DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

October 20 2020

Announcements

- HW 3 is out
  - Due on Thu, Oct. 29
- Project proposal
  - Due on Monday, Nov. 2
  - Team of 2
  - Resources and example projects on Piazza

Project Proposal

1 Page

- Project Title
- Project Team
- Problem Description
  - What is the prediction problem you are trying to solve?
- Dataset
  - Link to data, brief description, number of records, feature dimensionality (at least 10K records)
- Approach and methodology
  - Normalization
  - Feature selection
  - Machine learning models you will try
  - Splitting into training and testing, cross validation
  - Language and packages you plan to use
- Metrics (how you will evaluate your models)
- References
  - How did you find out about the dataset, did anyone else used the data for a similar prediction task

Outline

- Generative vs Discriminative Models
- Linear Discriminant Analysis (LDA)
  - LDA is a linear classifier
  - LDA vs Logistic Regression
  - Lab
- Density Estimation
- Naïve Bayes classifier

Generative vs Discriminative

- Generative model
  - Given X and Y, learns the joint probability $P(X, Y)$
  - Can generate more examples from distribution
  - Examples: LDA, Naïve Bayes, language models (GPT-2)

- Discriminative model
  - Given X and Y, learns a decision function for classification
  - Examples: logistic regression, kNN

LDA

Bayes Th

$$P[A|B] = \frac{P[B|A]\,P[A]}{P[B]}$$

- Classify to one of k classes
- Logistic regression computes directly
  - $\boxed{P[Y = 1|X = x]}$
  - Assume sigmoid function

LDA: $P[\underbrace{Y=k}_{\text{label}} \mid \underbrace{X=x}_{\text{data}}] = \dfrac{P[X=x \mid Y=k]\,\boxed{P[Y=k]}}{P[X=x]}$

$\pi_k = P[Y=k]$ PRIOR PB OF CLASS k

$x = (x_1, \dots, x_d)$ dim d

FEATURE VECTOR

$f_k(x) = P[X=x \mid Y=k]$

LDA

$\boxed{\text{Assume } f_k(x) \text{ is Gaussian!}}$ $\sim \mathcal{N}(\mu_k, \sigma_k)$

Unidimensional case (d=1)

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right)$$

std. deviation

mean

f_k is Gaussian for every class k

$$\underbrace{f_k(x) = P[X = x \mid Y = k]}$$

μ_k – mean of X in class k

σ_k – std. dev.

ESTIMATED FROM TRAINING DATA

Continuous Random Variables

- X:U⟶V is continuous RV if it takes infinite number of values
- The **cumulative distribution function CDF** $F: R \longrightarrow \{0,1\}$ for X is defined for every value x by:

$$\boxed{F(x) = \Pr(X \leq x)}$$

$$E[X] = \int_{-\infty}^{\infty} x\, f(x)\, dx$$

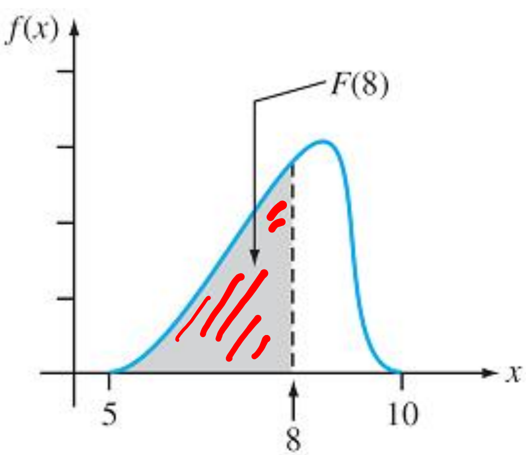
- The **probability distribution function PDF** $f(x)$ for X *is*

