COMS 4721: Machine Learning for Data Science
Lecture 7, 2/7/2017

Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

Columbia University

CLASSIFICATION

Terminology and Notation

Input: As with regression, in a *classification problem* we start with measurements $x_1, \ldots, x_n$ in an input space $\mathcal{X}$. (Again think $\mathcal{X} = \mathbb{R}^d$) input space

The y associated with an x is from discrete output space. (K classes)

Output: The *discrete* output space $\mathcal{Y}$ is composed of K possible *classes*:

- $\mathcal{Y} = \{-1, +1\}$ or $\{0, 1\}$ is called binary classification.
- $\mathcal{Y} = \{1, \ldots, K\}$ is called multiclass classification

Instead of a real-valued response, classification assigns x to a category.

- Regression: For pair (x, y), y is the response of x.
- Classification: For pair (x, y), y is the class of x.

Classification problem

Defining a classifier

Classification uses a function f (called a *classifier*) to map input x to class y.

$$y = f(x) \ : \ f \text{ takes in } x \in \mathcal{X} \text{ and declares its class to be } y \in \mathcal{Y}$$

As with regression, the problem is two-fold:

- Define the classifier f and its parameters. learn those parameters using labelled training data.
- Learn the classification rule using a training set of "labeled data."

NEAREST NEIGHBOR CLASSIFIERS

(simplest classifier one could come up with)

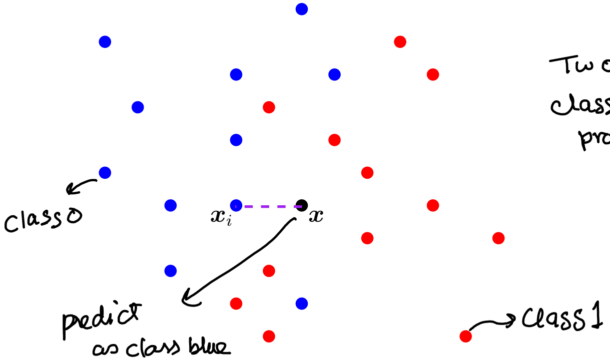
NEAREST NEIGHBOR (NN) CLASSIFIER

input in $\mathbb{R}^2$

Given data $(x_1, y_1), \ldots, (x_n, y_n)$, construct classifier $\hat{f}(x) \to y$ as follows:

For an input x not in the training data,

1. Let x_i be the point among $x_1, x_2, \ldots, x_n$ that is "closest" to x.
2. Return its label y_i.



Two class classification problem.

DISTANCES

Question: How should we measure distance between points?

The default distance for data in $\mathbb{R}^d$ is the Euclidean one:

$$\|u - v\|_2 = \left(\sum_{i=1}^{d}(u_i - v_i)^2\right)^{\frac{1}{2}} \quad \text{(line-of-sight distance)}$$

But there are other options that may sometimes be better:

- ℓ_p for $p \in [1, \infty]$: $\|u - v\|_p = \left(\sum_{i=1}^{d}|u_i - v_i|^p\right)^{\frac{1}{p}}$.
- Edit distance (for strings): How many add/delete/substitutions are required to transform one string (input) to the other. string in the training data.
- Correlation distance (for signal): Measures how correlated two vectors are for signal detection.

And then find the string in training set that requires few strings to match my input string. That would be the nearest neighbour

EXAMPLE: OCR WITH NN CLASSIFIER

Using Euclidean distance

- **Handwritten digits data**: grayscale 28×28 images, treated as vectors in $\mathbb{R}^{784}$, with labels indicating the digit they represent.

- Split into training set $\mathcal{S}$ (60K points) and testing set $\mathcal{T}$ (10K points).
- **Training error**: $\text{err}(\hat{f}, \mathcal{S}) = 0$ $\leftarrow$ declare its class to be its own class!
  Test error: $\text{err}(\hat{f}, \mathcal{T}) = 0.\underline{0309}$ $\leftarrow$ using ℓ_2 distance
  very less
- Examples of <u>mistakes:</u> (left) test point, (right) nearest neighbor in $\mathcal{S}$:

labelled as 8

- **Observation**: First mistake might have been avoided by looking at three nearest neighbors (whose labels are '8', '2', '2') …

Instead of picking the closest label.

test point   three nearest neighbors

closest   next two closest

we would have got it right because two of training sets would have voted to label this a two & one would have voted to label it an eight.

