CE 395 Special Topics in Machine Learning

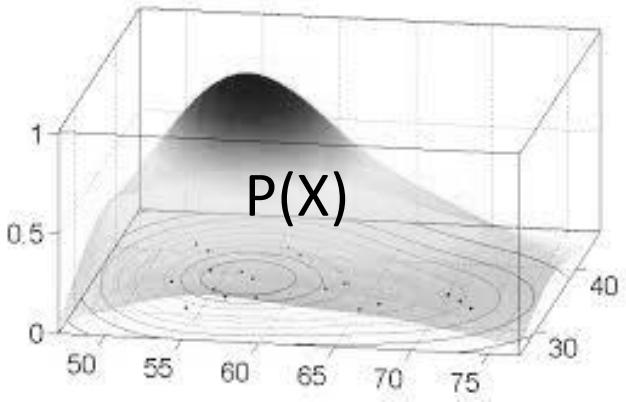
Assoc. Prof. Dr. Yuriy Mishchenko

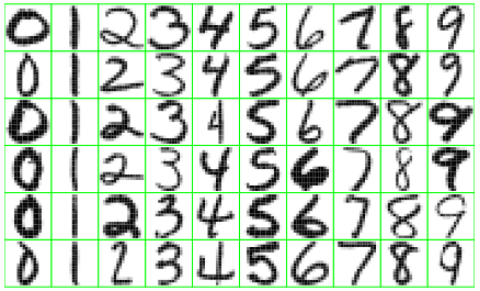
Fall 2017

STATISTICAL LEARNING

Statistical learning

P(X)=DATA





Statistical learning

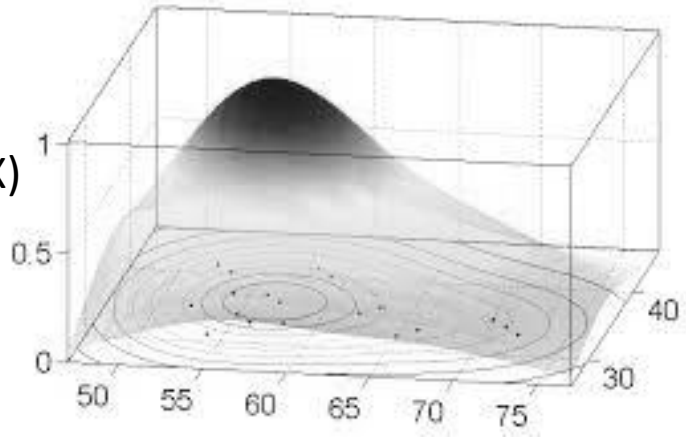
ML is statistical learning – learn P(X) through models $\pi(X)$: parametric such as $\pi(X;\theta)$ (linreg) or nonparametric $\pi(X;\{X_i\})$ (KNN, depends on data directly without parameters)

Statistical learning

- In SL we are primarily concerned with **data models**
- Data model is essentially P(X); it encodes everything there is to know about data as far as SL goes
- Data model can be generative in sense that different examples of data {Xi} can be generated from P(X) by random selection, P(X) $\rightarrow$ {Xi}

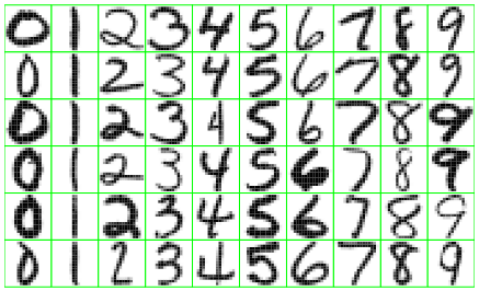
Data model:

Distribution P(X)



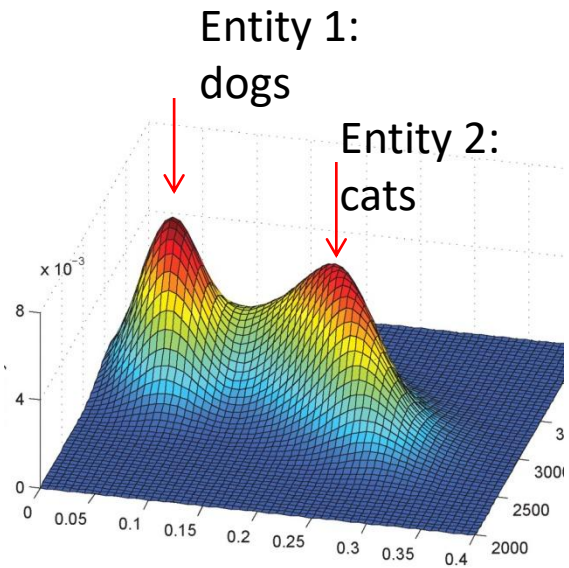
Domain $\mathcal{D}$

Example of data (random):



Statistical learning

P(X) encodes structure of data – clusters or groups generally imply **entities** in the data

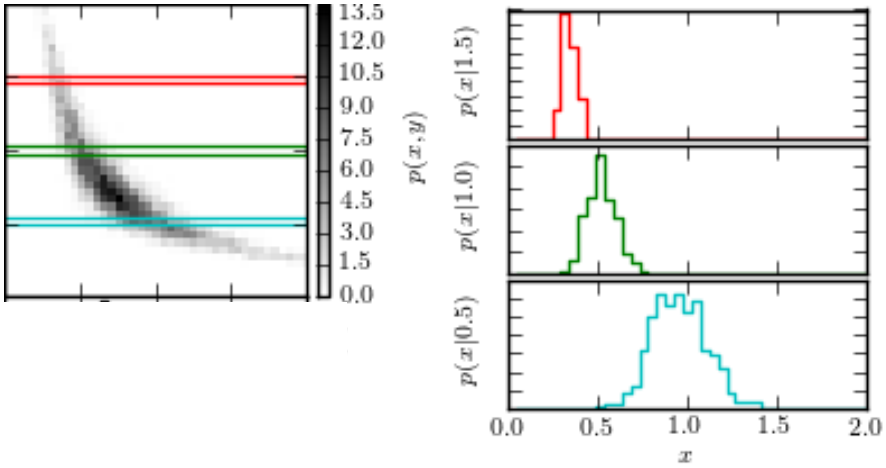


Images on the Internet

Statistical learning

Joint distributions P(X,Y) encode relationships or functions, and can be used for prediction X→Y.

P(X,Y)



X=1.5→P(Y|X=1.5)

X=1.0→P(Y|X=1.0)

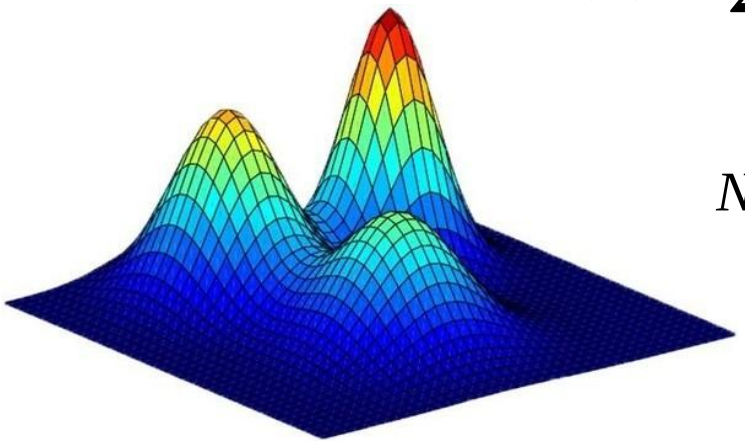
X=0.5→P(Y|X=0.5)

Gaussian mixture models

- We will use **Gaussian mixtures** in the context of generative data models for controlled examples used to understand some basic questions of SL
- Gaussian mixture models is also a ML tool for estimating P(X), from the part of ML called **kernel density estimation**