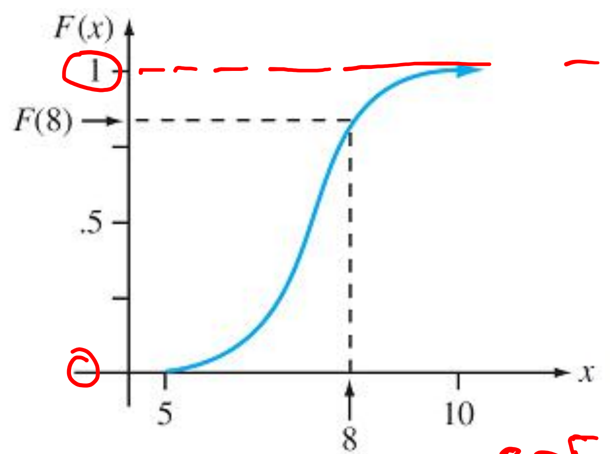
$$f(x) = dF(x)/dx$$

Discrete $E[X] = \sum r\, P[X = r]$

Increasing

PDF

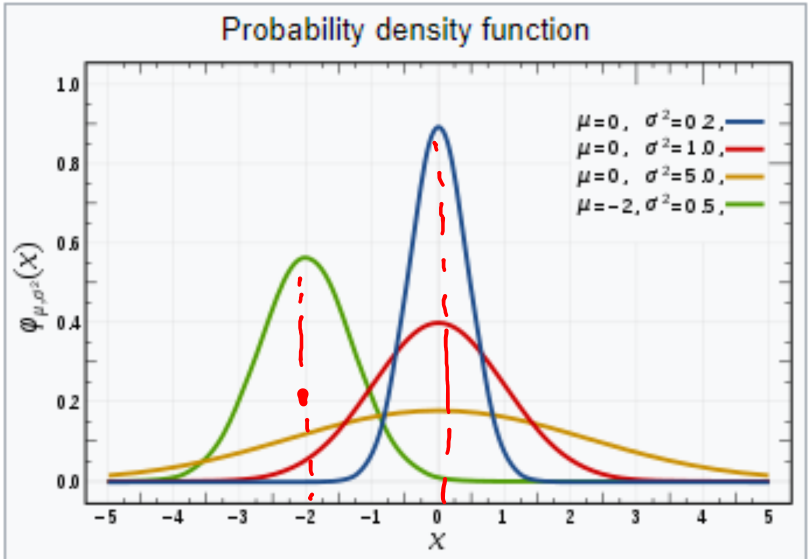




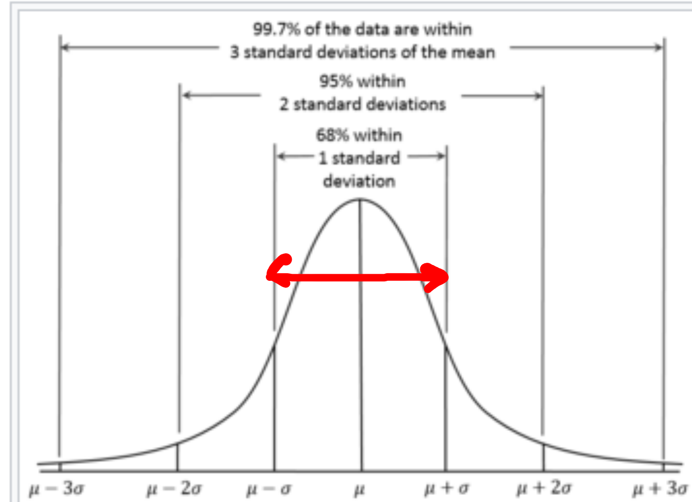
CDF

Gaussian Distribution

Normal Distribution



The red curve is the *standard normal distribution*



For the normal distribution, the values less than one standard deviation away from the mean account for 68.27% of the set; while two standard deviations from the mean account for 95.45%; and three standard deviations account for 99.73%.

Notation	$\mathcal{N}(\mu, \sigma^2)$
Parameters	$\mu \in \mathbb{R}$ = mean (location) $\sigma^2 > 0$ = variance (squared scale)
Support	$x \in \mathbb{R}$
PDF	$\boxed{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}}$

LDA

Assume $f_k(x)$ is Gaussian!
Unidimensional case (d=1)

ASSUME $\boxed{\sigma_1 = \sigma_2 = \ldots = \sigma_K = \sigma}$

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2 \right)$$

$p_k(x)$

$$= P[Y=k \mid X=x] = \frac{P[X=x \mid Y=k]\,P[Y=k]}{P[X=x]} = \frac{f_k(x)\cdot\pi_K}{P[X=x]}$$

$$= \boxed{\frac{\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)\cdot\pi_K}{\boxed{P[X=x]}}}$$

$$P[X=x] = \sum_k P[X=x \mid Y=k]\,P[Y=k]$$

NORMALIZING CONSTANT (C)