If we wanted to predict the label of one of labelled data points. We would predict label in NN classifier perfectly. Because any point in our training set is closest to itself. Therefore, it would pick its own label.

So the training error in a NN classifier is always zero.

k-NEAREST NEIGHBORS CLASSIFIER (extension of nearest neighbour classifier)

label data points

Given data $(x_1, y_1), \ldots, (x_n, y_n)$, construct the k-NN classifier as follows:

For a new input x,

1. Return the k points closest to x, indexed as $x_{i_1}, \ldots, x_{i_k}$.
2. Return the majority-vote of $y_{i_1}, y_{i_2}, \ldots, y_{i_k}$. [labelled examples]

(Break ties in both steps arbitrarily.)

Example: OCR with k-NN classifier

k	1	3	5	7	9
$\text{err}(\hat{f}_k, T)$	0.0309	0.0295	0.0312	0.0306	0.0341

So this an example of where we might want to do a sweep of values and then pick the best performing value.

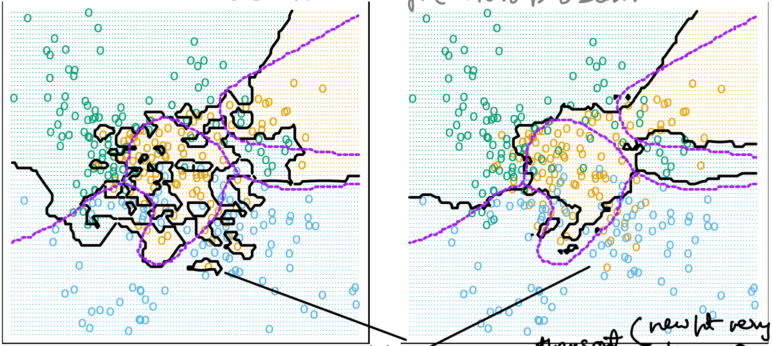
EFFECT OF k

So we can think of k as a smoothing parameter. As k gets bigger, we smooth out our decisions.

In general:

- Smaller $k \Rightarrow$ smaller training error.
- Larger $k \Rightarrow$ predictions are more "stable" due to voting.

Three class classification problem



1-NN

looking at 15 smoothens out

15-NN

(new pt very close to orange but most of them are blue so blue class)

Purple dotted lines : Can ignore for now.

Black solid lines : k-NN's decision boundaries.

Decision boundary for NN classifier:

That means, we look at every single potential new point and say what would I classify this new point and color code it.

We get a blue region which says that any point in this blue region is going to have its nearest neighbor be a blue class observation. And so the shaded blue region corresponds to what would be define to be class blue.

So the decision boundary is the point at which this changes

For ex: At this point, it is equally close to the blue class and green class.

And so we can see that we get a very fragmented decision boundary with nearest neighbor classifier. Essentially every point is staking out its own region.

If we look at 15 nearest neighbours instead we get a much smoother decision boundary.

 (Looking at other classifiers. Starting by thinking statistically)

How do we measure the quality of a classifier?

For any classifier we care about two sides of the same coin:

- Prediction accuracy: $P(f(x) = y)$. (Probability that a classifier is going to predict the correct label for x.)
- Prediction error: $\mathrm{err}(f) = P(f(x) \neq y)$. (Probability that we get it wrong. And so we want to minimize that.)

To calculate these values, we assume there is a distribution $\mathcal{P}$ over the space of labeled examples generating the data

$$* \quad (x_i, y_i) \overset{iid}{\sim} \mathcal{P}, \qquad i = 1, \ldots, n.$$

We don't know what $\mathcal{P}$ is, but can still talk about it in abstract terms.

And what this means is that the labelled examples that are coming to us under our assumption are going to IID distributed according to some underlying ground truth distribution that nature provides but we don't necessarily get to see.

* So there is some joint distribution on both the covariance x and label y.

① Why do you mention covariance specifically?

② Which of these is the likelihood distribution?

③ Is this distribution conditional? $x|y$?

④ How do you relate both of these distributions with Bayes rule?

When is there any hope for finding an accurate classifier?