Gaussian mixture models

GMM represents P(X) as a sum of Gaussian densities



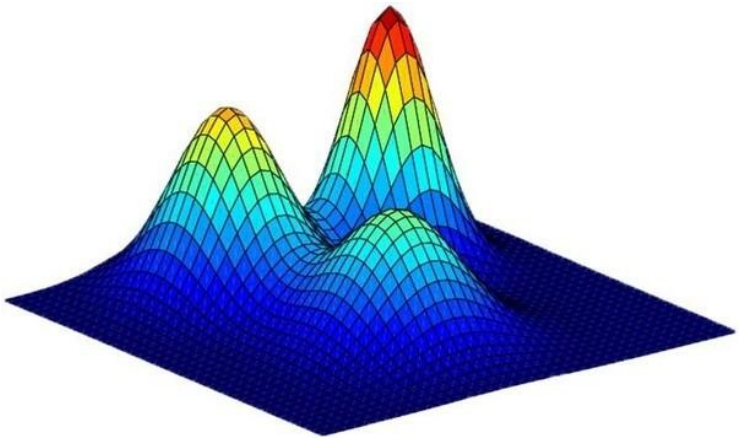
$$P(x) = \sum_k \alpha_k N(x; \mu_k, \Sigma_k), \ \sum_k \alpha_k = 1$$

$$N(x; \mu, \Sigma) = |2\pi\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\}$$

Gaussian mixture models

From generative perspective, GMM is a following random process:

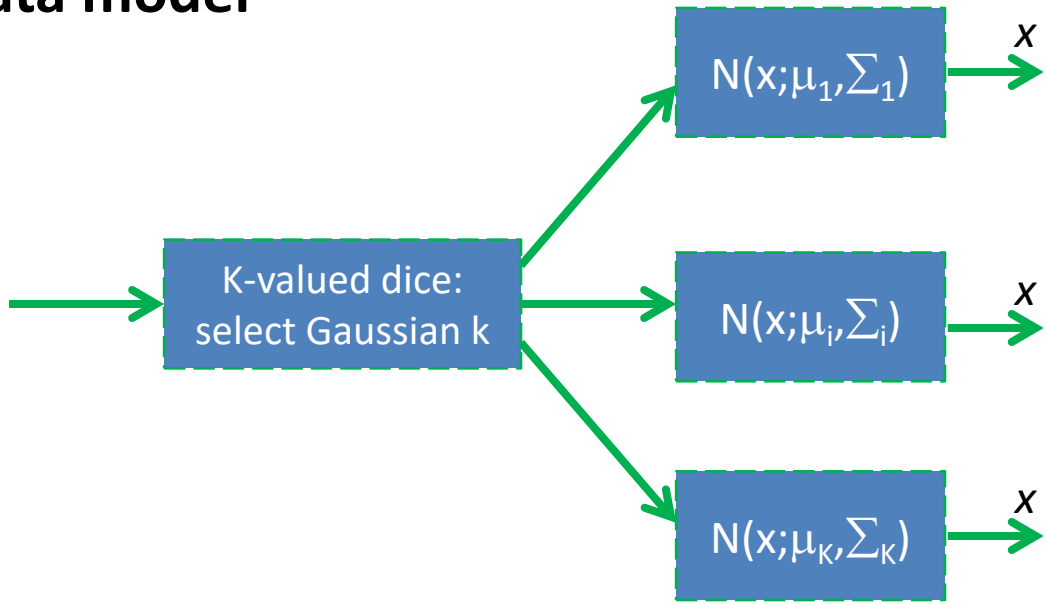
1. Choose one Gaussian with P{k}=α_k
2. Choose x from P(x)=N(x;μ_k,Σ_k)

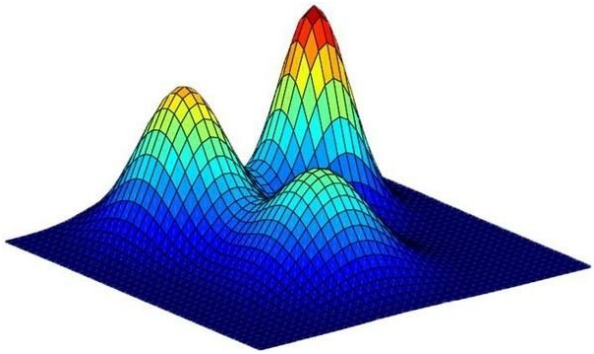


$$P(x) = \sum_k \alpha_k N(x; \mu_k, \Sigma_k)$$

Gaussian mixture models

GMM generative data model





$$P(x) = \sum_k \alpha_k N(x; \mu_k, \Sigma_k)$$

ML prediction problem

- Prediction is one of the main problems in ML
- Given data model P(X,Y) and an input instance x, predict the corresponding y=f(x)
- The functional form of the prediction problem, f:x$\rightarrow$y, has numerous real-life examples we have mentioned during past classes (we do not repeat those here)

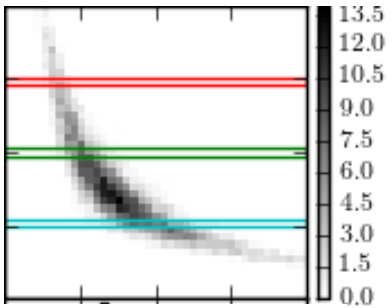
ML prediction problem

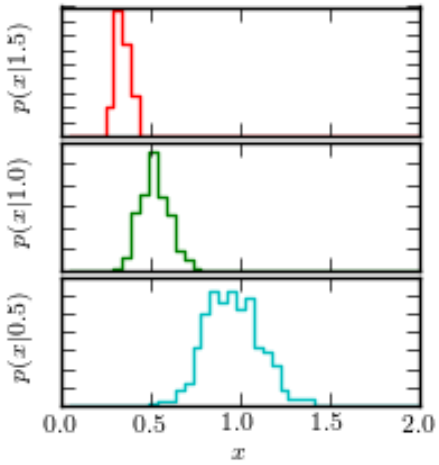
Impossibility of perfect prediction: If y is not precisely determined by x, what must be y(x) ?

Answer:

- *What should be the prediction y(1.5)?*
- *What should be the prediction y(1.0)?*
- *What should be the prediction y(0.5)?*

P(X,Y)





P(Y|X=1.5)

P(Y|X=1.0)

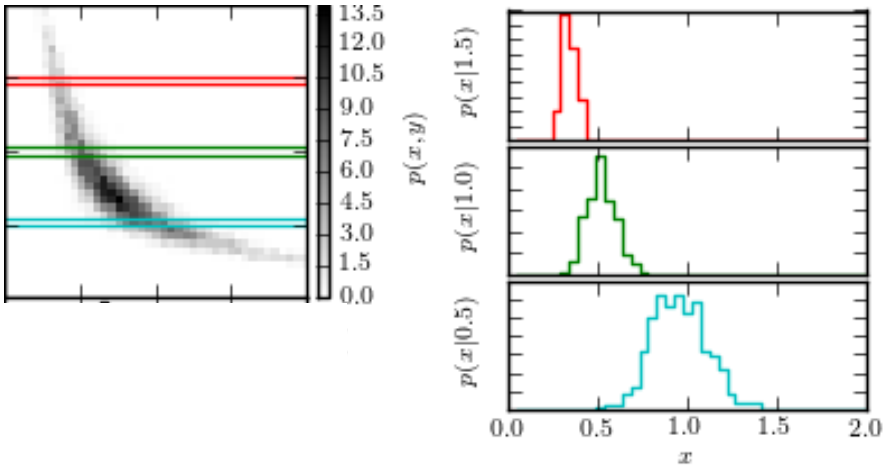
P(Y|X=0.5)

ML prediction problem

Despite that problem, a single value still needs to be chosen for every x if we are to predict anything. We call this value a **prediction** or **hat-value**, y^(x), where hat indicates an estimation.

y^(1.5)=0.35
y^(1.0)=0.50
y^(0.5)=1.00
---
y^(x)=0.5/x

P(X,Y)



Some terminology for prediction problem

In P(Y|X), X is called:

- **predictor**
- **independent**
- **explanatory or**
- **input**

In P(Y|X), Y is called:

- **prediction**
- **dependent**
- **response or**
- **output**

The general idea is: input→output, predictor→prediction, explanatory→response, independent→dependent

Some terminology for prediction problem

In P(Y|X), X and Y may be:

- **numerical** (real or integer values)
- **categorical** (a set number of values either numerical or non-numerical, for example {1,2,3}, {True,False}, {Alice|Bob,Sam})
- **ordinal** (a set number of ordered discrete values either numerical or non-numerical, for example {didn't liked, liked, loved})

Some terminology for prediction problem

Try name these yourself:

Total sample size, n = 22

Gender:	Female = 50% (n = 11) Male = 50% (n = 11)
Ethnicity:	White = 32% (n = 7) Black = 32% (n = 7) Asian = 9% (n = 2) Native American = 9% (n = 2) Latino/a = 18% (n = 4)
Program City	Boston = 50% (n = 11) New York = 50% (n = 11)
Participated in Summer Program:	Yes = 91% (n = 20) No =9% (n = 2)
Satisfied with Program Experience: *(valid n = 20)*	Very Dissatisfied = 10% (n = 2) Dissatisfied = 15% (n = 3) Satisfied = 60% (n = 12) Very Satisfied = 15% (n = 3)
Age	

