Classification with generative models

based on PDFs

ECE407

Machine learning using pdfs

The goal is to develop

procedures that exhibit a desired input-output behavior.

- Example Recognition Problem from Computer Vision

  Input: Picture of an animal.
  Output: Name of the animal.

Learning (or training of pdfs from data)

- We estimate PDFs from observed data.

- "We simply provide examples of (input,output) pairs and ask the machine to *learn* a suitable mapping itself."

- There are many ways of learning (Neural Networks, Generative PDF Models etc)

- In this lecture we use PDFs for learning and recognition

Inputs and outputs

Example 1:

- The input space, $\mathcal{X}$.
  E.g. 32×32 RGB images of animals.
- The output space, $\mathcal{Y}$.
  E.g. Names of 100 animals.

x:

y: "bear"

MNIST Digit Recognition Example:

The input: 28x28 hand-written digit image in gray-scale

The output: Value of the digit
Sample space for Y = {0,1,2,...,9}
http://yann.lecun.com/exdb/mnist/index.html
MATLAB reader:
https://www.mathworks.com/matlabcentral/fileexchange/27675-read-digits-and-labels-from-mnist-database?

A basic classifier: nearest neighbor (deterministic)

Given a labeled training set $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$.

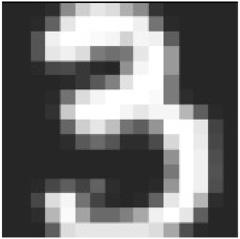
Example: the MNIST data set of handwritten digits.

To classify a new instance x:

- Find its nearest neighbor amongst the $x^{(i)}$
- Return $y^{(i)}$

The data space (Nearest Neighbor does not use probability values)during the recognition phase)

We need to choose a distance function.

$x=$ 

Each image is 28×28 grayscale.
One option: Treat images as 784-dimensional vectors, and use Euclidean (ℓ_2) distance:

$$\|x - x'\| = \sqrt{\sum_{i=1}^{784} (x_i - x'_i)^2}.$$

x' is the images of the labeled numbers in our database
x is the digit that we want to recognize.

Summary:

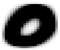
- Data space $\mathcal{X} = \mathbb{R}^{784}$ with ℓ_2 distance
- Label space $\mathcal{Y} = \{0, 1, \ldots, 9\}$

Simple Digit Recognizer:

Training:

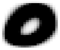
Training set has 60,000 digit images.

Estimate the mean images from the data

   .... 

Recognition:

Compare your digit with the mean images:

d(0,3) = ||  -  || , d(1,3) = ||  -  ||

d(3,3) should be the minimum value.

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier?

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier? **90%.**

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier? **90%.**
- Test error of nearest neighbor: **3.09%.**

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier? **90%.**
- Test error of nearest neighbor: **3.09%.**

Examples of errors:

Performance on MNIST

Training set of 60,000 points.

- What is the error rate on training points? **Zero.**
  In general, **training error** is an overly optimistic predictor of future performance.
- A better gauge: separate test set of 10,000 points.
  Test error = fraction of test points incorrectly classified.
- What test error would we expect for a random classifier? **90%.**
- Test error of nearest neighbor: **3.09%.**

Examples of errors:

Query

NN

Properties of NN: (1) Can model arbitrarily complex functions
(2) Unbounded in size

Quick review of conditional probability

Formula for conditional probability: for any events A, B,

$$\Pr(A|B) = \frac{\Pr(A \cap B)}{\Pr(B)}.$$

Quick review of conditional probability

Formula for conditional probability: for any events A, B,

$$\Pr(A|B) = \frac{\Pr(A \cap B)}{\Pr(B)}.$$

Applied twice, this yields Bayes' rule:

$$\Pr(H|E) \; = \; \frac{\Pr(E|H)}{\Pr(E)}\Pr(H).$$

Quick review of conditional probability

Formula for conditional probability: for any events A, B,

$$\Pr(A|B) = \frac{\Pr(A \cap B)}{\Pr(B)}.$$

Applied twice, this yields Bayes' rule:

$$\Pr(H|E) \;=\; \frac{\Pr(E|H)}{\Pr(E)}\Pr(H).$$

Total Probability: Suppose events $A_1, \ldots, A_k$ are disjoint events, one of which must occur. Then for any other event E,

$$\begin{aligned}\Pr(E) &= \Pr(E, A_1) + \Pr(E, A_2) + \cdots + \Pr(E, A_k) \\ &= \Pr(E|A_1)\Pr(A_1) + \Pr(E|A_2)\Pr(A_2) + \cdots + \Pr(E|A_k)\Pr(A_k)\end{aligned}$$