Pick k that max
$P[Y=k \mid X=x]$

LDA decision boundary

$p_k(x) = \frac{\pi_k f_k(x)}{C}$ Pick k to max $p_k(x)$

$\log p_k(x) = \log \pi_k + \log f_k(x) - \log C$

$= \log \pi_k + \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2}(x-\mu_k)^2 - \log C$

$= \boxed{\log \pi_k} + \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2}x^2 + \boxed{\frac{1}{2\sigma^2} x \cdot \mu_k} - \boxed{\frac{\mu_k^2}{2\sigma^2}} - \log C$

$\delta_k(x) = \log \pi_k + \frac{1}{2\sigma^2} x \mu_k - \frac{\mu_k^2}{2\sigma^2}$

Given x: Pick k that max $p_k(x)$

Equivalent to max $\delta_k(x)$

LDA decision boundary

Pick class k to maximize

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

Example: $k = 2, \pi_1 = \pi_2$ EQUAL PRIORS

$$\delta_1(x) = x \frac{\mu_1}{\sigma^2} - \frac{\mu_1^2}{2\sigma^2} + \boxed{\log(\pi_1)}$$

$$\delta_2(x) = x \cdot \frac{\mu_2}{\sigma^2} - \frac{\mu_2^2}{2\sigma^2} + \boxed{\log(\pi_2)}$$

CLASSIFY CLASS 1 IF $\delta_1(x) > \delta_2(x)$

$$x \cdot \frac{\mu_1}{\sigma^2} - \frac{\mu_1^2}{2\sigma^2} > x \cdot \frac{\mu_2}{\sigma^2} - \frac{\mu_2^2}{2\sigma^2} \iff 2x\mu_1 - \mu_1^2 > 2x\mu_2 - \mu_2^2$$

$$2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2 \Rightarrow x > \frac{\mu_1 + \mu_2}{2}$$

LDA decision boundary

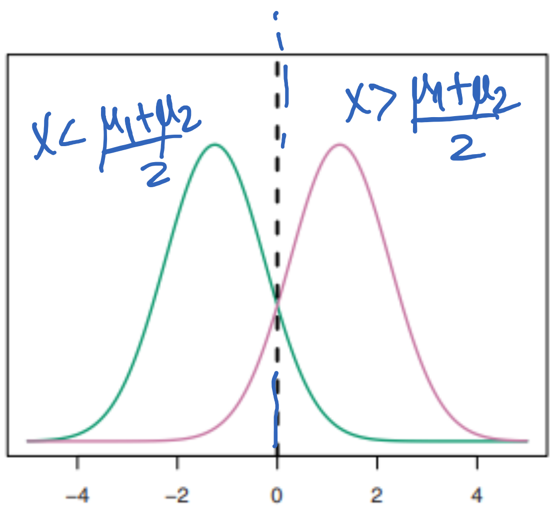
Pick class k to maximize

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

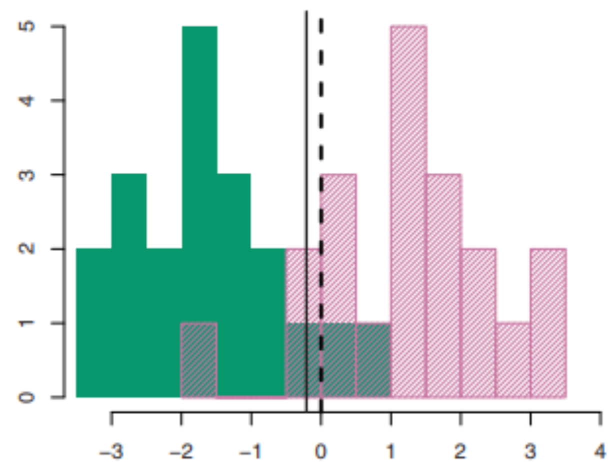
Example: $k = 2, \pi_1 = \pi_2$

Classify as class 1 if $x > \frac{\mu_1+\mu_2}{2\cancel{\sigma}}$

LINEAR



True decision boundary



Estimated decision boundary

LDA in practice

SINGLE DIM

Given training data $(x_i, y_i), i = 1, \ldots, n, y_i \in \{1, \ldots, K\}$

1. Estimate mean and variance

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

2. Estimate prior

$$\hat{\pi}_k = n_k/n.$$

Given testing point x, predict k that maximizes:

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

LINEAR

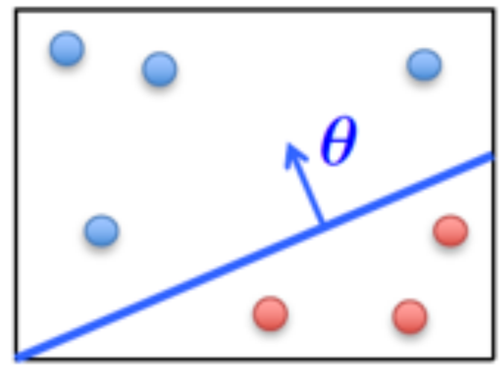
Linear models

- Perceptron

$$h(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{\theta}^\top \boldsymbol{x})$$

- Logistic regression

$$\boxed{h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\top \boldsymbol{x}}}}$$



- LDA

$$Max_k \;\; \delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

LDA vs Logistic Regression

- Logistic regression computes directly $\Pr[Y = 1|X = x]$ by assuming sigmoid function
  - Uses Maximum Likelihood Estimation
  - Discriminative Model
- LDA uses Bayes Theorem to estimate it
  - Estimates mean, co-variance, and prior from training data
  - Generative model
  - Assumes Gaussian distribution for $f_k(x) = \Pr[X = x|Y = k]$
- Which one is better?
  - LDA can be sensitive to outliers
  - LDA works well for Gaussian distribution
  - Logistic regression is more complex to solve, but more expressive

Linear Classifier Lab

```
data = pd.read_csv('heart.csv')
data = data.dropna()
x_columns = data.columns != 'target'
data = utils.shuffle(data)
data.head()
```

HEART COND

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
215	43	0	0	132	341	1	0	136	1	3.0	1	0	3	0
145	70	1	1	156	245	0	0	143	0	0.0	2	0	2	1
190	51	0	0	130	305	0	1	142	1	1.2	1	0	3	0
90	48	1	2	124	255	1	1	175	0	0.0	2	2	2	1
166	67	1	0	120	229	0	0	129	1	2.6	1	2	3	0

https://www.kaggle.com/ronitf/heart-disease-uci

Lab LDA

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(x_train, y_train)
print('Priors:')
print(lda.priors_)
print('Means:')
print(lda.means_)
print('Coefficients:')
print(lda.coef_)
print('Test Accuracy:')
print(lda.score(x_test, y_test))
```

```
Priors:
[0.41409692 0.58590308]
Means:
[[5.70744681e+01 8.19148936e-01 4.78723404e-01 1.34882979e+02
  2.49031915e+02 1.27659574e-01 4.36170213e-01 1.40021277e+02
  5.21276596e-01 1.62446809e+00 1.18085106e+00 1.24468085e+00
  2.57446809e+00]
 [5.24060150e+01 5.48872180e-01 1.36090226e+00 1.29548872e+02
  2.45052632e+02 1.27819549e-01 5.93984962e-01 1.59195489e+02
  1.35338346e-01 5.84962406e-01 1.64661654e+00 3.30827068e-01
  2.12030075e+00]]
Coefficients:
[[-5.12655671e-03 -1.65128336e+00  9.42708811e-01 -1.63429905e-02
  -8.26945654e-05  3.61220910e-01  6.53320414e-01  2.61543171e-02
  -1.10225766e+00 -5.26885663e-01  9.83938578e-01 -1.00983532e+00
  -1.16829536e+00]]
Test Accuracy:
0.8026315789473685
```

LDA Metrics

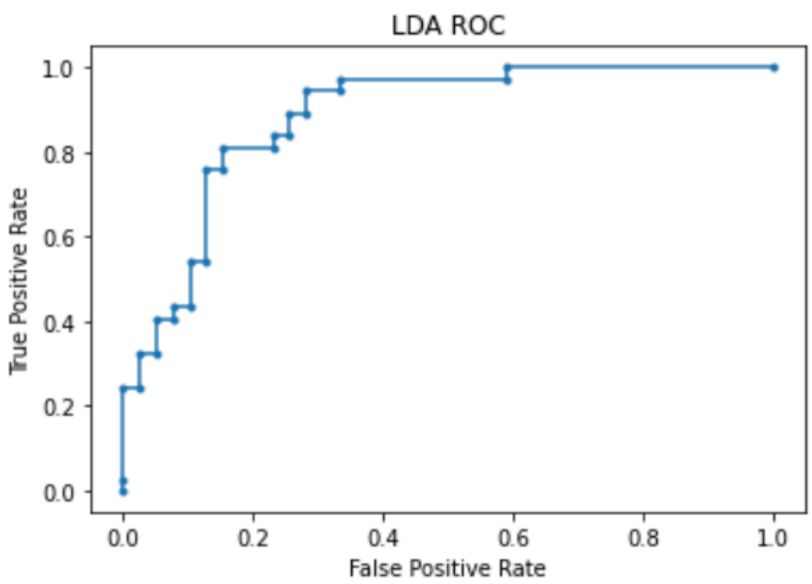
```
target_names = ['class 0', 'class 1']
pred_label_lda = lda.predict(x_test)
print(classification_report(y_test, pred_label_lda, target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 0       0.88      0.72      0.79        39
     class 1       0.75      0.89      0.81        37