Key assumption: Data $(x_1, y_1), \ldots, (x_n, y_n)$ are i.i.d. random labeled examples with distribution $\mathcal{P}$.

This assumption allows us to say that the past should look like the future.

comes to us according to the same exact distribution. ← new (unlabeled) example from P

past labeled examples from P → learning algorithm → learned predictor → predicted label

Assumes like a basic implicit assumption that is necessary.
What machine learning models don't make this assumption?

Regression makes similar assumptions.

We're simply going to make this assumption that our data is coming to us IID What we learn based on the past will generalize well for the future.
Regression is also making the same assumption.

Key assumption: By assuming our labelled data in our training set are coming to us IID according to some ground truth distribution and also that all future data that's going to come to us is generated according to that same distribution with the exception that with our ~~training~~ testing set, we only get to observe x.

We can take advantage of this statistical regularities there to assume that things that perform well on our training set are going to generalize well to our testing set.

which ones are you talking about?

BAYES CLASSIFIERS

Let's use this IID assumption to motivate something called the bayes classifier.

OPTIMAL CLASSIFIERS

The theory here at a high level is only going to motivate why a bayes classifier might be the right thing to do. But we aren't going to then say, that actually fund the optimal classifier.

Can we talk about what an “optimal” classifier looks like?

Assume our data has come to us IID according to some distribution where we're getting

Assume that $(X, Y) \overset{iid}{\sim} \mathcal{P}$. (Again, we don't know $\mathcal{P}$) the labelled pairs simultaneously. *

Some probability equalities with $\mathcal{P}$:

1. The expectation of an indicator of an event is the probability of the event, e.g.,

binary classifiers ** ***

$$\mathbb{E}_P[\mathbb{1}(Y = 1)] = P(Y = 1), \quad \leftarrow \mathbb{1}(\cdot) = 0 \text{ or } 1 \text{ depending if } \cdot \text{ is true}$$

The second principle is about tower property of conditional expectation.

2. Conditional expectations can be random variables, and their expectations remove the randomness,

$$C = \mathbb{E}[A \mid B] : \quad A \text{ and } B \text{ are } \textit{both} \text{ random, so } C \text{ is random}$$

$$\leftarrow \mathbb{E}[C] = \mathbb{E}[\mathbb{E}[A \mid B]] = \mathbb{E}[A] \quad \text{“tower property” of expectation}$$ ooo

This expectation of this random variable gets rid of all the randomness. Expectation of C where this is integrating over b. is equal to the expectation of the conditional expectation of a given b. oo

why?

* So there's a joint distribution on both covariates and labels.

↳ How does relate to generative models?
like you are assuming a normal over x & y?

** The expectation under p of our label being equal to 1 is simply the probability that the label equals 1 according to this distribution. So we've integrated our x and we get probability on being to 1 or 0 or some otherwise.

*** The expectation of indicator of the event is equal to the probability of the event under the distribution you're using to calculate the expectation.

- This is calculating the expectation of a. So we integrate out all uncertainity of a, condition on a particular b. So for a specific value of b we get the expectation of a and call that c. However, because b is now a random variable, this expectation also becomes a random variable. So we randomly generate a b plug it in here, get our expectation of a. But because b is random, this expectation is also random.

oo That's simply equal to the expectation of a where b now has been integrated out.

ooo Expectations of conditional expectations remove the conditioning essentially.

Optimal classifiers (Use the previous two properties to calculate what an optimal classifier might look like.)

How did you apply the tower property here? Why?

For any classifier $f: \mathcal{X} \to \mathcal{Y}$, its prediction error is

$$P(f(X) \neq Y) = \mathbb{E}[\mathbb{1}(f(X) \neq Y)] = \mathbb{E}[\underbrace{\mathbb{E}[\mathbb{1}(f(X) \neq Y) \mid X]}_{\text{a random variable}}] \qquad (\dagger)$$

That minimizes the probability of making a mistake. According to the assumption that our data is IID from some underlying distribution.

[X and Y are generated from randomly from some underlined distributions. They're independent from any other labelled pairs.]

Now, calculate that probability integrating over the distribution P. So from the previous slide, we can say that the probability that our classifier will make a mistake is equal to the expectation of the indicator that our classifier makes a mistake. This expectation uses the true underlying distribution P to calculate it.