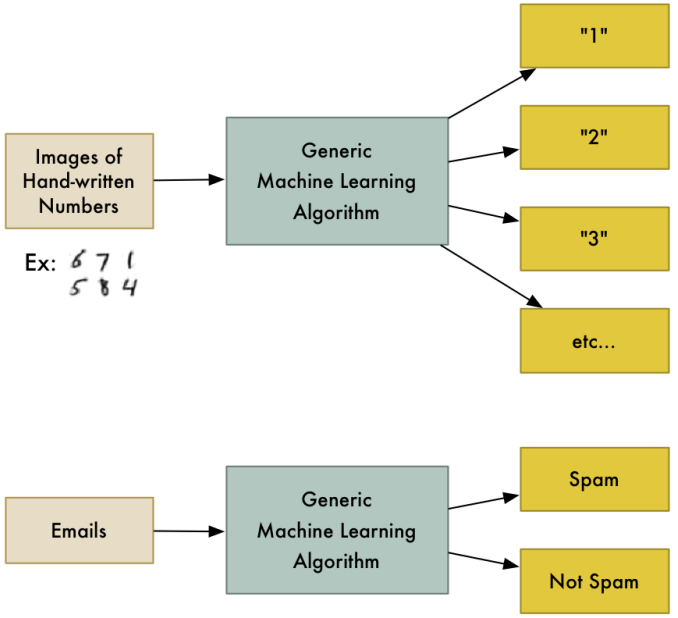
Some terminology for prediction problem

The learning prediction problems are divided into ***classification*** and ***regression*** depending on the type of the output variable

- Output numeric – regression
- Output categorical or ordinal – classification

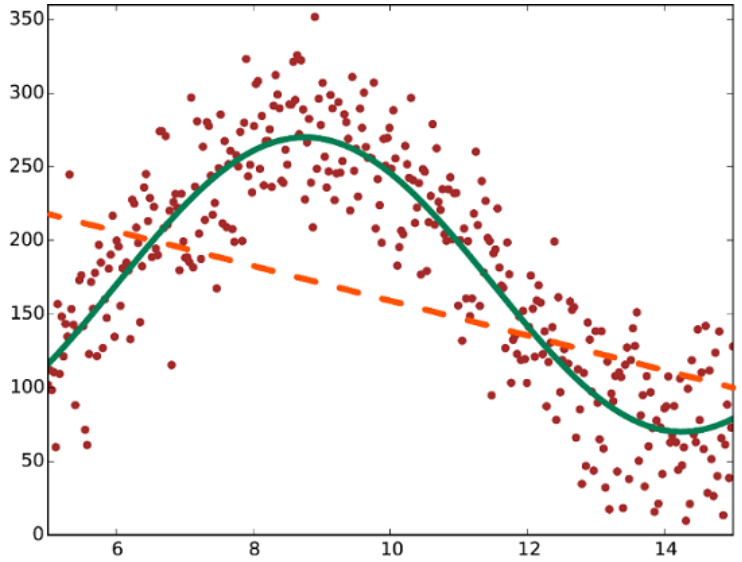
Some terminology for prediction problem

If Y is categorical or ordinal, prediction is understood as sorting inputs X into several bins Y, one for each distinct value or category of Y, and is **classification**



Some terminology for prediction problem

If Y is numerical the problem is understood as fitting a function Y=*f*(X) and is called **regression**



Some terminology for prediction problem

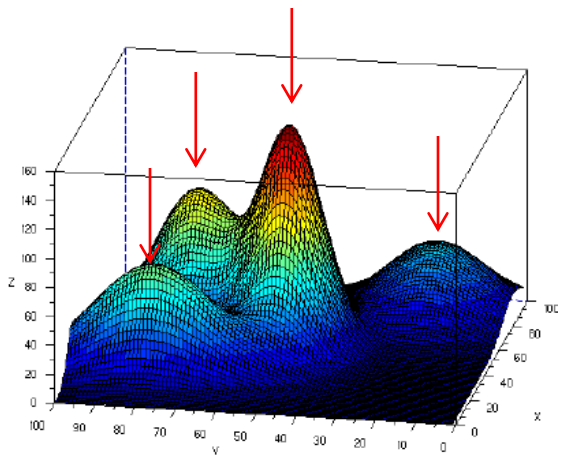
- Both **classification** and **regression** are examples of **supervised learning** – in supervised learning a relationship y^(x) is obtained by giving machine a set of examples ($x_i \rightarrow y_i$), or *teaching the machine*
- Supervised learning is very much like the education in school – you teach the machine/ algorithm by showing it examples of correctly doing things

Some terminology for prediction problem

- **Unsupervised learning** is learning from unstructured data – in this type of learning there is no labels Y attached to data, only data itself X. Machine needs to discover useful structure in the data, usually by locating clusters in P(X) which are distinct from other data

Some terminology for prediction problem

GMM discussed earlier is a good example of the above philosophy – every bump in P(X) is thought as a distinct examples' generator!



Two simple approaches to prediction

- Linear model is one of the oldest and best understood data models, spanning well into classical statistics of 1900's
- It associates to a realization of predictors X_i a prediction Y calculated according to $\mathbf{\hat{Y}=\theta_0+\Sigma\theta_i X_i}$
- The term θ_0 is known as offset (in statistics) or bias (in ML). We remove θ_0 by artificially extending the list of predictors X_i to include a new **dummy variable $X_0=1$**, so that $\mathbf{\hat{Y}=\theta^T X}$

Two simple approaches to prediction

- Linear model geometrically represents a **hyperplane** in (p+1)-dimension of inputs+output - (X,Y)
- With dummy bias variable, linreg hyperplane passes through the origin (0,0) and is also a subspace

Two simple approaches to prediction

- One of the most popular methods for fitting linear model is the method of **Least Squares** (**LS** for short), prescribing minimization of the **Residual Sum of Squares**:

$$RSS(\theta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - x_i^T \theta)^2$$

- RSS(θ) is a quadratic function and thus always has a minimum

Two simple approaches to prediction

Octave example (in class)

```
X=randn(100,2)
theta=[1;-0.5]
Y=X*theta+randn(100,1)
plot3(X(:,1),X(:,2),Y(:),'.')
RSS=@(th) sum((Y-th(1)*X(:,1)-th(2)*X(:,2)).^2)
RSS([0;0])
RSS([1;0])
RSS([1;1])
RSS([1;-1])
RSS([1;-0.5])
RSS([2;-0.5])
```

Two simple approaches to prediction

- We rewrite RSS(θ) into matrix notation:

$$RSS(\theta) = | y - \hat{y} |^2 = (y - X\theta)^T (y - X\theta)$$

  where **y** is now nx1 **vector of outputs** and **X** is nxp **matrix of input vectors stacked in rows** (aka the **feature matrix**)

- From optimization theory, the minimum condition is $\partial_\theta RSS(\theta)=0$, so that

$$X^T (y - X\hat{\theta}) = 0$$

$$\hat{\theta} = (X^T X)^{-1} X^T y$$

Two simple approaches to prediction

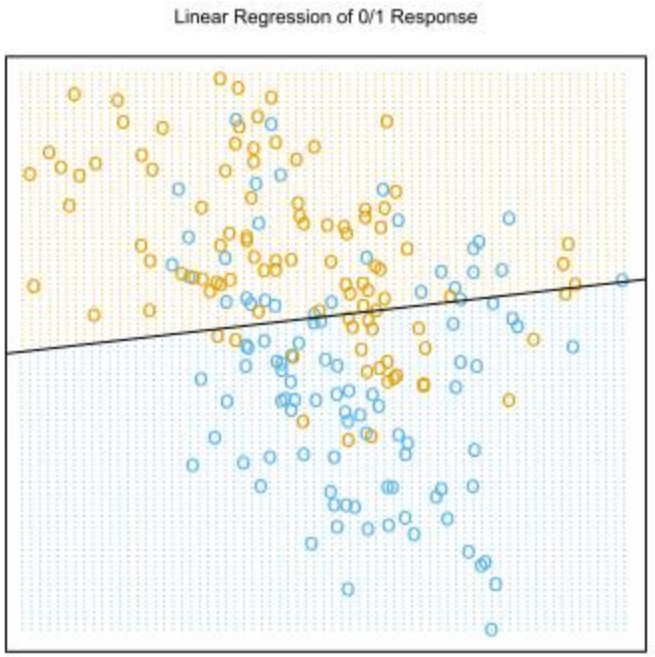
- Octave example exact solution:

  thhat=(X'*X)^-1*X'*Y
  RSS(thhat)

