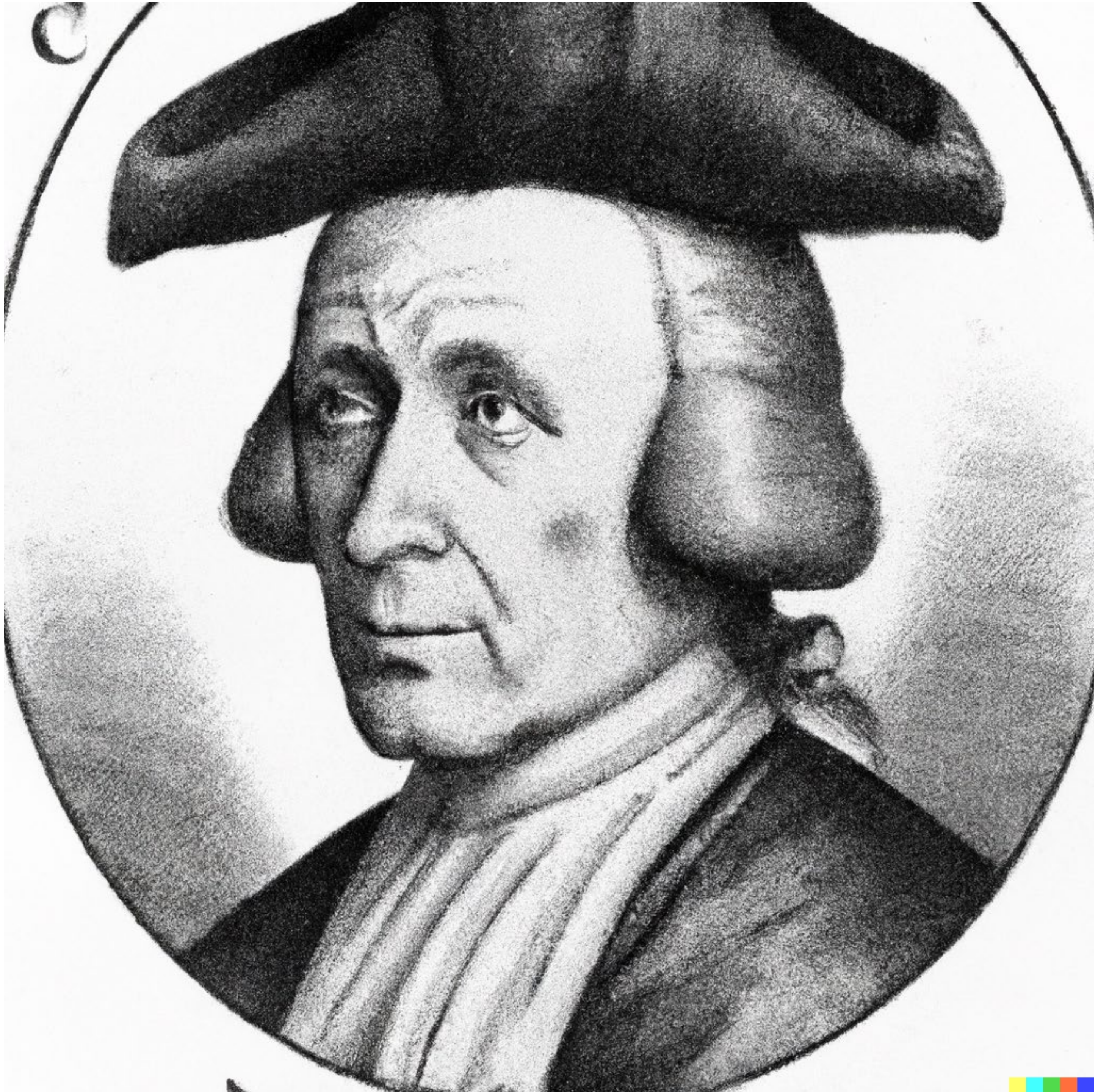


Probability and Naïve Bayes

Applied Machine Learning
Derek Hoiem

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

KNN Usage Example: Deep Face

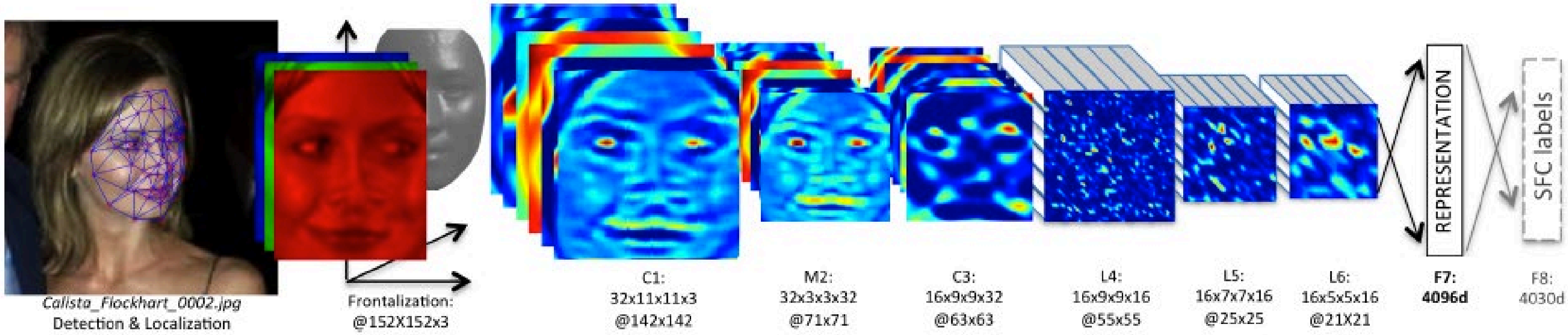
DeepFace: Closing the Gap to Human-Level Performance in Face Verification

Yaniv Taigman   Ming Yang   Marc'Aurelio Ranzato   Lior Wolf

Facebook AI Research
Menlo Park, CA, USA
{yaniv, mingyang, ranzato}@fb.com

Tel Aviv University
Tel Aviv, Israel
wolf@cs.tau.ac.il

CVPR 2014



1. Detect facial features
2. Align faces to be frontal
3. Extract features using deep network while training classifier to label image into person (dataset based on employee faces)
4. In testing, extract features from deep network and use nearest neighbor classifier to assign identity

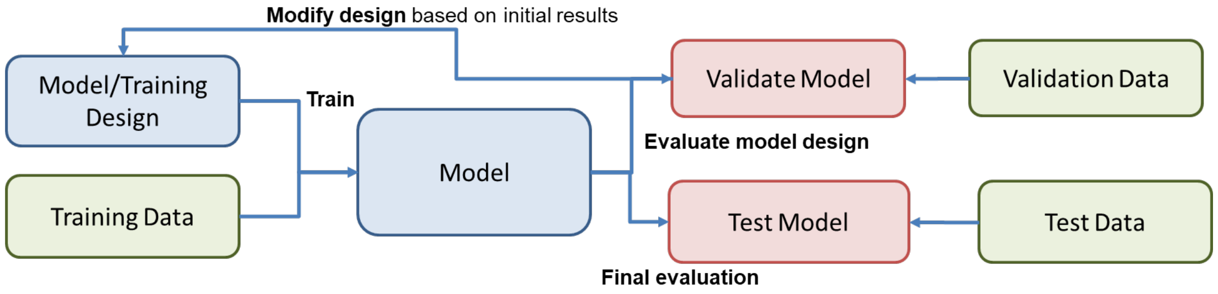
- Performs similarly to humans in the LFW dataset (labeled faces in the wild)
- Can be used to organize photo albums, identifying celebrities, or alert user when someone posts an image of them
- This algorithm is used by Facebook (though with expanded training data)
- Think about potential unintended consequences, e.g. due to how features are trained or its application

KNN Summary

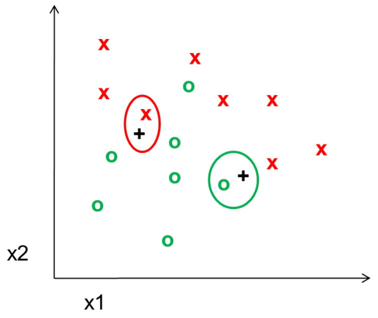
- Key Assumptions
  - Samples with similar input features will have similar output predictions
  - Depending on distance measure, may assume all dimensions are equally important
- Model Parameters
  - Features and predictions of the training set
- Designs
  - K (number of nearest neighbors to use for prediction)
  - How to combine multiple predictions if K > 1
  - Feature design (selection, transformations)
  - Distance function (e.g. L2, L1, Mahalanobis)
- When to Use
  - Few examples per class, many classes
  - Features are all roughly equally important
  - Training data available for prediction changes frequently
  - Can be applied to classification or regression, with discrete or continuous features
  - Most powerful when combined with feature learning
- When Not to Use
  - Many examples are available per class (feature learning with linear classifier may be better)
  - Limited storage (cannot store many training examples)
  - Limited computation (linear model may be faster to evaluate)

Things to remember (from last class)

- Supervised machine learning involves:
  1. Fitting parameters to a model using training data
  2. Refining the model based on validation performance
  3. Evaluating the final model on a held out test set