    accuracy                           0.80        76
   macro avg       0.81      0.80      0.80        76
weighted avg       0.81      0.80      0.80        76
```

LDA ROC Curve

```
pred_lda = lda.predict_proba(x_test)[:,1]
r_auc_lda = roc_auc_score(y_test, pred_lda)
print("AUC=",r_auc_lda)

lda_fpr, lda_tpr, _ = roc_curve(y_test, pred_lda)
pyplot.plot(lda_fpr, lda_tpr, marker='.', label='Logistic')
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.title('LDA ROC')
```

```
AUC= 0.8842688842688843
```



Outline

- Generative vs Discriminative Models
- Linear Discriminant Analysis (LDA)
  – LDA is a linear classifier
  – LDA vs Logistic Regression
  – Lab
- Density Estimation
- Naïve Bayes classifier

Essential probability concepts

- Marginalization: $P(B) = \sum_{v \in \text{values}(A)} P(B \wedge A = v)$

- Conditional Probability: $P(A \mid B) = \dfrac{P(A \wedge B)}{P(B)}$

- Bayes' Rule: $P(A \mid B) = \dfrac{P(B \mid A) \times P(A)}{P(B)}$

- Independence:

$$A \perp\!\!\!\perp B \quad \leftrightarrow \quad \left\{ \begin{array}{l} P(A \wedge B) = P(A) \times P(B) \\ \leftrightarrow \; P(A \mid B) = P(A) \end{array} \right.$$

INDEPENDENCE

$$A \perp\!\!\!\perp B \mid C \quad \leftrightarrow \quad \underbrace{P(A \wedge B \mid C) = P(A \mid C) \times P(B \mid C)}$$

CONDITIONAL INDEPENDENCE

Prior and Joint Probabilities

- **Prior probability**: degree of belief without any other evidence
- **Joint probability**: matrix of combined probabilities of a set of variables

Russell & Norvig's Alarm Domain: (boolean RVs)