Two simple approaches to prediction

Classification of data using linear model:

- X is 100x2 matrix of 100 example points [X1,X2]
- Y is 100x1 vector of +1 if point is orange and 0 if point is blue
- $\theta\hat{}$ is calculated using $(X^TX)^{-1}X^TY$ (2x1 the optimal linear model weights)



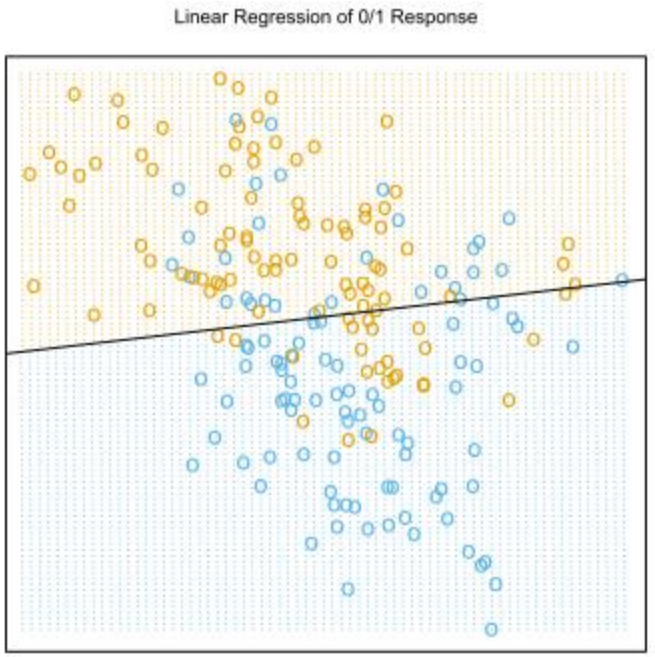
- Predictions $Y\hat{}=x^T\theta\hat{}$, $Y\hat{}>0.5$ taken as orange class

Two simple approaches to prediction

Classification of synthetic data using linear model:

$$\hat{G} = \begin{cases} \text{"}orange\text{"} \text{ if } \hat{Y} > 0.5 \\ \text{"}blue\text{"} \quad \text{if } \hat{Y} \leq 0.5 \end{cases}$$

$$\text{Decision Boundary } \{x : x^T \hat{\theta} = 0.5\}$$



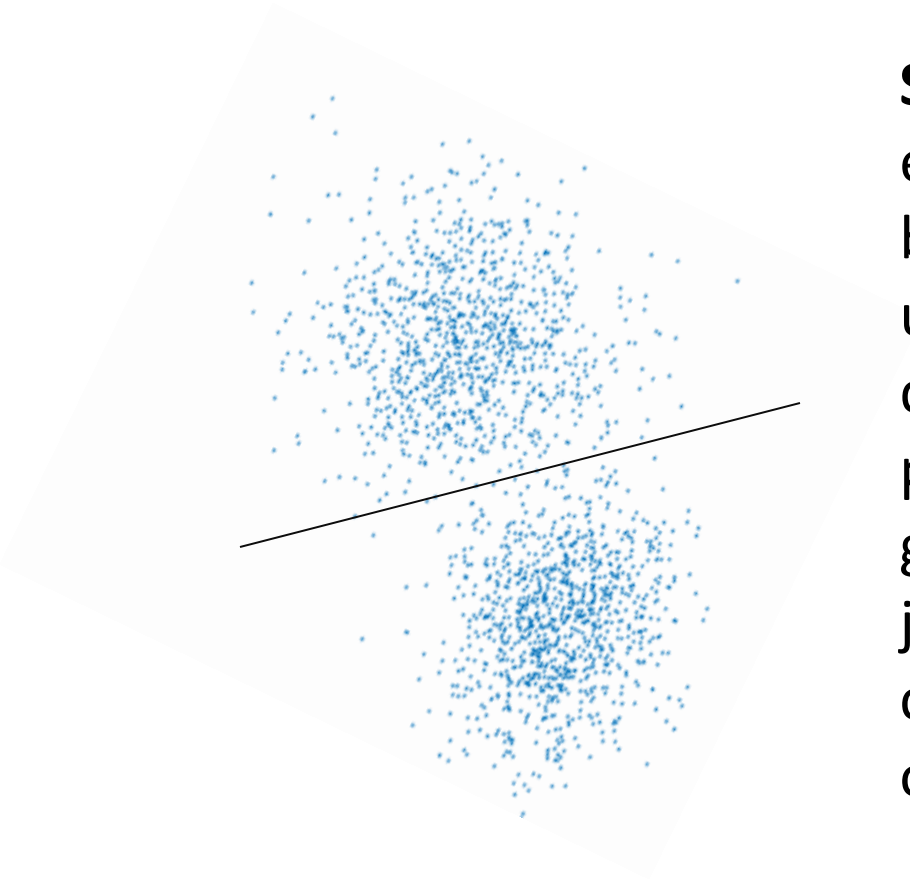
Is this good result or bad (there seem to be lots of points on the wrong sides of divide??? Perhaps our model is too rigid, or are such errors unavoidable?)

Two simple approaches to prediction

Two scenarios:

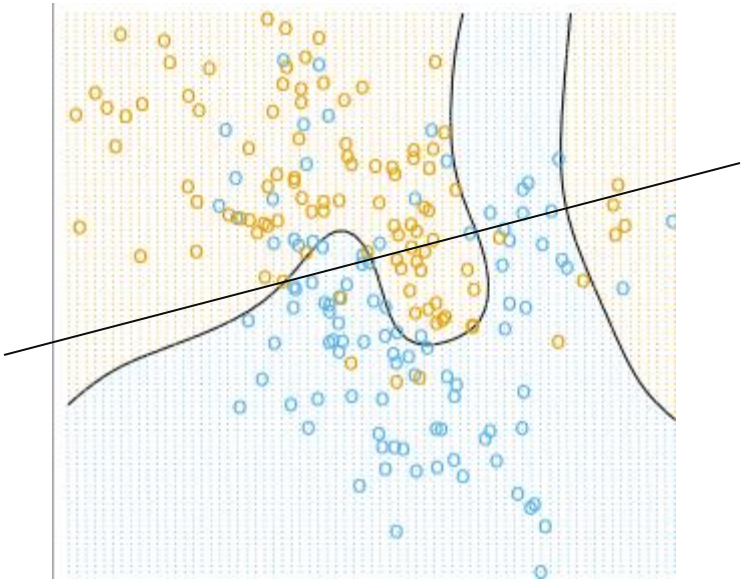
- **Scenario 1:** The training data in each class was generated from a bivariate Gaussian distributions with uncorrelated components and different means
- **Scenario 2:** The training data in each class came from a mixture of 10 low-variance Gaussians, with individual means themselves distributed as Gaussians

Two simple approaches to prediction



Scenario 1: The training data in each class was generated from bivariate (X_1,X_2) Gaussian with uncorrelated components and different means – in that case our performance is likely as good as it gets and wrong points we see are just the result of the chance crossings of (X_1,X_2) over to the other side

Two simple approaches to prediction



Scenario 2: The training data in each class came from a mixture of 10 Gaussians clusters, with individual means distributed according to a Gaussians distribution – performance is not so good in that case, and one can want to improve it a lot by considering better **decision boundary**