Tower property of conditional expectation. Says that we can 1st calculate the conditional expectation of this indicator given the value of x.

*

* So let's 1st pretend like we know the value of x, and then say for that specific value, what is the expectation of that indicator that I don't predict the label correctly. And then take the expectation of that.
Why is this a random variable? So a classifier is not random. Given x, f(x) is not a random variable. So a classifier is not random.

So our classifier will always predict the same label for particular x.

However, we assumed that the underlying distribution does not have a deterministic distribution on y given the x.

So for example the same email might 95% of the time truly be a spam email. But 5% of the time, that exact same email might not be spam.

So f is deterministic, but P is not? Then we can never achieve 100% accuracy right?

So in this sense, y is what's random ∴ ind t t's why this a random variable. And now we can take the expectation of that, we get rid of the impact of x.

Didn't understand the last line.

For any classifier $f : \mathcal{X} \to \mathcal{Y}$, its prediction error is

$$P(f(X) \neq Y) = \mathbb{E}[\mathbb{1}(f(X) \neq Y)] = \mathbb{E}[\underbrace{\mathbb{E}[\mathbb{1}(f(X) \neq Y) \,|\, X]}_{\text{a random variable}}] \qquad (\dagger)$$

For each $x \in \mathcal{X}$, Evaluating conditional expectation at a specific value x

$$\mathbb{E}[\mathbb{1}(f(X) \neq Y) \,|\, X = x] = \sum_{y \in \mathcal{Y}} P(Y = y \,|\, X = x) \cdot \mathbb{1}(f(x) \neq y), \qquad (\ddagger)$$

we are taking the expectation of the indicator that our classifier gets it wrong at this particular x. Using the conditional distribution of the probability of y given the x that we're evaluating at. So x is fixed but y is random.

Meaning?

Our goal is for every point x, pointwise we want to minimize this function.

For any classifier $f : \mathcal{X} \to \mathcal{Y}$, its prediction error is

$$P(f(X) \neq Y) = \mathbb{E}[\mathbb{1}(f(X) \neq Y)] = \mathbb{E}[\underbrace{\mathbb{E}[\mathbb{1}(f(X) \neq Y)\,|X]}_{\text{a random variable}}] \qquad (\dagger)$$

So in order to minimize it, what it means is that our classifier is going to assign one of the k possible labels to every point x.

For each $x \in \mathcal{X}$, And so this is going to be equal to 0 only for 1 value of y. *

$$\mathbb{E}[\mathbb{1}(f(X) \neq Y)\,|\,X = x] = \sum_{y \in \mathcal{Y}} P(Y = y\,|\,X = x) \cdot \underline{\mathbb{1}(f(x) \neq y)}, \qquad (\ddagger)$$

what is this doing here?

The above quantity ($\ddagger$) is minimized for this particular $x \in \mathcal{X}$ when

$$f(x) = \arg\max_{y \in \mathcal{Y}} P(Y = y|X = x). \qquad (\star)$$

**

The classifier f with property ($\star$) for all $x \in \mathcal{X}$ is called the *Bayes classifier*, and <u>it has the smallest prediction error ($\dagger$) among *all classifiers*.</u> by making prediction acc. to **

how to show this?

* So what this is really doing is saying for the label that we assign to x. Sum the probabilities of all other labels other than that. In other words, sum the probability that we get it wrong.

So if we want to minimize the equality, we can do so by assigning the label x to the most probable label according to nature. And we're summing up the K-1 smallest probabilites acc. to this distribution.

** Bayes classifier: For a particular input x, predict the label to be the most probable label conditioned on x, according to some true underlying distribution given to us from nature.

<u>Problem:</u> If we get our labelled data as IID from some underlying distribution from nature. And then for every single x, we predict the label to be the most probable label according to that distribution. Then we're gonna minimize our probability of making an error.

So of course, the problem is that we don't know what the underlying distribution is from nature, so we don't actually know what this " is. So we need to <u>approximate it</u>. ← how do you approximate it?

And the approximation is going to therefore not allow say that we have the optimal classifier.

The Bayes classifier

labelled data is iid from some underlying distribution $\mathcal{P}$.

