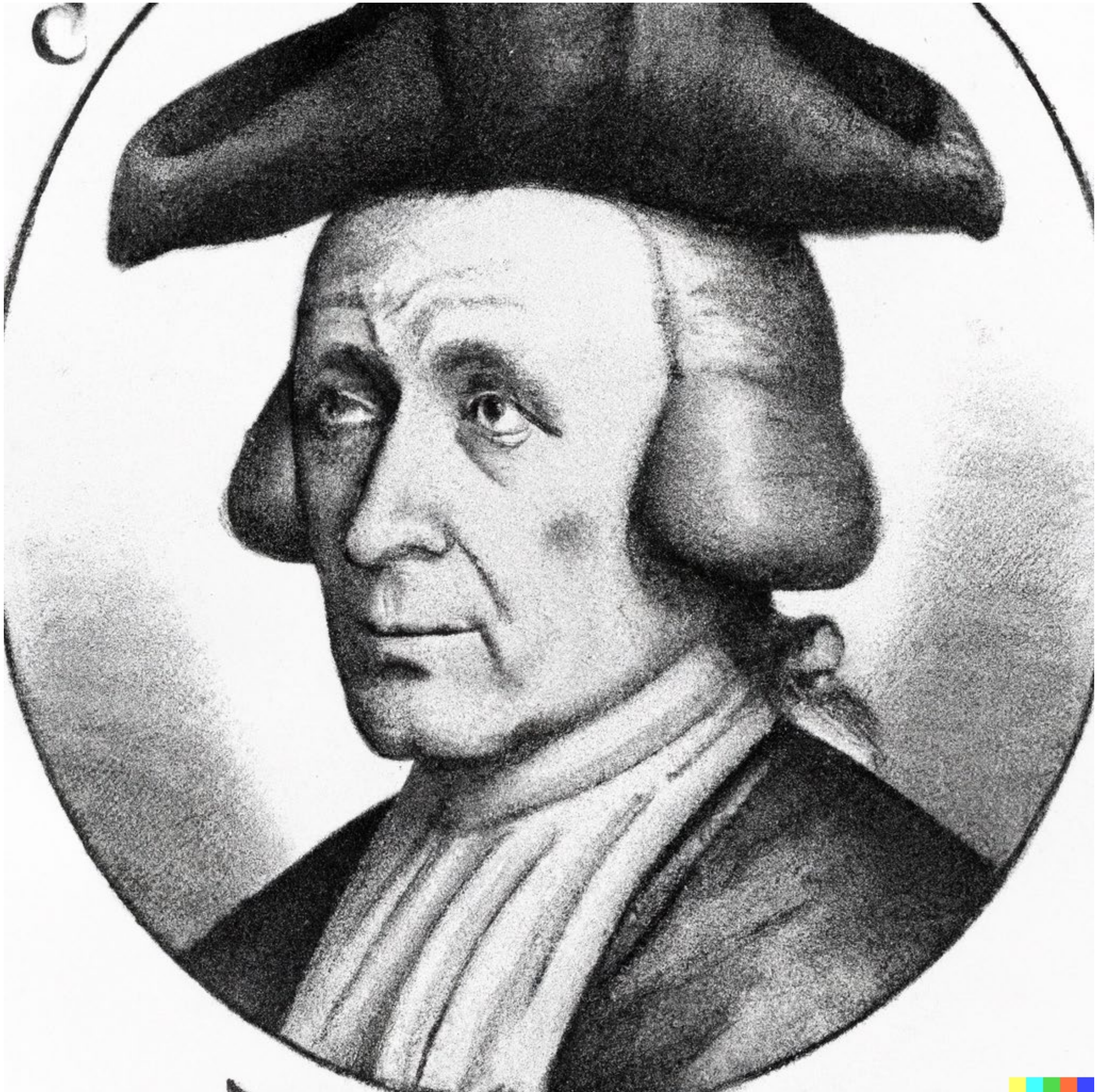


Probability and Naïve Bayes

Applied Machine Learning
Derek Hoiem

Dall-E: portrait of Thomas Bayes with a Dunce Cap on his head

Recap of approaches we’ve seen so far

- Nearest neighbor is widely used
  – Super-powers: can instantly learn new classes and predict from one or many examples

- Logistic Regression is widely used
  – Super-powers: Effective prediction from high-dimensional features

- Linear Regression is widely used
  – Super-powers: Can extrapolate, explain relationships, and predict continuous values from many variables