Two simple approaches to prediction

K-Nearest-Neighbor Method – the prediction at each new point x is the average of $y_{(l)}$ of K-nearest neighbors $x_{(l)}$ of x in the data

$$\hat{Y}(x) = \frac{1}{K} \sum_{x_i \in N_K(x)} y_i$$

$N_K(x)$ stands for the set of K closest neighbors of x

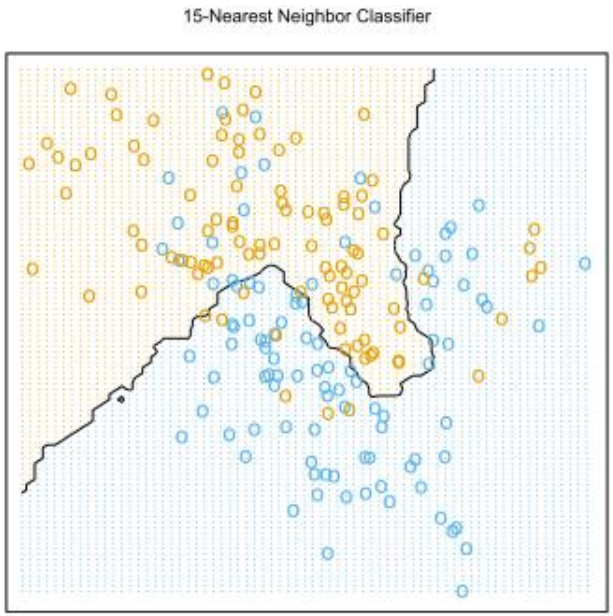
$$\hat{G} = \begin{cases} "orange" \text{ if } \hat{Y} > 0.5 \\ "blue" \quad \text{if } \hat{Y} \leq 0.5 \end{cases}$$

Two simple approaches to prediction

The outcomes:

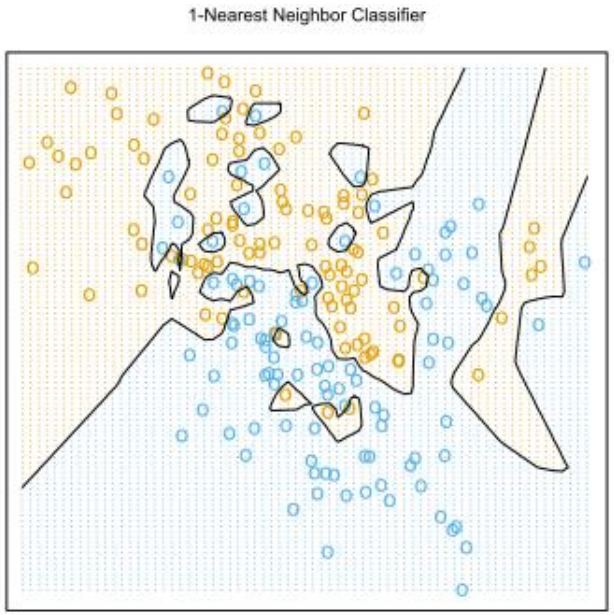
Linear regression is very rigid and simple, provides simple and clear but poor interpretation of the data

$$\hat{Y}(x) = x^T\theta$$



K=15-NN classifier is able to fit better the apparent excursions of orange data into blue domain

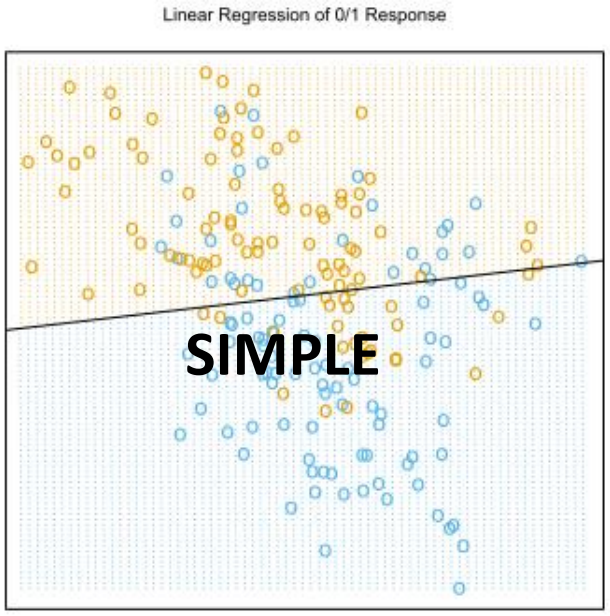
$$\hat{Y}(x) = \frac{1}{K}\sum_{x_i \in N_K(x)} y_i$$



But K=1-NN classifier essentially creates a separate domain for each point, resulting in very jagged and noisy-looking decision boundary

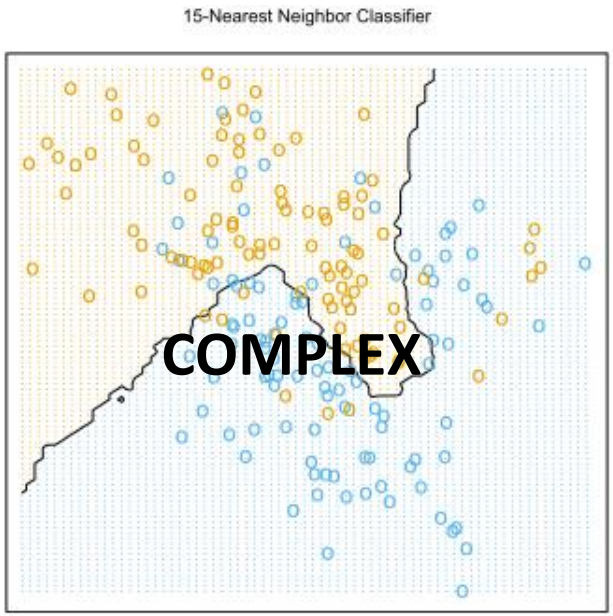
Two simple approaches to prediction

The outcomes:



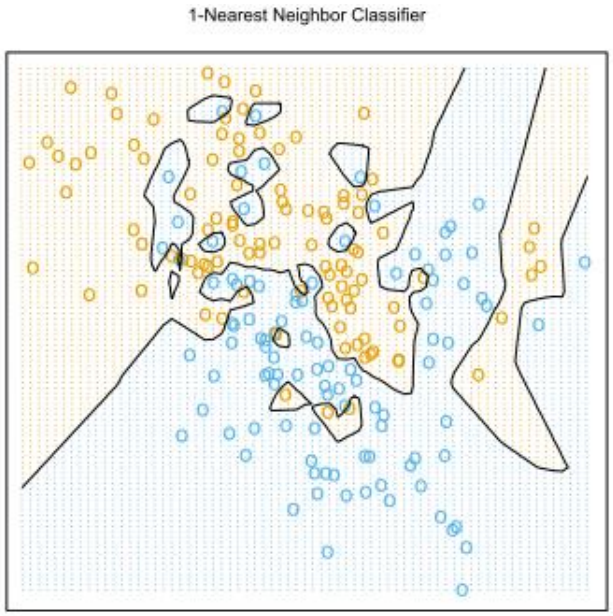
Linear regression is very rigid and simple, provides simple and clear but poor interpretation of the data

$$\hat{Y}(x) = x^T\theta$$



K=15-NN classifier is able to fit better the apparent excursions of orange data into blue domain

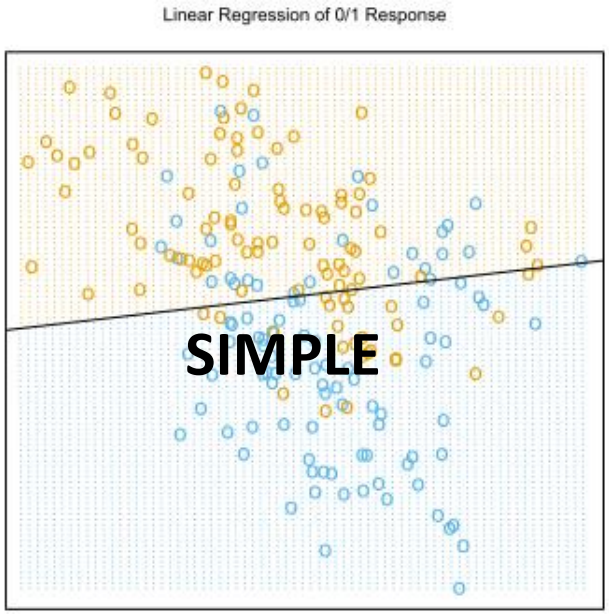
$$\hat{Y}(x) = \frac{1}{K}\sum_{x_i \in N_K(x)} y_i$$