Under the assumption $(X, Y) \overset{iid}{\sim} \mathcal{P}$, the optimal classifier is

This a joint distribution how do we marginalise it to get a conditional distribution?

③ should predict for a particular X, the label data that is most probable given that X.

$$f^\star(x) := \arg\max_{y \in \mathcal{Y}} P(Y = y | X = x).$$

we don't this P right?

From Bayes rule we equivalently have *

The problem is that we don't have this probability distribution so we somehow need to approximate it. And let's do this by Bayes rule.

* to maximise ① & ② to maximise ③

$$f^\star(x) = \arg\max_{y \in \mathcal{Y}} \underbrace{P(Y = y)}_{\text{class prior } ①} \times \underbrace{P(X = x | Y = y)}_{② \text{ data likelihood} \,|\, \text{class}}.$$

- $P(Y = y)$ is called the *class prior*. (a prior prevalence of any of the classes)
- $P(X = x | Y = y)$ is called the *class conditional distribution* of X. **
- In practice we don't know either of these, so we approximate them. (distributions)

Aside: If X is a continuous-valued random variable, replace $P(X = x|Y = y)$ with *class conditional density* $p(x|Y = y)$.

** So given that our vector x comes from the class y, we have class specific distribution on x.

From bayesrule, we equivantenly have that:

$$\text{Posteriorprob. of the label of } x \text{ given the value of } x \propto \frac{\text{prior prob. of that label } \times \text{ likelihood of } x \text{ given the label}}{\text{Normalizing constant that doesn't depend on } y. \text{ (probability of } x\text{, where } y \text{ is marginalised out)}}$$

Example: Gaussian class conditional densities

we get a value from $\mathbb{R}$ randomly and we now want to label it to be either class 0 or 1.

Suppose $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \{0, 1\}$, and the distribution $\mathcal{P}$ of (X, Y) is as follows.

Assume that the nature generates labelled pairs as follows:

Binary classification problem.

- **Class prior**: $P(Y = y) = \pi_y, \quad y \in \{0, 1\}$. $\pi_1 \to$ prob. to class 1, $\pi_0 \to$ " " " 0

  (picks a class for label according to a probability)

- **Class conditional density** for class $y \in \{0, 1\}$: $p_y(x) = N(x|\mu_y, \sigma_y^2)$. univariate Gaussian

  Given our data x is in class 1, we generate x from a Gaussian with mean and variance

- **Bayes classifier**: specific to class 1.

predict the label x to be argmax

$$
\begin{aligned}
f^\star(x) &= \underset{y \in \{0,1\}}{\operatorname{argmax}} \; \underbrace{p(X = x|Y = y)}_{\text{likelihood of x given label Y}} \underbrace{P(Y = y)}_{\text{prior of the label}} \\
&= \begin{cases} 1 & \text{if } \frac{\pi_1}{\sigma_1} \exp\left[-\frac{(x - \mu_1)^2}{2\sigma_1^2}\right] > \frac{\pi_0}{\sigma_0} \exp\left[-\frac{(x - \mu_0)^2}{2\sigma_0^2}\right] \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

we assume some distribution on both x & y. And we define these distributions & then we want to predict based on the conditional distribution of y given x, where we use bayes rule to switch that.

more probable aposteri the label is 1.

This type of classifier is called a *generative* model.

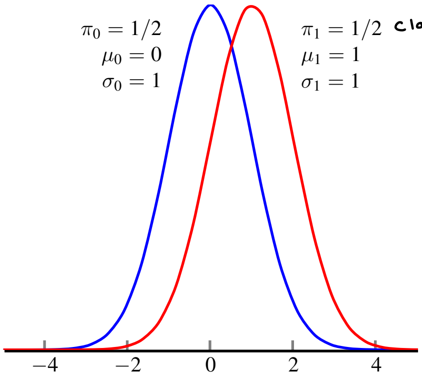
This type of classifier is called generative model because we have distributions on both X & Y.

- **Generative model**: Model x and y with distributions.
- **Discriminative model**: Plug x into a distribution on y (used thus far).

  (ones we discussed up until now.)

  ↳ we want to predict some response/class y conditioned on an input X, where we don't make any distribution assumptions on X. → examples of this.