Generative models

Generating a point (x, y) in two steps:

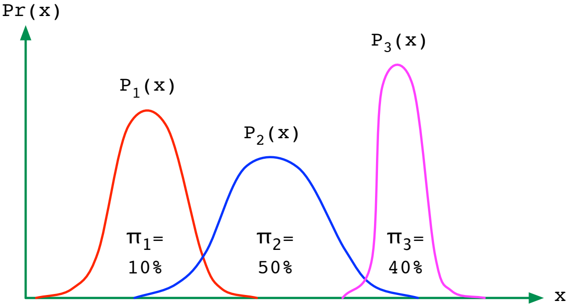
1. First choose y
2. Then choose x given y

Generative models

Generating a point (x, y) in two steps:

1. First choose y
2. Then choose x given y

Example:
$\mathcal{X} = \mathbb{R}$
$\mathcal{Y} = \{1, 2, 3\}$

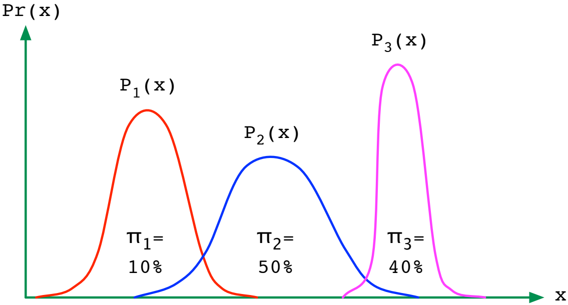


Generative models

Generating a point (x, y) in two steps:

1. First choose y
2. Then choose x given y

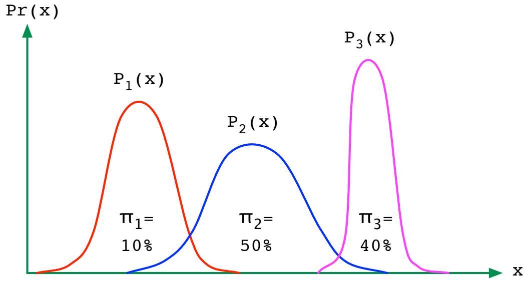
Example:
$\mathcal{X} = \mathbb{R}$
$\mathcal{Y} = \{1, 2, 3\}$



The overall density is a mixture of the individual densities,

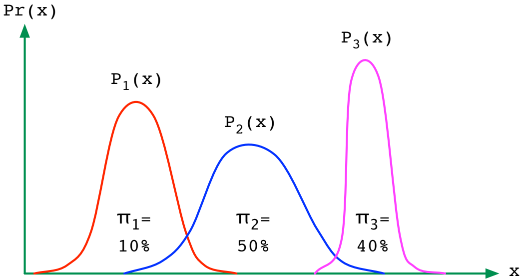
$$\Pr(x) = \pi_1 P_1(x) + \cdots + \pi_k P_k(x).$$

The Bayes-optimal prediction



Labels $\mathcal{Y} = \{1, 2, \ldots, k\}$, density $\Pr(x) = \pi_1 P_1(x) + \cdots + \pi_k P_k(x)$.

The Bayes-optimal prediction

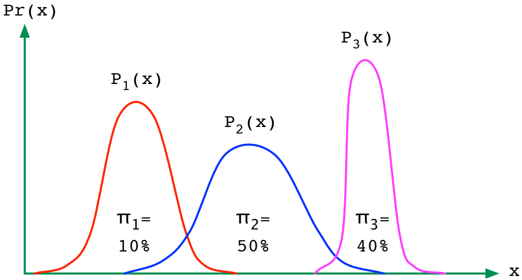


Labels $\mathcal{Y} = \{1, 2, \ldots, k\}$, density $\Pr(x) = \pi_1 P_1(x) + \cdots + \pi_k P_k(x)$.

For any $x \in \mathcal{X}$ and any label j,

$$\Pr(y = j|x) = \frac{\Pr(y = j)\Pr(x|y = j)}{\Pr(x)} = \frac{\pi_j P_j(x)}{\sum_{i=1}^{k} \pi_i P_i(x)}$$

The Bayes-optimal prediction



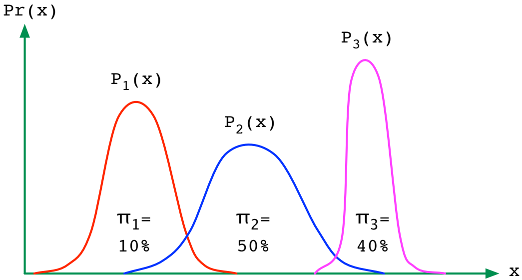
Labels $\mathcal{Y} = \{1, 2, \ldots, k\}$, density $\mathrm{Pr}(x) = \pi_1 P_1(x) + \cdots + \pi_k P_k(x)$.

