\theta)$
- The likelihood function represents the probability of the data as a function of the model parameters:

$$
\begin{aligned}
L(\theta|D) &= L(\theta|x(1), ..., x(n)) \\
&= p(x(1), ..., x(n)|\theta) \\
&= \prod_{i=1}^{n} p(x(i)|\theta)
\end{aligned}
$$

If instances are independent, likelihood is product of probs

Likelihood (cont')

- Likelihood is not a probability distribution
  - Gives relative probability of data given a parameter
  - Numerical value of L is not relevant, only the ratio of two scores is relevant, e.g.,:

$$\frac{L(\theta_1|D)}{L(\theta_2|D)}$$

- **Likelihood function**: allows us to determine unknown parameters based on known outcomes
- **Probability distribution**: allows us to predict unknown outcomes based on known parameters

NBCs: Likelihood

- NBC likelihood uses the NBC probabilities for each data instance (i.e., probability of the class given the attributes)

$$L(\theta|D) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i;\theta)$$

General likelihood

$$\propto \prod_{i=1}^{n} p(\mathbf{x}_i|y_i;\theta)p(y_i|\theta)$$

Bayes rule

$$\propto \prod_{i=1}^{n} \prod_{j=1}^{p} p(x_{ij}|y_i;\theta)p(y_i|\theta)$$

Naive assumption

Search

Maximum likelihood estimation

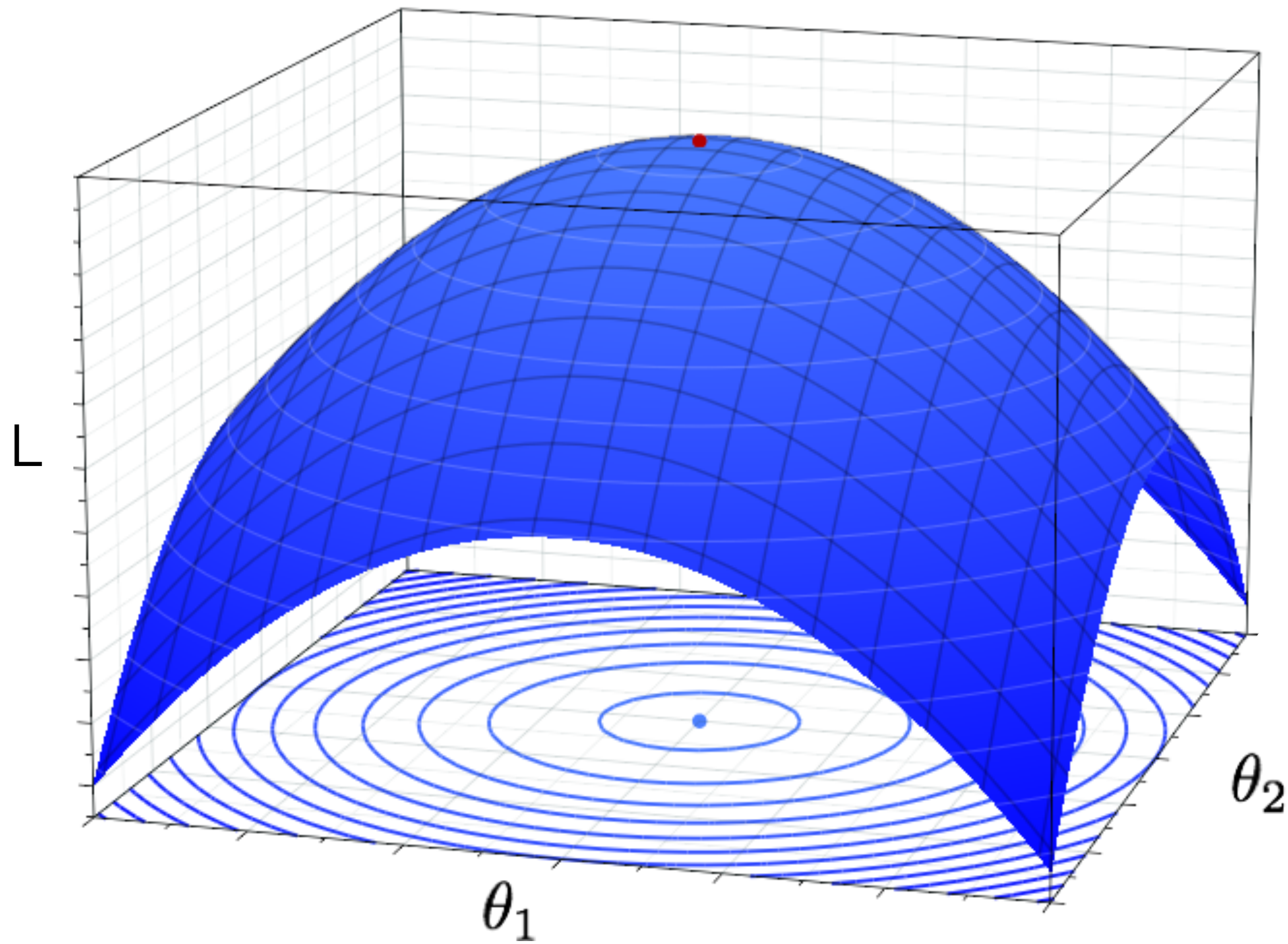
- Most widely used method of parameter estimation
- "Learn" the best parameters by finding the values of θ that maximizes likelihood:

$$\hat{\theta}_{MLE} = \arg\max_{\theta} L(\theta)$$

- Often easier to work with loglikelihood:

$$\begin{aligned} l(\theta|D) &= log\, L(\theta|D) \\ &= log \prod_{i=1}^{n} p(x(i)|\theta) \\ &= \sum_{i=1}^{n} logp(x(i)|\theta) \end{aligned}$$