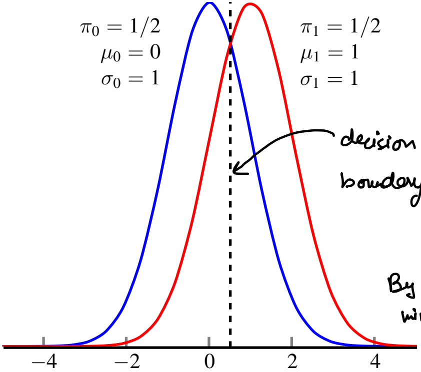
EXAMPLE: GAUSSIAN CLASS CONDITIONAL DENSITIES



class prevalence 50:50

$1/2$ of x's from $N(0, 1) \rightarrow y = 0$
$1/2$ of x's from $N(1, 1) \rightarrow y = 1$

EXAMPLE: GAUSSIAN CLASS CONDITIONAL DENSITIES



decision boundary

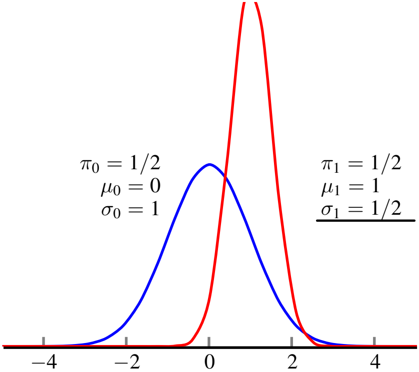
$1/2$ of x's from $N(0,1) \to y = 0$
$1/2$ of x's from $N(1,1) \to y = 1$

Bayes classifier:

$$f^\star(x) = \begin{cases} 1 & \text{if } x > 1/2; \\ 0 & \text{otherwise.} \end{cases}$$

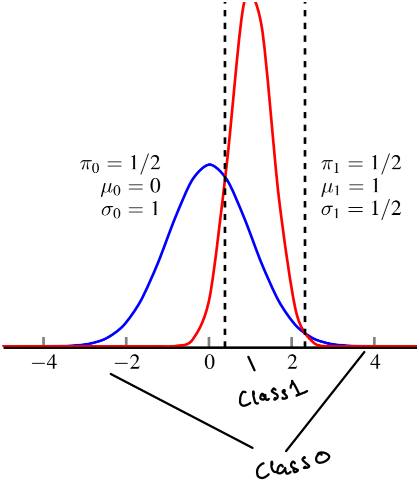
By making this decision boundary, we're simply minimizing the probability of getting it wrong according to these 2 probability distributions

Example: Gaussian class conditional densities



$1/2$ of x's from $\mathcal{N}(0, 1) \rightarrow y = 0$
$1/2$ of x's from $\mathcal{N}(1, 1/4) \rightarrow y = 1$

Example: Gaussian class conditional densities



$1/2$ of x's from $\mathcal{N}(0, 1) \rightarrow y = 0$
$1/2$ of x's from $\mathcal{N}(1, 1/4) \rightarrow y = 1$

Bayes classifier:

$$f^\star(x) = \begin{cases} 1 & \text{if } x \in [0.38, 2.29]; \\ 0 & \text{otherwise.} \end{cases}$$

In this case we get a decision region, we don't have a 1 decision boundary.

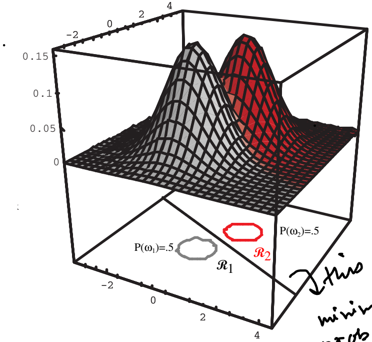
EXAMPLE: MULTIVARIATE GAUSSIANS (what it looks like in $\mathbb{R}^2$)

we are going to assume that the class conditional densities are both Gaussians with unique mean and variance.