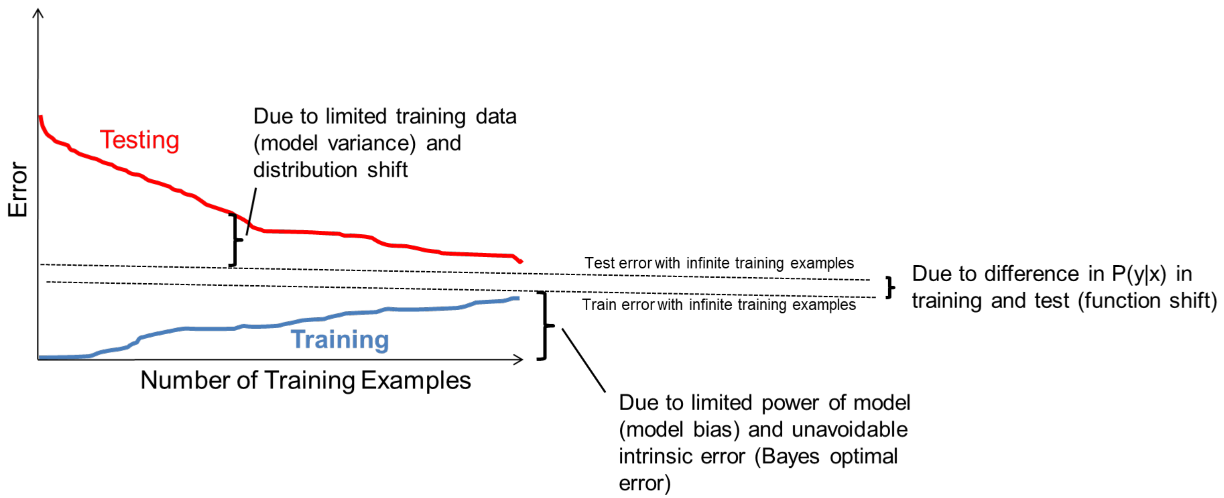


- KNN is a simple but effective classifier/regressor that predicts the label of the most similar training example(s)



- With more samples, fitting the training data becomes harder, but test error is expected to decrease

- Test errors have many sources
  - intrinsic to problem
  - model bias / limited power
  - model variance / limited training data
  - differences in training and test distributions



- Model design and fitting is just one part of a larger process in collecting data, developing, and deploying models

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

Probabilistic model

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

Or equivalently…

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

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

Example

x: Larger than 10 lbs?

y		F	T
	Cat	15	25
	Dog	5	40

$$P(y = Cat) =$$

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

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

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

A is independent of B if (and only if)

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

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

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

Estimate probabilities of discrete variables by counting

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

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

Naïve Bayes Model

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

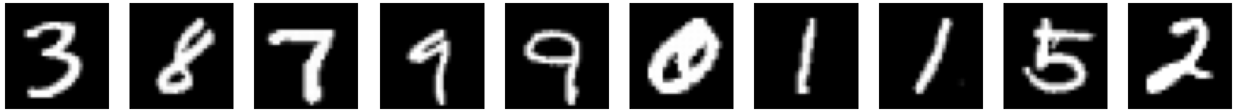
$$P(\boldsymbol{x}|y) = \prod_i P(x_i|y)$$

Then

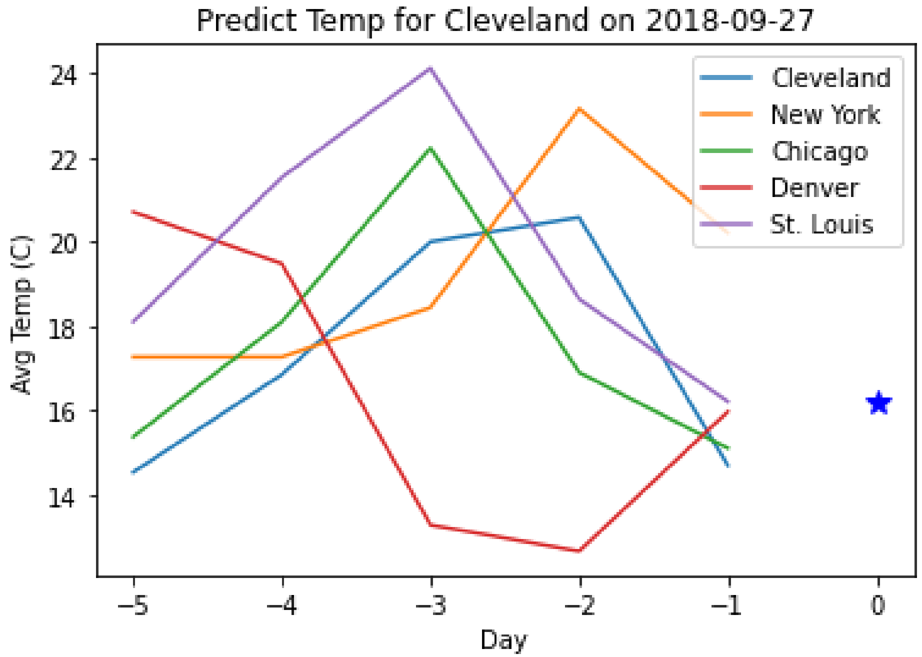
$$y^* = \underset{y}{\operatorname{argmax}} \prod_i P(x_i|y)P(y)$$