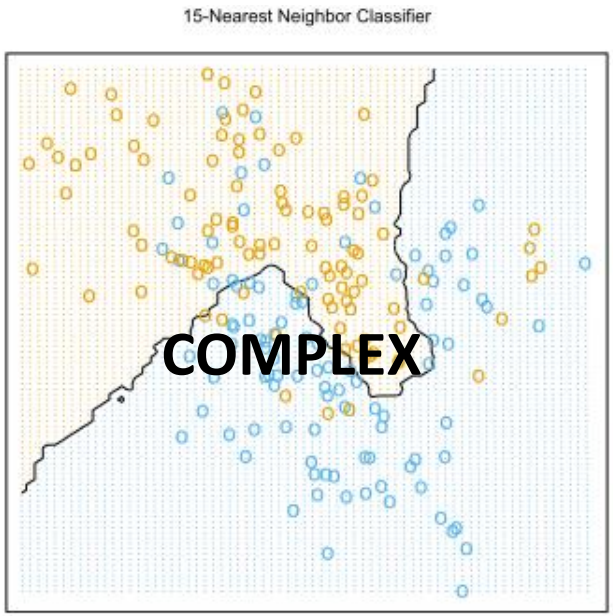


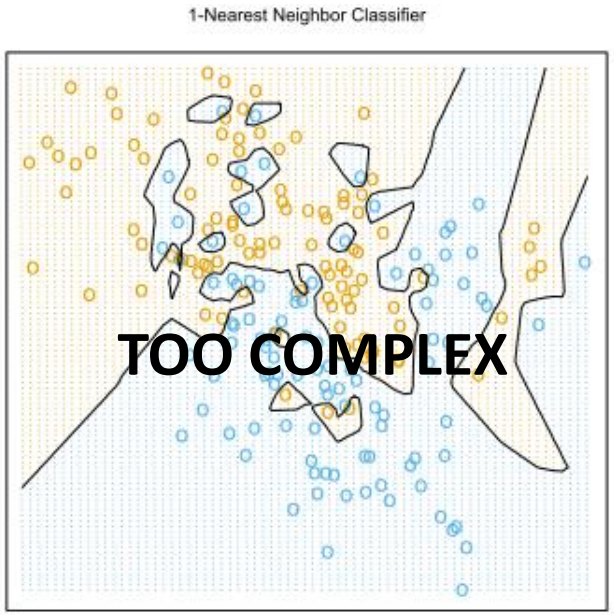
But K=1-NN classifier essentially creates a separate domain for each point, resulting in very jagged and noisy-looking decision boundary

Two simple approaches to prediction

The outcomes:







Linear regression is very rigid and simple, provides simple and clear but poor interpretation of the data

$$\hat{Y}(x) = x^T\theta$$

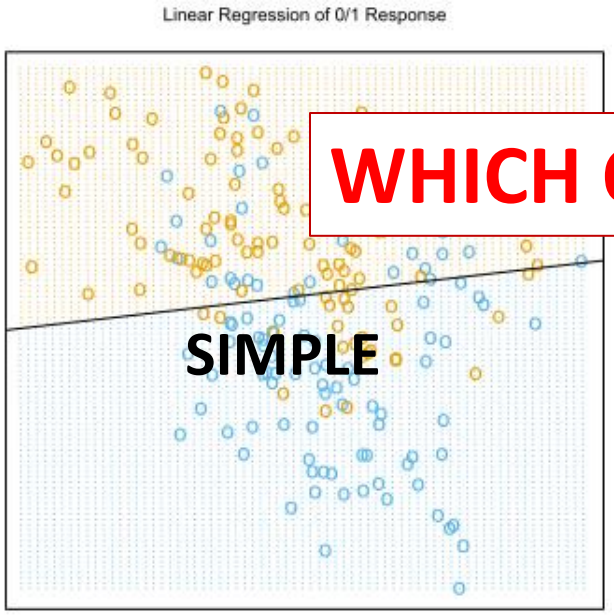
K=15-NN classifier is able to fit better the apparent excursions of orange data into blue domain

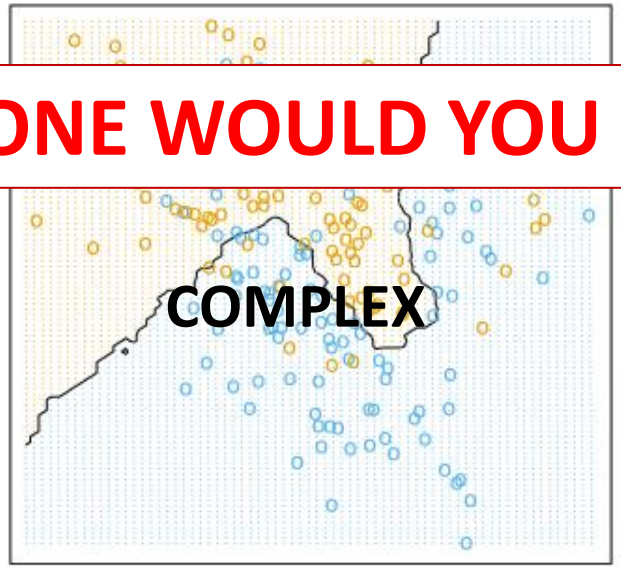
$$\hat{Y}(x) = \frac{1}{K}\sum_{x_i \in N_K(x)} y_i$$

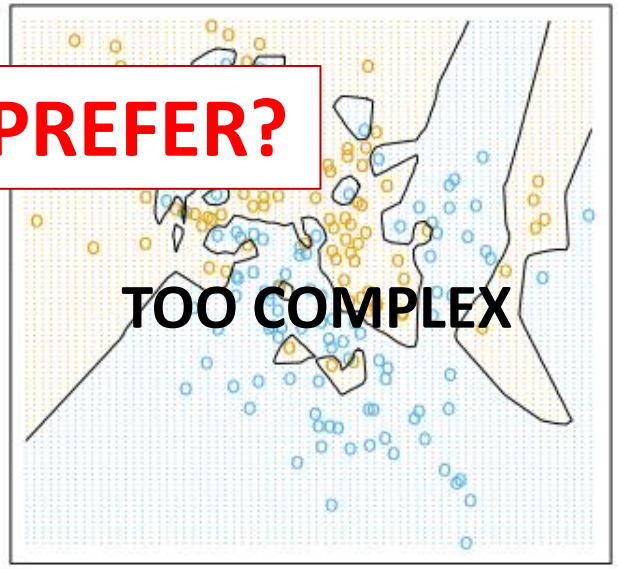
But K=1-NN classifier essentially creates a separate domain for each point, resulting in very jagged and noisy-looking decision boundary

Two simple approaches to prediction

The outcomes:







WHICH ONE WOULD YOU PREFER?

Linear regression is very rigid and simple, provides simple and clear but poor interpretation of the data

$$\hat{Y}(x) = x^T\theta$$

K=15-NN classifier is able to fit better the apparent excursions of orange data into blue domain

$$\hat{Y}(x) = \frac{1}{K}\sum_{x_i \in N_K(x)} y_i$$

But K=1-NN classifier essentially creates a separate domain for each point, resulting in very jagged and noisy-looking decision boundary

Two simple approaches to prediction

Two bottom line conclusions:

- Complexity is certainly not something that we like in our data models – too complex has the appearance of wrong
- It is not really possible to tell which model is correct just by looking at data

Two simple approaches to prediction

So what was our data really?

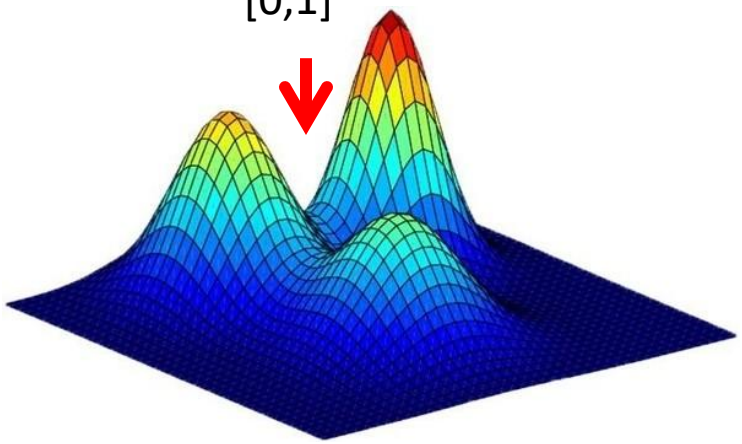
- Our data was generated from a mixture of 20 Gaussians, 10 per orange and blue class, each centered on a random center-point, themselves chosen from a bivariate Normal distribution with mean [0,1] for orange and [1,0] for blue class, and the covariance $1/5 \cdot \mathrm{I}$

Two simple approaches to prediction

So what was our data really?