- Almost all algorithms involve nearest neighbor, logistic regression, or linear regression
  – The main learning challenge is typically **feature learning**

Today’s Lecture

- Introduce probabilistic models
- Review of probability
- Naïve Bayes Classifier
  - Assumptions / model
  - How to estimate from data
  - How to predict given new features
- “Semi-naïve Bayes” object detector

Probabilistic model

$$y^* = \underset{y}{\operatorname{argmax}}\, P(y|x)$$

Joint and conditional probability

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

$$P(a, b, c) = P(a|b, c)P(b|c)P(c)$$

Bayes Rule: $$P(x|y) = \frac{P(x, y)}{P(y)} = \frac{P(y|x)P(x)}{P(y)}$$

Law of total probability $\left[\sum_{v \in x} P(x = v)\right] = 1$

Marginalization $\left[\sum_{v \in x} P(x = v, y)\right] = P(y)$

For continuous variables, replace sum over possible values with integral over domain

Estimate probabilities of discrete variables by counting

$$P(x = v) = \frac{1}{|N|} \sum_{n} \delta(x_n = v)$$

Example

x: Larger than 10 lbs?

y		F	T
	Cat	15	25
	Dog	5	40

$$P(y = Cat) =$$

$$P(y = Cat|x = F) =$$

$$P(x = F|y = Cat) =$$

A is independent of B if (and only if)

$$P(A, B) = P(A)P(B)$$

$$P(A|B) = P(A), \qquad P(B|A) = P(B)$$

What if you have 100 variables? How can you count all combinations?

Fully modeling dependencies between many variables (more than 3 or 4) is challenging and requires a lot of data

Probabilistic model

$$y^* = \underset{y}{\operatorname{argmax}}\, P(y|x)$$

Or equivalently…

$$y^* = \underset{y}{\operatorname{argmax}}\, P(x|y)P(y)$$

$$\underset{y}{\operatorname{argmax}}\, P(y|x) = \underset{y}{\operatorname{argmax}}\, P(y|x)P(x) = \underset{y}{\operatorname{argmax}}\, P(y,x) = \underset{y}{\operatorname{argmax}} P(x|y)P(y)$$

Notation

- x_i is the ith feature variable
  - i indicates the feature index
- x_n is the nth feature vector
  - n indicates the sample index
  - y_n is the nth label
- x_{ni} is the ith feature of the nth sample
- $\delta(x_{ni} = v)$ returns 1 if $x_{ni} = v$; 0 otherwise
  - v indicates a feature value
  - δ is an indicator function, mapping from true/false to 1/0

Naïve Bayes Model

Assume features $x_1..x_m$ are independent given the label y:

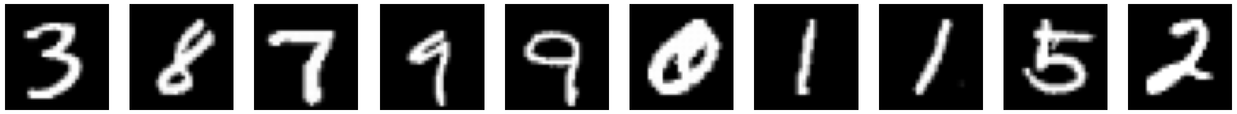
$$P(\boldsymbol{x}|y) = \prod_i P(x_i|y)$$