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 ($y-x_i$ 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?

- Binomial (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)
```

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

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

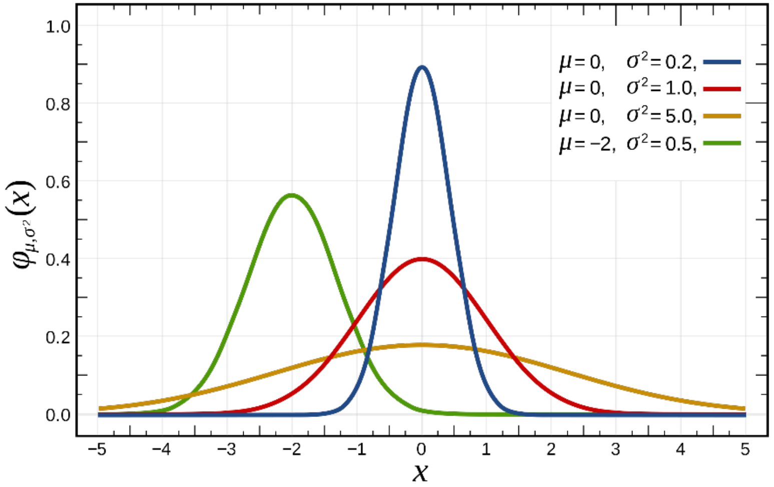
How to estimate $P(x_i|y)$ from data?

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

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

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

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



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

- $(y\text{-}x_i)$ 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 covariance or 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 multinomial

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 multinomial

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 multinomial
  - Model P(is_hard | fruit) as a Bernoulli (binary)

How to predict y from x?

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

$$= \underset{y}{\operatorname{argmax}} \sum_i \log P(x_i | y) + \log P(y)$$

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?

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: $y-x_i$ is Gaussian (HW1)

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

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

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

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

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

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

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. $\alpha = 1$)

  $P(x_i=v|y=k) = (\alpha + \text{count}(x_i=v, y=k)) / \text{sum}_v[\alpha + \text{count}(x_i=v, y=k)]$

```
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))
```

Example

#	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		

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

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

Example

#	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

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

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

⇒

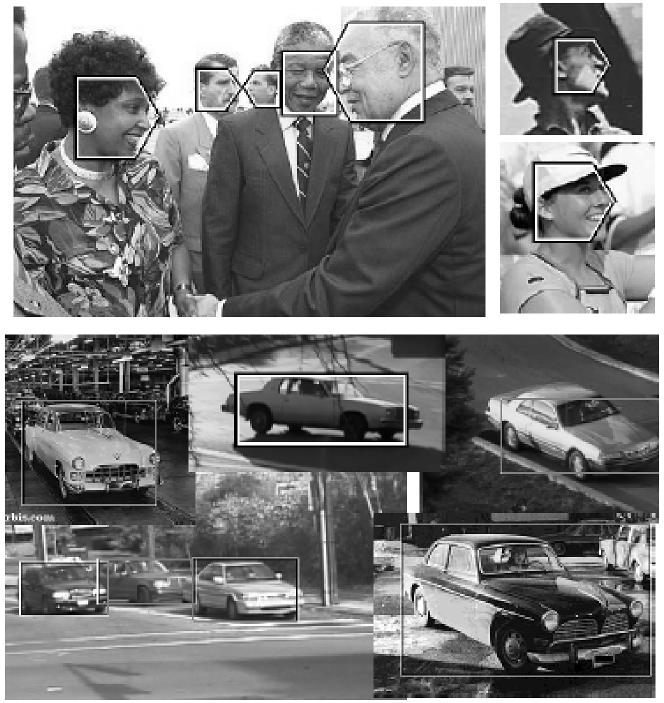
	$y = 0$	$y = 1$
	2/4	2/4

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 class

- Logistic Regression and Linear Regression