Data: $\mathcal{X} = \mathbb{R}^2$, Label: $\mathcal{Y} = \{0, 1\}$

Class conditional densities are Gaussians in $\mathbb{R}^2$ with covariance Σ_0 and Σ_1.

each Gaussian has its own covariance

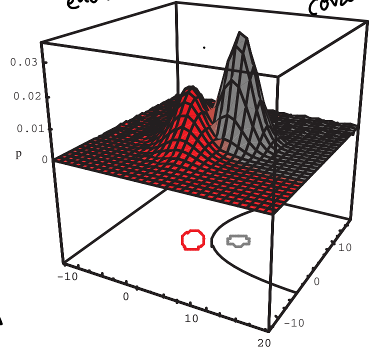


this line minimizes the prob of making an error.

$\Sigma_0 = \Sigma_1$

Bayes classifier:

linear separator

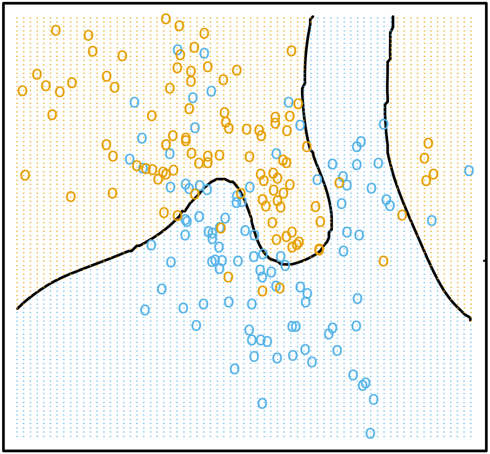


$\Sigma_0 \neq \Sigma_1$

Bayes classifier:

quadratic separator

Bayes classifier in general



depending upon
how complex the
class conditional
density is.

In general, the Bayes classifier may be rather complicated!
This one uses more than a single Gaussian for the class-conditional density.

PLUG-IN CLASSIFIERS

Assuming up to now that we know the class conditional density of each class in the bayes classifier. But in general we don't know the density, we have to pick it and approximate it.

Even though the base classifier has smallest prediction error of all classifier, its conditioned on our knowing class conditional densities & class apriori, which we don't know of course.

Bayes classifier

All we have are the labelled examples that we assume are generated IID according to this distribution from nature.

The Bayes classifier has the smallest prediction error of *all* classifiers.

Problem: We can't construct the Bayes classifier without knowing $\mathcal{P}$.

- What is $P(Y = y|X = x)$, or equiv., $P(X = x|Y = y)$ and $P(Y = y)$?
- All we have are labeled examples drawn from the distribution $\mathcal{P}$.

Plug-in classifiers

(If we use a beyes classifier in practice)

Use the available data to approximate $\underbrace{P(Y = y)}_{\text{class prior distribution}}$ and $\underbrace{P(X = x|Y = y)}_{\text{class condontional distribution}}$.

- Of course, the result may no longer give the best results among all the classifiers we can choose from (e.g., k-NN and those discussed later).

we are choosing things to approximate, we are going to pick a distribution that is not going to be the one nature uses.

The approximation means we no longer can claim optimality.

EXAMPLE: GAUSSIAN CLASS CONDITIONAL DENSITIES

assume each class has its own multivariate Gaussians from which the observations are generated (Assumption on how data is generated)

Here, $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{1, \ldots, K\}$. Estimate Bayes classifier via MLE:

we estimate our class priors and class conditionals using a multi-variate Gaussian.

why define like this?

- **Class priors**: The MLE estimate of π_y is $\hat{\pi}_y = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y_i = y)$.
- **Class conditional density**: Choose $p(x|Y = y) = N(x|\mu_y, \Sigma_y)$.
  The MLE estimate of (μ_y, Σ_y) is

MLE for class prior is just empirical distribution of labels in our dataset. Sum up no. of times we class

Now, do maximum likelihood to learn this classifier

$$\hat{\mu}_y = \frac{1}{n_y}\sum_{i=1}^{n} \mathbb{1}(y_i = y)x_i,$$

$$\hat{\Sigma}_y = \frac{1}{n_y}\sum_{i=1}^{n} \mathbb{1}(y_i = y)(x_i - \hat{\mu}_y)(x_i - \hat{\mu}_y)^T.$$

*

This is just the empirical mean and covariance of class y.

And now so we've approximated these 2 distributions, we then use the plugin classifiers.