Then

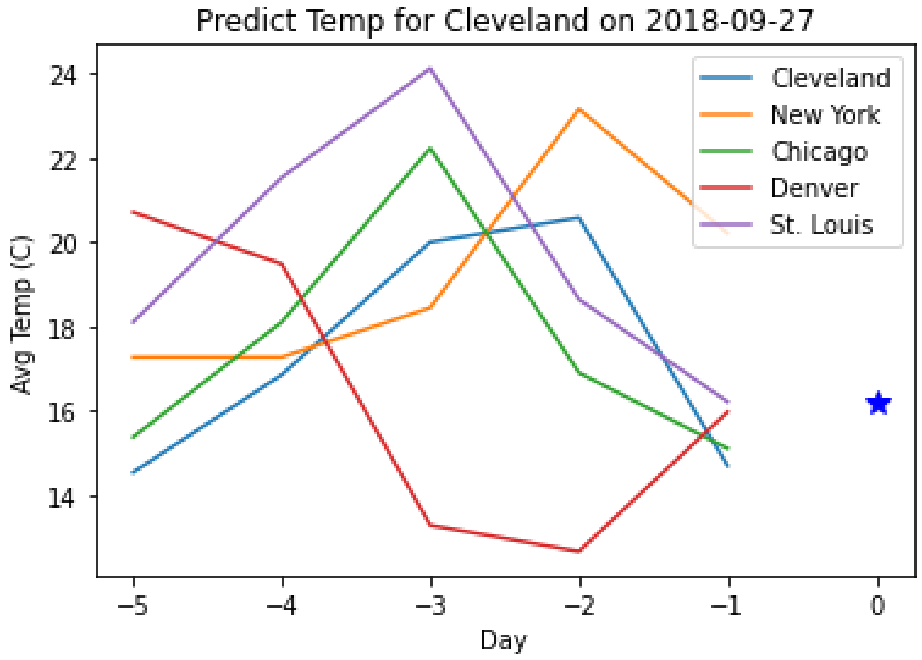
$$y^* = \operatorname*{argmax}_y \prod_i P(x_i|y)P(y)$$

Examples

- Digit classification: choose the label that maximizes the product of likelihoods of each pixel intensity



- Temperature prediction: each feature predicts y with some offset and variance ($y - xi$ is univariate Gaussian)



Naïve Bayes Algorithm

- Training
  1. Estimate parameters for $P(x_i|y)$ for each i
  2. Estimate parameters for $P(y)$
- Prediction
  1. Solve for y that maximizes $P(x, y)$

$$y^* = \underset{y}{\operatorname{argmax}} \prod_i P(x_i|y)P(y)$$

How to estimate $P(x_i|y)$ from data?

- Basic principles of fitting likelihood parameters from data
  - MLE (maximum likelihood estimation): Choose the parameter that maximizes the likelihood of the data
  - MAP (maximum a priori): Choose the parameter that maximizes the data likelihood and its own prior
    - As Warren Buffet says, it's not just about maximizing expected return – it's about making sure there are no zeros.

How to estimate $P(x_i|y)$ from data?

- Bernoulli (x is binary; y is discrete)

$$P(x_i|y=k) = \theta_{ki}^{x_i}(1-\theta_{ki})^{1-x_i}$$

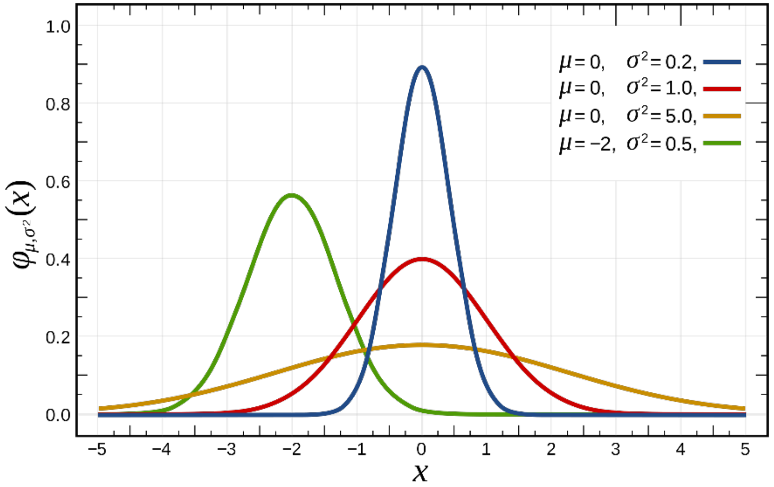
$$\theta_{ki} = \sum_n \delta(x_{ni}=1, y_n=k) / \sum_n \delta(y_n=k)$$

```
theta_ki[k,i] = np.sum((X[:,i]==1) & (y==k)) / np.sum(y==k)
```

- Categorical (x is has multiple discrete values, y is discrete)

$$\theta_{kiv} = \sum_n \delta(x_{ni}=v, y_n=k) / \sum_n \delta(y_n=k)$$

How to estimate $P(x_i|y)$ from data?



- x_i is Gaussian (aka Normal), y is discrete

$$P(x_i|y=k) = \frac{1}{\sqrt{2\pi}\,\sigma_{ki}} \cdot \exp\left(-\frac{1}{2}\frac{(x_i-\mu_{ki})^2}{\sigma_{ki}^2}\right)$$

$$\mu_{ki} = \sum_n \left[X_{ni} \cdot \delta(y_n=k)\right] / \sum_n \delta(y_n=k)$$

$$\sigma_{ki}^2 = \sum_n \left[(X_{ni}-\mu_{ki})^2 \cdot \delta(y_n=k)\right] / \sum_n \delta(y_n=k)$$

How to estimate $P(x_i|y)$ from data?