- A world has a specific instantiation of variables:

  $(\text{alarm} \wedge \text{theft} \wedge \neg\text{earthquake})$

- The joint probability is given by:

$P(\underline{\text{Alarm}}, \underline{\text{Theft}}) =$

	alarm	¬alarm
theft	0.09	0.01
¬theft	0.1	0.8

$P[\text{THEFT}] = 0.09 + 0.01 = 0.1$

$P[T] = \underbrace{P[T \cap A] + P[T \cap \neg A]}_{} = P[T|A]P[A] + P[T|\neg A]P[\neg A]$

MARGINALIZATION

Computing Prior Probabilities

	alarm		¬alarm	
	earthquake	¬earthquake	earthquake	¬earthquake
theft	0.01	0.08	0.001	0.009
¬ theft	0.01	0.09	0.01	0.79

$$P[T] = \sum_{a,e} P[T \cap \text{Alarm} = a \cap \text{Earthquake} = e] = 0.1$$

$$P[A] = 0.19$$

The Joint Distribution

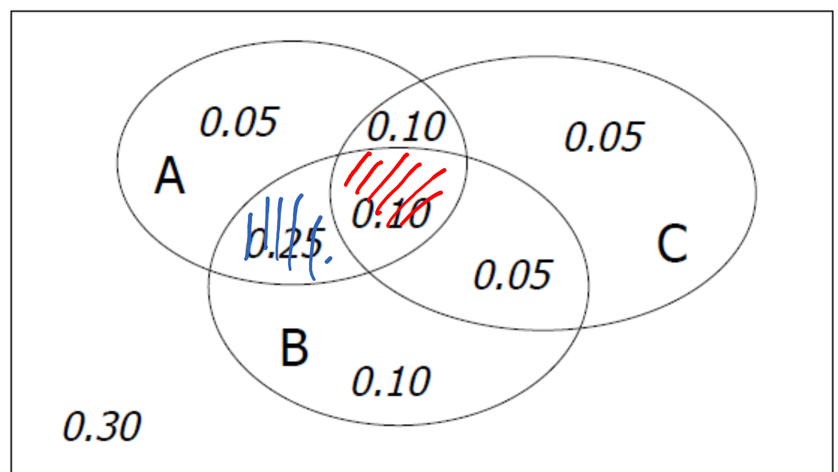
Recipe for making a joint distribution of d variables:

1. Make a truth table listing all combinations of values of your variables (if there are d Boolean variables then the table will have 2^d rows).
2. For each combination of values, say how probable it is.
3. If you subscribe to the axioms of probability, those numbers must sum to 1.

e.g., Boolean variables A, B, C

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10

8 entries



Slide © Andrew Moore

Learning Joint Distributions

Step 1:
Build a JD table for your attributes in which the probabilities are unspecified

A	B	C	Prob
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

Step 2:
Then, fill in each row with:

$$\hat{P}(\text{row}) = \frac{\text{records matching row}}{\text{total number of records}}$$

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	**0.25**
1	1	1	0.10