- **Plug-in classifier**:

$$\hat{f}(x) \overset{**}{=} \arg\max_{y \in \mathcal{Y}} \underbrace{\hat{\pi}_y}_{\text{prior of coming from that class.}} \overbrace{|\hat{\Sigma}_y|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x - \hat{\mu}_y)^T \hat{\Sigma}_y^{-1}(x - \hat{\mu}_y)\right\}}^{\text{likelihood of the observ. given that it did come from that class.}}$$

* This something that we saw in the 1^{st} lecture, where we did MLE for multi-variate Gaussian. The maximum likelihood we are doing for the class that we're learning.

** We evaluate this function in the new x for each value of y and then we predict it to be the max.

EXAMPLE: SPAM FILTERING

Representing emails

- **Input**: x, a vector of word counts. For example, if index $\{j \to \text{“car”}\}$ $x(j) = 3$ means that the word “car” occurs three times in the email.
- **Output**: $\mathcal{Y} = \{-1, +1\}$. Map $\{\text{email} \to -1, \text{spam} \to +1\}$

Example dimensions

	george	you	your	hp	free	work	!	our	re	click	remove
spam	0	4	1	0	4	0	5	5	1	3	2
email	1	3	4	1	1	4	0	1	1	0	0

Using a Bayes classifier

$$f(x) = \underset{y \in \{-1,+1\}}{\operatorname{argmax}} \; p(x|Y = y)P(Y = y)$$

Predict the label of the email to be the maximum of the likelihood of the mail given the class times the prior of the class.

① NAIVE BAYES [That means we have to define these distributions]

class prior, use bionomial dist. Learn binomial parameter using MLE like before, amounts to calculating the empirical distribution of our labelled data.

② For class conditional distribution

We have to *define* $p(X = x|Y = y)$.

on word histogram X given that it comes from class y, we can't use a multivariate Gaussian anymore because it is not a resonable assumption. → why is it not?

Simplifying assumption

Naive Bayes is a Bayes classifier that makes the assumption

motivates ↑ dist. of word counts given class assignment is independent * which means the vector X of word counts can be written as a product over the likelihood of each individual count for a particular word given the class.

$$p(X = x|Y = y) = \prod_{j=1}^{d} p_j(x(j)|Y = y),$$

i.e., it treats the dimensions of X as *conditionally independent* given y.

This is Naive because we are breaking up any other correlations in our distribution, assuming that everything is independent given the class, which is a naive assumption, but also one that works fairly well.

In spam example

- Correlations between words is ignored.
- Can help make it easier to define the distribution.

↓ why

Why did you assume conditional independence instead of just independence.

* Meaning : That the probability of the no. of times that I see a particular word J given its class is independent to the probability that I see any other word.

ESTIMATION (of the parameters)

Class prior

of email being spam or not spam apriori

The distribution $P(Y = y)$ is again easy to estimate from the training data:

$$P(Y = y) = \frac{\#\text{observations in class } y}{\#\text{observations}}$$

For class conditional we make the naive assumption that the word counts are all independent of each other, so we can write it this way.

Class-conditional distributions

Now we pick a particular probability dist. for each specific word. So Poission, we can pick others.

For the spam model we define

$$P(X = x|Y = y) = \prod_j p_j(x(j)|Y = y) = \prod_j \text{Poisson}(x(j)|\lambda_j^{(y)}) *$$

We then approximate each $\lambda_j^{(y)}$ from the data. For example, the MLE is

show that MLE update for $\lambda_j^{(y)}$?

$$\lambda_j^{(y)} = \frac{\#\text{unique uses of word } j \text{ in observations from class } y}{\#\text{observations in class } y}$$

how ?

* This distribution says given that an email comes from class y, the no. of occurences of word j in that email is going to be Poisson distributed with parameter equal to λ_j for class y.

So each word now has a parameter associated with it for its Poission distribution that's also class dependent

can we have different distributions for each word?

So for the spam detection problem, to summarize the class conditional distribution on a spam email, what we would do is we would count for every particular word in our vocabulary, like the word car. We would count how many times does the word car appear in an email labeled spam.

So take all of our spam emails,count the total number of occurrences of the word car, that's the numerator. And then divide that by the number of emails in that class, so the spam class.

Do that for every single word in our vocabulary and do that for both classes, and we now have the parameter for the class specific distribution on the word histogram.