- $(y - xi)$ is Gaussian

$$P(y-x_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{1}{2}\frac{(y-x_i-\mu_i)^2}{\sigma_i^2}\right)$$

$$\mu_i = \sum_n^N (y-x_i)/N$$

$$\sigma_i^2 = \sum_n^N (y-x_i-\mu_i)^2/N$$

```
mu[i] = np.mean(y-X[:,i], axis=0)
std[i] = np.std(y-X[:,i], axis=0)
```

How to estimate $P(x_i|y)$ from data?

- x_i and y are jointly Gaussian

$$P(x_i|y) = N([x_i, y]; \underline{\mu}_i, \underline{\Sigma}_i) / N(y; \mu_y, \sigma_y)$$

  – $N(.)$ stands for normal distribution with given value, mean, and (co-)variance

How to estimate $P(x_i|y)$ from data?

- x_i is continuous (non-Gaussian), y is discrete
  - First turn x into discrete (e.g. if values range [0, 1), assign `x=floor(x*10)`
  - Now can estimate as categorical

How to estimate $P(x_i|y)$ from data?

- If x is text, e.g. "blue", "orange", "green"
  - Map each possible text value into an integer and solve as categorical

How to estimate $P(y)$?

Three options:

- Assume that y is "uniform" (every value is equally likely) and ignore
- If y is discrete, count
- If y is continuous, model as Gaussian or convert to discrete and count

Stretch break: Simple Naive Bayes example

- Suppose I want to classify a fruit based on description
  - Features: weight, color, shape, whether it’s hard
  - E.g.
    - 0.5 lb, “red”, “round”, yes
    - 15 lb, “green”, “oval”, yes
    - 0.01 lb, “purple”, “round”, no

Q1: What are these three fruit?

Q2: How might you model $P(x_i|\text{fruit})$ for each of these four features?

Simple Naive Bayes example

- Suppose I want to classify a fruit based on description
  - Features: weight, color, shape, whether it's hard
  - E.g.
    - 0.5 lb, "red", "round", yes — Apple
    - 15 lb, "green", "oval", yes — Watermelon
    - 0.01 lb, "purple", "round", no — Grape
  - Model P(weight | fruit) as a Gaussian
  - Model P(color | fruit) as a discrete distribution (multinomial)
  - Model P(shape | fruit) as a categorical
  - Model P(is_hard | fruit) as a Bernoulli (binary)

How to predict y from x?

$$y^* = \arg\max_y \prod_i P(x_i | y)\, P(y)$$
$$= \arg\max_y \sum_i \log P(x_i | y) + \log P(y)$$

If y is discrete:

1. Compute $P(x, y)$ for each value of y
2. Choose value with maximum likelihood

Turning product into sum of logs is an important frequently used trick for argmax/argmin!

How to predict y from x when $(y - x_i)$ is Gaussian

If y is continuous,

$$\frac{\partial}{\partial y} \sum_i \log P(x_i|y) + \log P(y) = 0$$

General formulation (set partial derivative wrt y of $\log P(x, y)$ to 0)

$$\frac{\partial}{\partial y} \sum_i -\frac{1}{2}\frac{(y - x_i - \mu_i)^2}{\sigma_i^2} - \frac{1}{2}\frac{(y-\mu_y)^2}{\sigma_y^2} = 0$$

Example of Temperature regression: $y - x_i$ is Gaussian

$$\frac{\partial}{\partial y} \sum_i \frac{1}{2}\frac{-y^2}{\sigma_i^2} + \frac{yx_i}{\sigma_i^2} + \frac{y\mu_i}{\sigma_i^2} \quad \frac{-1y^2}{2\sigma_y^2} + \frac{y\mu_y}{\sigma_y^2} = 0$$

$$\sum_i \left(\frac{-y}{\sigma_i^2} + \frac{x}{\sigma_i^2} + \frac{\mu_i}{\sigma_i^2} \right) - (y - \mu_y)/\sigma_y^2 = 0$$

$$y\left(\sum_i \frac{1}{\sigma_i^2} + \frac{1}{\sigma_y^2}\right) = \sum_i \frac{x+\mu_i}{\sigma_i^2} + \frac{\mu_y}{\sigma_y^2}$$

$$y = \frac{1}{\sum_i \frac{1}{\sigma_i^2} + \frac{1}{\sigma_y^2}} \cdot \left[\sum_i \frac{x+\mu_i}{\sigma_i^2} + \frac{\mu_y}{\sigma_y^2} \right]$$

$$y = \frac{1}{\sum_i w_i + w_y} \cdot \left[\sum_i (x + \mu_i) w_i + \mu_y \cdot w_y \right]$$