For any $x \in \mathcal{X}$ and any label j,

$$\mathrm{Pr}(y = j|x) = \frac{\mathrm{Pr}(y = j)\mathrm{Pr}(x|y = j)}{\mathrm{Pr}(x)} = \frac{\pi_j P_j(x)}{\sum_{i=1}^{k} \pi_i P_i(x)}$$

Bayes-optimal (minimum-error) prediction: $h^*(x) = \arg\max_j \pi_j P_j(x)$.

The Bayes-optimal prediction



Labels $\mathcal{Y} = \{1, 2, \ldots, k\}$, density $\mathrm{Pr}(x) = \pi_1 P_1(x) + \cdots + \pi_k P_k(x)$.

For any $x \in \mathcal{X}$ and any label j,

$$\mathrm{Pr}(y = j|x) = \frac{\mathrm{Pr}(y = j)\mathrm{Pr}(x|y = j)}{\mathrm{Pr}(x)} = \frac{\pi_j P_j(x)}{\sum_{i=1}^{k} \pi_i P_i(x)}$$

Bayes-optimal (minimum-error) prediction: $h^*(x) = \arg\max_j \pi_j P_j(x)$.

Estimating the π_j is easy. Estimating the P_j is hard.

Estimating class-conditional distributions

Estimating an arbitrary distribution in $\mathbb{R}^p$:

- Can be done, e.g. with kernel density estimation.
- But number of samples needed is exponential in p.

Estimating class-conditional distributions

Estimating an arbitrary distribution in $\mathbb{R}^p$:

- Can be done, e.g. with kernel density estimation.
- But number of samples needed is exponential in p.

Instead: approximate each P_j with a simple, parametric distribution.

Estimating class-conditional distributions

Estimating a distribution in $\mathbb{R}^p$:

Approximate each P_j with a simple, parametric distribution.

Some options:

- Product distributions.
  Assume coordinates are independent: naive Bayes.
- Multivariate Gaussians.
  Linear and quadratic discriminant analysis.
- More general graphical models.

Naive Bayes

Labels $\mathcal{Y} = \{1, 2, \ldots, k\}$, density $\Pr(x) = \pi_1 P_1(x) + \cdots + \pi_k P_k(x)$.

Binarized MNIST:

- $k = 10$ classes
- $\mathcal{X} = \{0, 1\}^{784}$

P_1 is the pdf of the digit 1
P_2 is the pdf of the digit 2
:
:
P_{10} is the pdf of the digit 0

Naive Bayes

Labels $\mathcal{Y} = \{1, 2, \ldots, k\}$, density $\Pr(x) = \pi_1 P_1(x) + \cdots + \pi_k P_k(x)$.

Binarized MNIST:

- $k = 10$ classes
- $\mathcal{X} = \{0, 1\}^{784}$

Assume that **within each class**, the individual pixel values are independent:

$$P_j(x) = P_{j1}(x_1) \cdot P_{j2}(x_2) \cdots P_{j,784}(x_{784}).$$

Naive Bayes

Labels $\mathcal{Y} = \{1, 2, \ldots, k\}$, density $\Pr(x) = \pi_1 P_1(x) + \cdots + \pi_k P_k(x)$.

Binarized MNIST:

- $k = 10$ classes
- $\mathcal{X} = \{0, 1\}^{784}$

Each pixel value takes the value of 1 or 0

Assume that **within each class**, the individual pixel values are independent:

$$P_j(x) = P_{j1}(x_1) \cdot P_{j2}(x_2) \cdots P_{j,784}(x_{784}).$$

Each P_{ji} is a coin flip (Bernoulli): trivial to estimate!

Smoothed estimate of coin bias

Pick a class j and a pixel i. We need to estimate

$$p_{ji} = \Pr(x_i = 1 | y = j).$$

Smoothed estimate of coin bias

Pick a class j and a pixel i. We need to estimate

$$p_{ji} = \Pr(x_i = 1 | y = j).$$

Out of a training set of size n,

$$\begin{aligned} n_j &= \# \text{ of instances of class } j \\ n_{ji} &= \# \text{ of instances of class } j \text{ with } x_i = 1 \end{aligned}$$

Then the maximum-likelihood estimate of p_{ji} is

$$\widehat{p_{ji}} = n_{ji} / n_j.$$

Smoothed estimate of coin bias

Pick a class j and a pixel i. We need to estimate

$$p_{ji} = \Pr(x_i = 1 | y = j).$$

Out of a training set of size n,

$$n_j = \# \text{ of instances of class } j$$
$$n_{ji} = \# \text{ of instances of class } j \text{ with } x_i = 1$$

Then the maximum-likelihood estimate of p_{ji} is

$$\widehat{p}_{ji} = n_{ji}/n_j.$$

This causes problems if $n_{ji} = 0$.