Likelihood surface



If the likelihood surface is convex we can often determine the parameters that maximize the function analytically

MLE for multinomials

- Let $X \in \{1, \ldots, k\}$ be a discrete random variable with k values, where $P(X=j)=\theta_j$

- Then P(X) is a multinomial distribution:
$$P(X|\theta) = \prod_{j=1}^{k} \theta_j^{I(X=j)}$$
where I(X=j) is an indicator function

- The likelihood for a data set D=[x_1, ..., x_N] is:
$$P(D|\theta) = \prod_{n=1}^{N} \prod_{j=1}^{k} \theta_j^{I(x_n=j)} = \prod_{j} \theta_j^{N_j}$$

- The ML estimates for each parameter are: (using Lagrange multipliers)

$$\boxed{\hat{\theta}_j = \frac{N_j}{N}}$$

In this case, MLE can be determined analytically by counting

Learning CPDs from examples

		X_1		
		Low	Medium	High
Y	Yes	10	13	17
	No	2	13	0

$$P[\, X_1 = \text{Low} \mid Y = \text{Yes}] = \frac{10}{(10+13+17)}$$

$$P[\, Y = \text{No}] = \frac{(2+13)}{(2+13+10+13+17)}$$

NBC learning

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- Estimate prior P(BC) and conditional probability distributions P(A | BC), P(I | BC), P(S | BC), P(CR | BC) independently with maximum likelihood estimation

P(BC)

BC	θ
yes	$9/14$
no	$5/14$

P(A | BC)

BC	A	θ
yes	$<= 30$	$2/9$
	$31..40$	$4/9$
	> 40	$3/9$
no	$<= 30$	$3/5$
	$31..40$	$0/5$
	> 40	$2/5$

P(I | BC)

BC	I	θ
yes	high	$2/9$
	med	$4/9$
	low	$3/9$
no	high	$2/5$
	med	$2/5$
	low	$1/5$

P(S | BC)

BC	S	θ
yes	yes	$6/9$
	no	$3/9$
no	yes	$1/5$
	no	$4/5$

P(CR | BC)

BC	CR	θ
yes	exc	$3/9$
	fair	$6/9$
no	exc	$4/5$
	fair	$1/5$

NBC prediction

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no
31..40	**high**	**no**	**excellent**	**?**

- What is the probability that a new person will buy a computer?

$$P(BC = yes|A = 31..40, I = high, S = no, CR = exc)$$
$$\propto P(A = 31..40|BC = yes)P(I = high|BC = yes)$$
$$P(S = no|BC = yes)P(CR = exc|BC = yes)P(BC = yes)$$

P(BC)

BC	θ
yes	$9/14$
no	$5/14$

P(A | BC)

BC	A	θ
	$<= 30$	$2/9$
yes	$31..40$	$4/9$
	> 40	$3/9$
	$<= 30$	$3/5$
no	$31..40$	$0/5$
	> 40	$2/5$

P(I | BC)

BC	I	θ
	high	$2/9$
yes	med	$4/9$
	low	$3/9$
	high	$2/5$
no	med	$2/5$
	low	$1/5$

P(S | BC)

BC	S	θ
yes	yes	$6/9$
	no	$3/9$
no	yes	$1/5$
	no	$4/5$

P(CR | BC)

BC	CR	θ
yes	exc	$3/9$
	fair	$6/9$
no	exc	$4/5$
	fair	$1/5$

Zero counts are a problem

- If an attribute value does not occur in training example, we assign **zero** probability to that value
- How does that affect the conditional probability P[f(x) | x] ?
- It equals 0!!!
- Why is this a problem?
- Adjust for zero counts by "smoothing" probability estimates

Smoothing: Laplace correction

		X_1	
	Low	Medium	High
Y: Yes	10	13	17
Y: No	2	13	0

$$P[X_1 = \text{High} \mid Y = \text{No}] = \frac{0 \quad +1}{(2+13+0)+3}$$

Laplace correction
Numerator: *add 1*
Denominator: *add k*,
where k=number of possible values of X

Adds uniform prior

Naive Bayes classifier

- **Simplifying (naive) assumption:**
  attributes are conditionally independent given the class
- **Strengths**:
  - Easy to implement
  - Often performs well even when assumption is violated
  - Learning is really fast! *(why?)*
- **Weaknesses:**
  - Class conditional assumption produces skewed probability estimates
  - Dependencies among variables cannot be modeled

NBC learning

- **Model space**
  - Parametric model with specific form
    (i.e., based on Bayes rule and assumption of conditional independence),
  - Models vary based on parameter estimates in CPDs
- **Search algorithm**
  - MLE optimization of parameters
    (convex optimization results in exact solution)
- **Scoring function**
  - Likelihood of data given NBC model form

Example question: Compare NBC to DT to KNN

- **Hypothesis space**
  - What type of functions are used? Which one is more expressive?
- **Scoring function**
  - How is each model scored?
- **Search**
  - Which search procedure is used?
  - Are we guaranteed to find the optimal model?
  - What is the complexity of the search procedure?

Numerical Stability

- Recall: NB classifier: $$\propto \prod_{i=1}^{m} P(X_i|Y)P(Y)$$

  - **Multiplying probabilities can get us into problems!**
  - Imagine computing the probability of 2000 independent coin flips
  - Most programming environments: $(.5)^{2000}=0$

Numerical Stability

- Our problem: **Underflow Prevention**
- Recall: *log(xy) = log(x) + log(y)*
- better to sum logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)$$