Prediction is weighted average of means, where weights are inverse variance

$$P(x_i|y) \sim N(y - x_i, \sigma^2) = \frac{1}{\sqrt{2\pi}\,\sigma_i} \exp\left(-\frac{1}{2}\frac{(y - x_i - \mu_i)^2}{\sigma_i^2}\right)$$

Using priors

- Priors on the likelihood parameters prevent a single feature from having zero or extremely low likelihood due to insufficient training data
- Discrete: initialize counts with α (e.g. $\alpha = 1$)

  $P(x_i=v|y=k) = (\alpha + count(x_i=v, y=k)) / sum_v[\alpha + count(x_i=v, y=k)]$

```
theta_kiv[k,i,v] = (np.sum((X[:,i]==v) & (y==k))+alpha) / (np.sum(y==k)+alpha*num_v)
```

- Continuous: add some ϵ to the variance (e.g. $\epsilon = 0.1/N$)
  – For multivariate, add to diagonal of covariance

```
std[i] = np.std(y-X[:,i], axis=0)+np.sqrt(0.1/len(X))
```

MLE and MAP estimates of binary variable likelihoods

- MLE (maximize data likelihood)

$$P(x=1|y=1) = \frac{\sum_n \delta(x_n = 1, y_n = 1)}{\sum_n \delta(x_n = 0, y_n = 1) + \sum_n \delta(x_n = 1, y_n = 1)}$$

- MAP (maximum a posteriori) with prior α

$$P(x=1|y=1) = \frac{\alpha + \sum_n \delta(x_n = 1, y_n = 1)}{(\alpha + \sum_n \delta(x_n = 0, y_n = 1)) + (\alpha + \sum_n \delta(x_n = 1, y_n = 1))}$$

- This is a Bayesian prior that implies $P(x=0|y) \approx P(x=1|y)$, unless data tells us differently
- Similar concept to regularization that we saw in linear regression and classification
- Important because it avoids zeros that could dominate the overall likelihood and provides a more stable estimate with limited data
- With more data, the prior has less effect