Smoothed estimate of coin bias

Pick a class j and a pixel i. We need to estimate

$$p_{ji} = \Pr(x_i = 1 | y = j).$$

Out of a training set of size n,

$$\begin{aligned} n_j &= \#\text{ of instances of class } j \\ n_{ji} &= \#\text{ of instances of class } j \text{ with } x_i = 1 \end{aligned}$$

Then the maximum-likelihood estimate of p_{ji} is

$$\widehat{p}_{ji} = n_{ji}/n_j.$$

This causes problems if $n_{ji} = 0$. Instead, use "Laplace smoothing":

$$\widehat{p}_{ji} = \frac{n_{ji} + 1}{n_j + 2}.$$

Form of the classifier

Data space $\mathcal{X} = \{0,1\}^p$, label space $\mathcal{Y} = \{1,\ldots,k\}$. Estimate:

- $\{\pi_j : 1 \leq j \leq k\}$
- $\{p_{ji} : 1 \leq j \leq k, 1 \leq i \leq p\}$

Then classify point x as

$$\arg\max_j \quad \pi_j \prod_{i=1}^{p} p_{ji}^{x_i} (1 - p_{ji})^{1-x_i}.$$

Form of the classifier

Data space $\mathcal{X} = \{0,1\}^p$, label space $\mathcal{Y} = \{1,\ldots,k\}$. Estimate:

- $\{\pi_j : 1 \leq j \leq k\}$
- $\{p_{ji} : 1 \leq j \leq k, 1 \leq i \leq p\}$

Then classify point x as

$$\arg\max_j \quad \pi_j \prod_{i=1}^{p} p_{ji}^{x_i} (1 - p_{ji})^{1-x_i}.$$

To avoid underflow: take the log:

$$\arg\max_j \quad \underbrace{\log \pi_j + \sum_{i=1}^{p} (x_i \log p_{ji} + (1-x_i)\log(1-p_{ji}))}$$

Example: MNIST

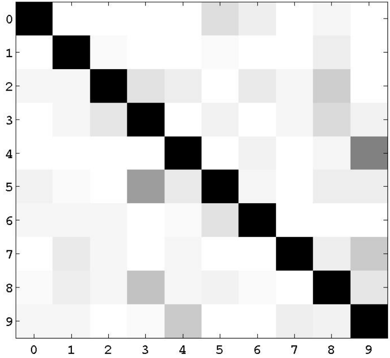
Result of training: mean vectors for each class.

Example: MNIST

Result of training: mean vectors for each class.

Test error rate: 15.54%.

Visualization of the "confusion matrix" $\longrightarrow$



Other types of data

How would you handle data:

- Whose features take on more than two discrete values (such as ten possible colors)?
- Whose features are real-valued?
- Whose features are positive integers?
- Whose features are mixed: some real, some Boolean, etc?

Other types of data

How would you handle data:

- Whose features take on more than two discrete values (such as ten possible colors)?
- Whose features are real-valued?
- Whose features are positive integers?
- Whose features are mixed: some real, some Boolean, etc?

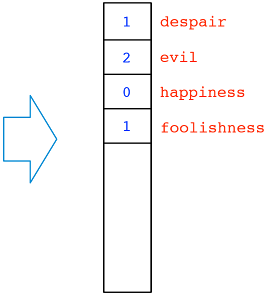
How would you handle "missing data": situations in which data points occasionally (or regularly) have missing entries?

- At train time: **???**
- At test time: **???**

Handling text data

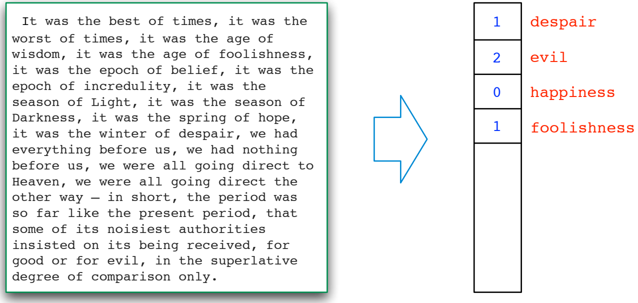
Bag-of-words: vectorial representation of text documents.

It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way – in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.



Handling text data

Bag-of-words: vectorial representation of text documents.