Fraction of all records in which
A and B are true but C is false

Example – Learning Joint Probability Distribution

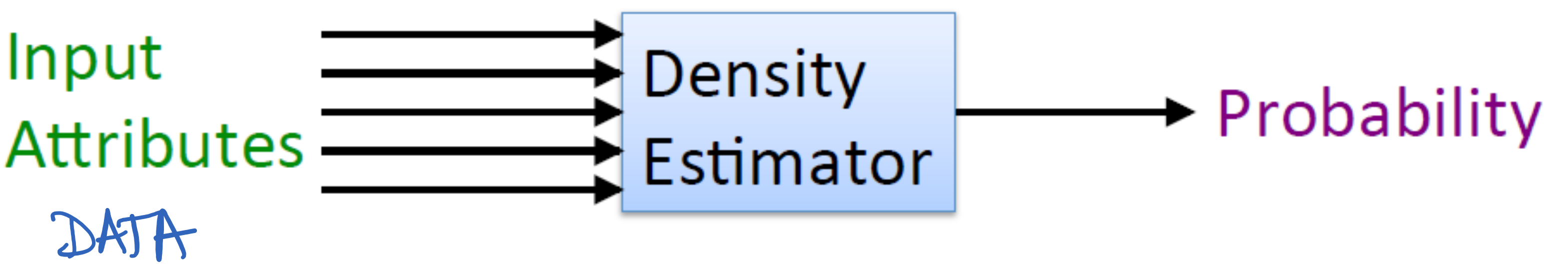
This Joint PD was obtained by learning from three attributes in the UCI "Adult" Census Database [Kohavi 1995]

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

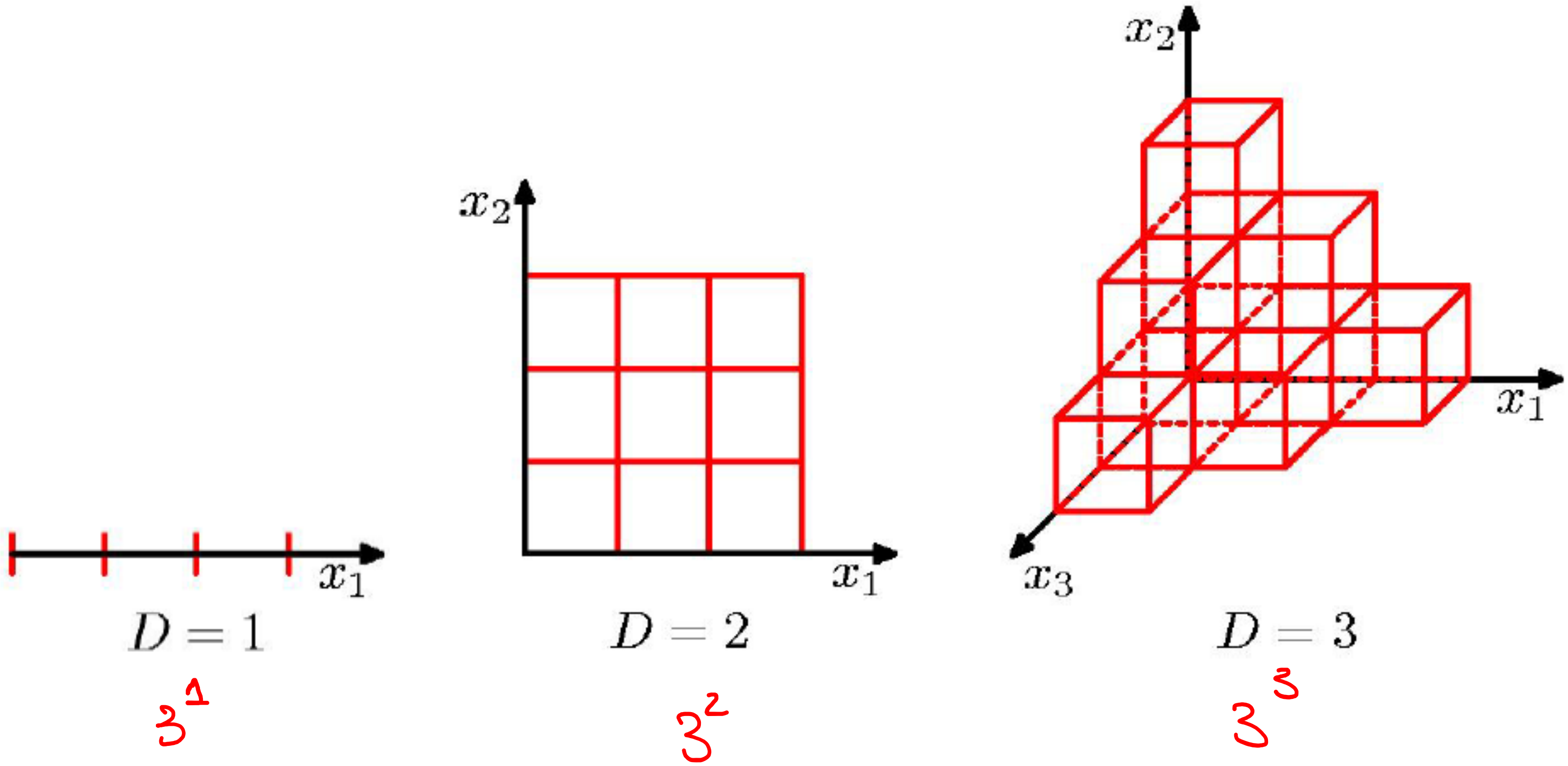
Density Estimation

- Our joint distribution learner is an example of something called **Density Estimation**
- A Density Estimator learns a mapping from a set of attributes to a probability

METHODS :- JOINT DENSITY TABLES (DISCRETE)
- GAUSSIAN ASSUMPTION - LDA (CONTINOUS)
- KERNEL DENSITY ESTIMATION (CONTINOUS)
KDE



Curse of Dimensionality



Naïve Bayes Classifier

$$P[Y=k \mid X=x] = \frac{\boxed{P[X=x \mid Y=k]}\,\boxed{P[Y=k]}}{P[X=x]}$$

Bayes Th

Similar to LDA

$\pi_k = P[Y=k]$ PRIOR PB.

$$P[X=x \mid Y=k] = P[\underbrace{X_1=x_1 \cap \ldots \cap X_d=x_d}_{} \mid Y=k]$$

2^d values

$x=(x_1, \ldots, x_d)$

DISCRETE RV. | FEATURES

ASSUMPTION: $X_1, \ldots, X_d$ are CONDITIONALLY INDEP. GIVEN Y

$$P[X_1=x_1 \cap \ldots \cap X_d=x_d \mid Y=k] = P[X_1=x_1 \mid Y=k] \cdot \ldots \cdot P[X_d=x_d \mid Y=k]$$

$$P[X=x \mid Y=k] = \prod_{j=1}^{d} \underbrace{P[X_j=x_j \mid Y=k]}_{}$$

$k \cdot d$ values to estimate

Naïve Bayes Classifier

Problem: estimating the joint density isn't practical

- Severely overfits, as we saw before

However, if we make the assumption that the attributes are independent given the class label, estimation is easy!

Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
- Thanks!