Example: estimate joint probability under Naïve Bayes assumption

#	x1	x2	y
1	1	1	1
2	0	1	1
3	1	0	0
4	0	1	0
5	1	1	1
6	1	0	0
7	1	0	1
8	0	1	0

$P(x1|y)$

x1	$y=0$	$y=1$
0		
1		

$P(x2|y)$

x2	$y=0$	$y=1$
0		
1		

	$y=0$	$y=1$
$P(y)$		

$P(y=0, x1=1, x2=1) = ?$

$P(y=1, x1=1, x2=1) = ?$

$P(y=0|x1=1, x2=1) = ?$

#	x1	x2	y
1	1	1	1
2	0	1	1
3	1	0	0
4	0	1	0
5	1	1	1
6	1	0	0
7	1	0	1
8	0	1	0

$P(x1|y)$

x1	$y=0$	$y=1$
0	2/4	1/4
1	2/4	3/4

$P(x2|y)$

x2	$y=0$	$y=1$
0	2/4	1/4
1	2/4	3/4

	$y=0$	$y=1$
$P(y)$	2/4	2/4

$$P(y=0, x1=1, x2=1) = 1/8$$

$$P(y=1, x1=1, x2=1) = 9/32$$

$$P(y=0|x1=1, x2=1) = 4/13$$

Prior over parameters: initialize each count with α

#	x1	x2	y
1	1	1	1
2	0	1	1
3	1	0	0
4	0	1	0
5	1	1	1
6	1	0	0
7	1	0	1
8	0	1	0

$\alpha = 1$

$P(x1|y)$

x1	$y = 0$	$y = 1$
0	2/4	1/4
1	2/4	3/4

⇒

x1	$y = 0$	$y = 1$
0	3/6	2/6
1	3/6	4/6

$P(x2|y)$

x2	$y = 0$	$y = 1$
0	2/4	1/4
1	2/4	3/4

⇒

x2	$y = 0$	$y = 1$
0	3/6	2/6
1	3/6	4/6

	$y = 0$	$y = 1$
$P(y)$	2/4	2/4

⇒

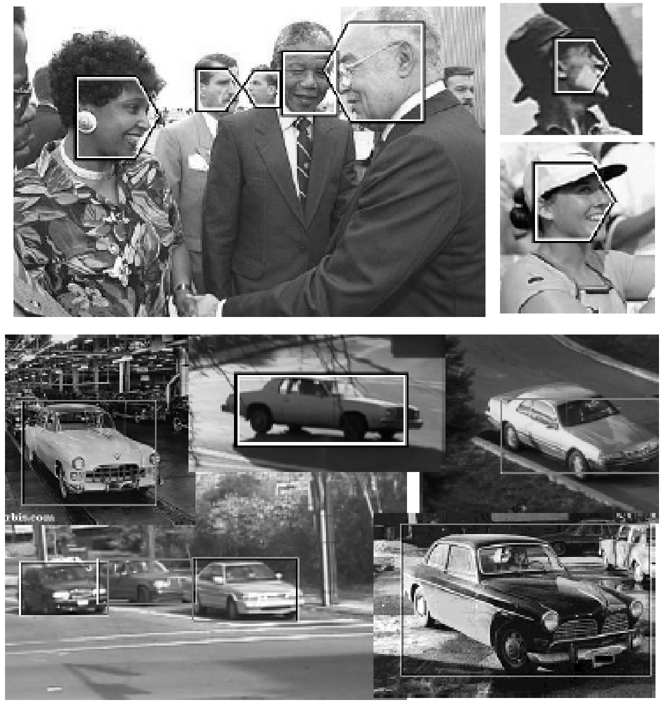
	$y = 0$	$y = 1$
	2/4	2/4

Use case: “Semi-naïve Bayes” object detection

- Best performing face/car detector in 2000-2005
- Model probabilities of small groups of features (wavelet coefficients)
- Search for groupings, discretize features, estimate parameters

A Statistical Method for 3D Object Detection Applied to Faces and Cars

Henry Schneiderman and Takeo Kanade



$$\frac{\prod_{x,y \in \text{region}} \prod_{k=1}^{17} P_k(pattern_k(x,y), x, y \mid \text{object})}{\prod_{x,y \in \text{region}} \prod_{k=1}^{17} P_k(pattern_k(x,y), x, y \mid \text{non-object})} > \lambda$$

https://www.cs.cmu.edu/afs/cs.cmu.edu/user/hws/www/CVPR00.pdf

Naïve Bayes Summary

- Key Assumptions
  - Features are independent, given the labels
- Model Parameters
  - Parameters of probability functions $P(x_i|y)$ and $P(y)$
- Designs
  - Choice of probability function
- When to Use
  - Limited training data
  - Features are not highly interdependent
  - Want something fast to code, train, and test
- When Not to Use
  - Logistic or linear regression will usually work better if there is sufficient data (more flexible / fewer assumptions than Naïve Bayes)
  - Does not provide a good confidence estimate because it "overcounts" influence of dependent variables

Naïve Bayes

- Pros
  - Easy and fast to train
  - Fast inference
  - Can be used with continuous, discrete, or mixed features
- Cons
  - Does not account for feature interactions
  - Does not provide good confidence estimate
- Notes
  - Best when used with discrete variables, variables that are well fit by Gaussian, or kernel density estimation

Things to remember

- Probabilistic models are a large class of machine learning methods

- Naïve Bayes assumes that features are independent given the label
  - Easy/fast to estimate parameters
  - Less risk of overfitting when data is limited

$$P(\boldsymbol{x}, y) = \prod_i P(x_i|y)P(y)$$

- You can look up how to estimate parameters for most common probability models
  - Or take partial derivative of total data/label likelihood given parameter

- Prediction involves finding y that maximizes $P(x, y)$, either by trying all y or solving partial derivative

- Maximizing $\log P(x, y)$ is equivalent to maximizing $P(x, y)$ and often much easier

$$y^* = \operatorname{argmax}_y \prod_i P(x_i|y)\,P(y)$$
$$= \operatorname{argmax}_y \sum_i \log P(x_i|y) + \log P(y)$$

Next week

- EM and Density Estimation