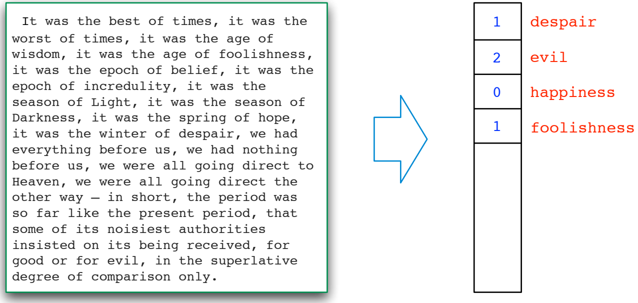
- Fix V = some vocabulary.
- Treat each document as a vector of length $|V|$:

$$x = (x_1, x_2, \ldots, x_{|V|}),$$

where x_i = # of times the ith word appears in the document.

Handling text data

Bag-of-words: vectorial representation of text documents.



- Fix V = some vocabulary.
- Treat each document as a vector of length $|V|$:

$$x = (x_1, x_2, \ldots, x_{|V|}),$$

where x_i = # of times the ith word appears in the document.

A standard distribution over such document-vectors x: the **multinomial**.

Multinomial naive Bayes

Multinomial distribution over a vocabulary V:

$$p = (p_1, \ldots, p_{|V|}), \quad \text{such that} \quad p_i \geq 0 \text{ and } \sum_i p_i = 1$$

Document $x = (x_1, \ldots, x_{|V|})$ has probability $\propto p_1^{x_1} p_2^{x_2} \cdots p_{|V|}^{x_{|V|}}$.

Multinomial naive Bayes

Multinomial distribution over a vocabulary V:

$$p = (p_1, \ldots, p_{|V|}), \quad \text{such that} \quad p_i \geq 0 \text{ and } \sum_i p_i = 1$$

Document $x = (x_1, \ldots, x_{|V|})$ has probability $\propto p_1^{x_1} p_2^{x_2} \cdots p_{|V|}^{x_{|V|}}$.

For naive Bayes: one multinomial distribution per class.

- Class probabilities $\pi_1, \ldots, \pi_k$
- Multinomials $p^{(1)} = (p_{11}, \ldots, p_{1|V|}), \ldots, p^{(k)} = (p_{k1}, \ldots, p_{k|V|})$

Multinomial naive Bayes

Multinomial distribution over a vocabulary V:

$$p = (p_1, \ldots, p_{|V|}), \quad \text{such that } \; p_i \geq 0 \text{ and } \sum_i p_i = 1$$

Document $x = (x_1, \ldots, x_{|V|})$ has probability $\propto p_1^{x_1} p_2^{x_2} \cdots p_{|V|}^{x_{|V|}}$.

For naive Bayes: one multinomial distribution per class.

- Class probabilities $\pi_1, \ldots, \pi_k$
- Multinomials $p^{(1)} = (p_{11}, \ldots, p_{1|V|}), \ldots, p^{(k)} = (p_{k1}, \ldots, p_{k|V|})$

Classify document x as

$$\arg\max_j \quad \pi_j \prod_{i=1}^{|V|} p_{ji}^{x_i}.$$

(As always, take log to avoid underflow: linear classifier.)

Improving performance of multinomial naive Bayes

A variety of heuristics that are standard in text retrieval, such as:

1. Compensating for burstiness.
   Problem: Once a word has appeared in a document, it has a much higher chance of appearing again.

Improving performance of multinomial naive Bayes

A variety of heuristics that are standard in text retrieval, such as:

1. Compensating for burstiness.
   Problem: Once a word has appeared in a document, it has a much higher chance of appearing again.

   Solution: Instead of the number of occurrences f of a word, use $\log(1 + f)$.

Improving performance of multinomial naive Bayes

A variety of heuristics that are standard in text retrieval, such as:

1. Compensating for burstiness.
   Problem: Once a word has appeared in a document, it has a much higher chance of appearing again.

   Solution: Instead of the number of occurrences f of a word, use $\log(1 + f)$.

2. Downweighting common words.
   Problem: Common words can have a unduly large influence on classification.

Improving performance of multinomial naive Bayes

A variety of heuristics that are standard in text retrieval, such as:

1. Compensating for burstiness.
   Problem: Once a word has appeared in a document, it has a much higher chance of appearing again.

   Solution: Instead of the number of occurrences f of a word, use $\log(1 + f)$.

2. Downweighting common words.
   Problem: Common words can have a unduly large influence on classification.

   Solution: Weight each word w by **inverse document frequency**:

$$\log \frac{\#\ \text{docs}}{\#(\text{docs containing } w)}$$