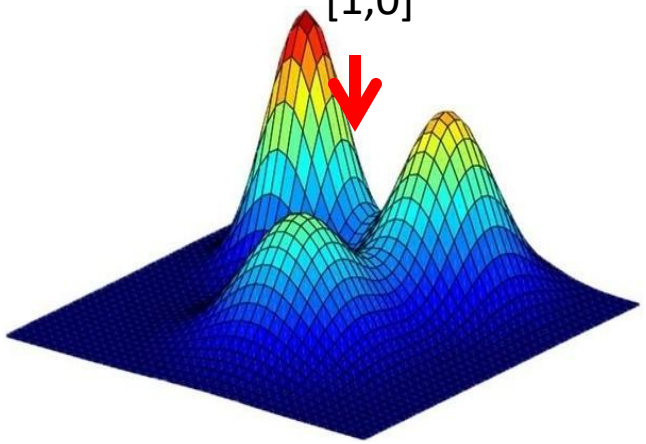
Orange P(X)

[0,1]



Blue P(X)

[1,0]



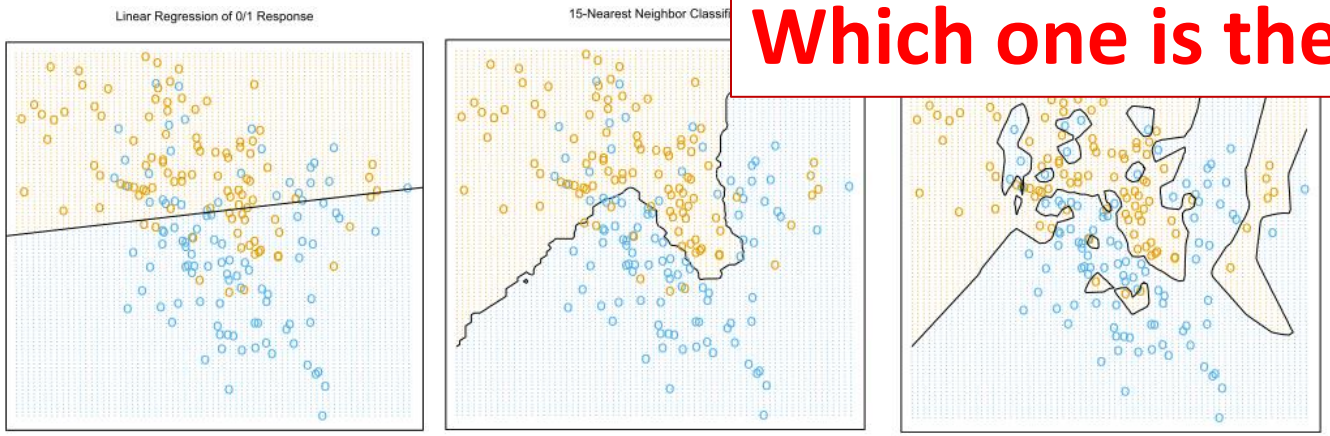
Two simple approaches to prediction

Statistically best decision?

- Optimal decision rule – classify "orange" if $P_{orange}(X) > P_{blue}(X)$, where $P_{orange/blue}(X)$ are the exact densities calculated from their GMM formulas (which we know since this is a synthetic example)

$$gmm: P(x) = \sum_k \alpha_k N(x; \mu_k, \Sigma_k)$$

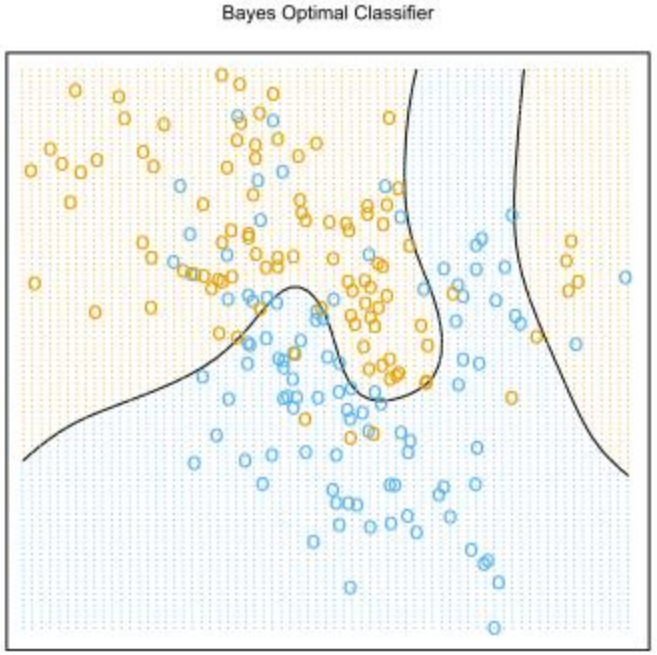
Two simple approaches to prediction



Which one is the best match?

Best (optimal) decision partition:

$$P(x \mid orange) > P(x \mid blue)$$

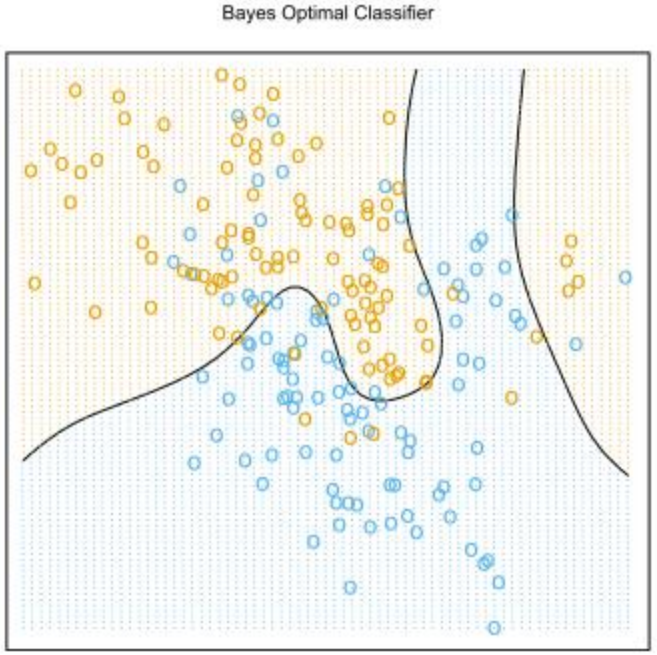


Two simple approaches to prediction

Note that **best decision rule** doesn't mean 100% correct decision rule – there are still a lot of orange points to the bottom and blue points to the top of the boundary– it is just that **it is not possible to do better**!

Best (optimal) decision partition:

$$P(x \mid orange) > P(x \mid blue)$$



Statistical Decision Theory

Suppose that we have a perfect knowledge of data model, P(X,Y), as in the last example. How can the optimal “prediction” rule Y^(*x*) be constructed in that situation?

Statistical Decision Theory

- In order to find the best prediction rule, it is necessary to define a *goodness criterion* of how good or bad each specific choice of the prediction rule y^(x) is, known as the **loss function**
- **Loss function** is written as **L(y,y^)** and describes how bad a guess y^ is given the true value y; **the lower the loss – the better the guess**
- **Square Loss $L(y,\hat{y})=(y-\hat{y})^2$** is by far the most common and convenient choice in ML

Statistical Decision Theory

- **Expected Prediction Error** also **Expected Loss** for a prediction rule y^(x)=f(x) is defined as

$$EPE(f) = E_{P(X,Y)}[L(Y, f(X))] = \sum_{(X,Y)} L(Y, f(X))P(X,Y)$$

(note that EPE is defined as the **exact expectation value** over the **true distribution of data P(X,Y)**, and thus is **generally not computable** because either is unknown)

- For square-loss, the minimum of EPE can be shown to be achieved when

$$\hat{y}(x) = E[Y \mid X = x]$$

Statistical Decision Theory

Derivation of the optimal prediction rule for the square-loss function:

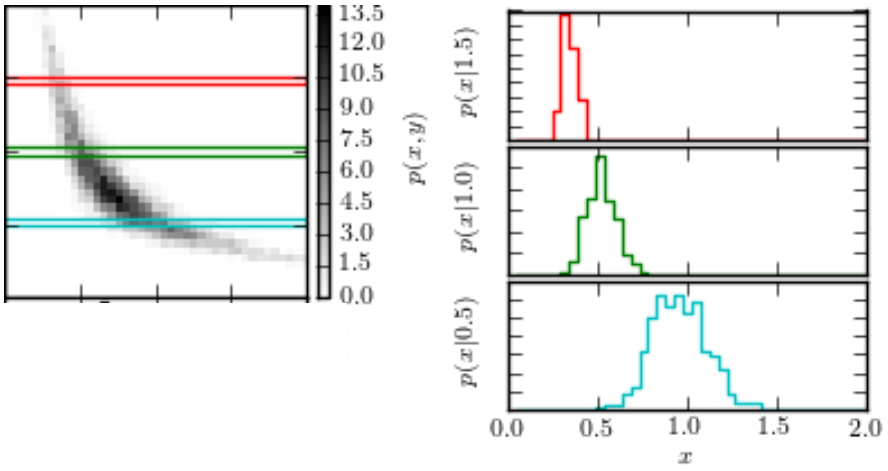
$$
\begin{aligned}
EPE(f) &= \sum (y - f(x))^2 P(x, y) \\
&= \sum_x P(x) \sum_y (y - f(x))^2 P(y \mid x) \\
&= \sum_x P(x) \sum_y (y - E[Y \mid X = x] + E[Y \mid X = x] - f(x))^2 P(y \mid x) \\
&= \sum_x P(x) \left[\mathrm{var}(Y \mid X = x) + (E[Y \mid X = x] - f(x))^2 \right] \\
&= E_X[\mathrm{var}(Y \mid X = x)] + E_X(E[Y \mid X = x] - f(x))^2 \\
\Rightarrow \hat{y}_{opt}(x) &= f_{opt}(x) = E[Y \mid X = x]
\end{aligned}
$$

Note that we still suffer irreducible error $E_x \mathrm{var}(y|x)$ even with the optimal prediction rule – this is due to uncertainty in $Y|X$

Statistical Decision Theory

From the example before:
y^(1.5)=E[Y|X=1.5]=0.33 (the mean at X=1.5)
y^(1.0)=E[Y|X=1.0]=0.50 (the mean at X=1.0)
y^(0.5)=E[Y|X=0.5]=1.00 (the mean at X=0.5)
Overall optimal prediction rule is y^(x)=0.5/x

P(X,Y)



X=1.5→P(Y|X=1.5)

X=1.0→P(Y|X=1.0)

X=0.5→P(Y|X=0.5)

Statistical Decision Theory

Connections with the KNN method:

- If we approximate the true expectation value E[Y|X=x] with the average over example points (x_i,y_i) that are close to x instead, then we get what looks very much like the KNN method:

$$\hat{f}(x) = Ave(y_i \mid x_i \in N_K(x)) \approx E[Y \mid X = x]$$

- In fact, one can show that KNN **becomes exactly** E[Y|X=x] as $n \to \infty$

Statistical Decision Theory

Theorem: KNN is optimal universal regressor equivalent to calculating the **local expectation value** E[Y|X=x] in the limit n,k$\rightarrow\infty$, k/n$\rightarrow$0

$$\hat{f}_{KNN}(x) = Ave(y_i \mid x_i \in N_k(x)) = E[Y \mid X = x], n, k \rightarrow \infty \text{ and } k/n \rightarrow 0$$

Statistical Decision Theory

If we already have a provably optimal universal approximator for the best predictions for any P(X,Y), why not call the case closed then? What's the problem?

Statistical Decision Theory

- The answer is: we rarely (actually never) have the situation $n,k \to \infty, k/n \to 0$. Usually, the amount of data one has is small, if not miniscule, on the scales of the above theorem.
- To see that, consider example where we want to model $E[Y|X]$ on a 2D domain $[0,1]^2$ with a step 0.01 and by using averages of at least 100 examples. Then we need 100x100x100=1,000,000 examples of the data (X,Y), and that won't even result in a particularly accurate approximation to $E[Y|X]$ since the average of 100 random numbers still contains about $\approx 1/\sqrt{100}=10\%$ random variation ! (That is, the **statistical error** of our predictions will be at least 10% with10^6 data points in the dataset!)

Statistical Decision Theory

Let's ask if the situation is the same in linear model

- Let's say real data did come from linear model

$$y \leftarrow x^T\theta^* + \varepsilon,\ \varepsilon \sim N(\mu, \Sigma)$$

- One can show theoretically (from EPE) that for the true parameter following is true

$$\theta^* = E[XX^T]^{-1}E[XY]$$

- Comparing that to linreg formulas from before, we observe that those are simply the approximations of the above expectation values via averages:

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = Ave(x_i x_i^T)^{-1} Ave(x_i y_i) \approx E[XX^T]^{-1}E[XY]$$

Statistical Decision Theory

Linreg is an approximation using averages of the true linear model formula in expectation values:

$$\theta^* = E[XX^T]^{-1}E[XY]$$

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\mathbf{X} = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \\ \dots \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \dots \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{y} = \sum_{i=1}^{n} y_i \cdot x_i = nAve(x_i y_i) \approx nE[XY]$$

$$(\mathbf{X}^T\mathbf{X}) = \sum_{i=1}^{n} x_i x_i^T = nAve[x_i x_i^T] \approx nE[XX^T]$$

Statistical Decision Theory

How accurate is that approximation?

- Each element of $\theta\hat{}$ is calculated by using averages of n data points to estimate E[XY] and $E[XX^T]$.
- By LLN, the statistical random error in those estimates is of the order $1/\sqrt{n}$, and so for n=10,000 already the $\theta\hat{}$ estimate is 1% accurate!

Statistical Decision Theory

- Linear regression is by miles and miles superior to KNN in this case: In KNN, a really bad 10% prediction required 10^6 data points. In linear regression, superior 1% predictions could be achieved with just 10^4 data points.
- The progress here was made by allowing **all data points** to bear on the estimation of θ^*, not just local subsample! In KNN, only about **100 out of 10^6 data points are used** in estimation of each y^(x). In linreg – **all data points are used** for estimating y^(x)!

Statistical Decision Theory

This observation is the first manifestation of a very widely seen principle in ML, which we can express for now as the discrepancy between complexity and accuracy:

- Simple models, where few parameters affect lots of data, can be estimated with superior accuracy from limited data due to above their property, and used for superior predictions
- Complex models, where lots of parameters affect lots of points, cannot be estimated well from small amounts of data, leading to bad predictions

Statistical Decision Theory

- However, simplicity of a model implies that one may be unable to construct models sufficiently rich to describe real data
- **If generative model was not linear to begin with, what is the worth of having a super-accurate θ^* estimate of data which never came from $Y=X^T\theta$?**

Statistical Decision Theory

- We now turn to statistical decision theory for classification: If the optimal prediction rule in regression was $f(x)=\mathrm{E}[Y|X=x]$, what should it be in classification?

Statistical Decision Theory

- In classification problem, Y is a categorical variable taking on a discrete set of possible values from some set G; these are also frequently called **labels**.
- A very logical choice for the loss function here is the **zero-one loss function**:

$$L_{zero-one}(y,\hat{y}) = \begin{cases} 0, \; if \;\; y = \hat{y} \\ 1, \; if \;\; y \neq \hat{y} \end{cases}$$

Statistical Decision Theory

- 0-1 loss function counts the number of errors made during classification, and the expected 0-1 loss is the probability of misclassification:

$$\sum L_{zero-one}(y, f(x)) = \#\, of\ incorrect\ predictions$$

$$E[L_{zero-one}(y, f(x))] = \Pr\{incorrect\ prediction\}$$

Statistical Decision Theory

We can perform a similar analysis for the **zero-one loss function** that we performed for the **square-loss function**:

$$
\begin{aligned}
EPE(f) &= \sum L_{0-1}(y, f(x))P(x, y) \\
&= \sum_x P(x) \sum_y L_{0-1}(y, f(x))P(y \mid x) \\
&= \sum_x P(x)(1 - \Pr\{y = f(x)\}) \\
\min EPE \Rightarrow f(x) &= \arg\max_y P(y \mid X = x)
\end{aligned}
$$

Statistical Decision Theory

Choosing for each input x the output class y with the largest Pr{y|x} minimizes the expected 0-1 loss

$$EPE(f) = \sum L_{0-1}(y, f(x))P(x, y)$$
$$= 1 - \sum_{x} P(x)\Pr\{y = f(x)\}$$
$$\Rightarrow f(x) = \arg\max_{y} P(y \mid X = x)$$

Statistical Decision Theory

Example P(Y|X) for optimal prediction rule in classification:

	X=-4	X=-2	X=0	X=1	X=3
Y=C1	0.25	0.2	0.4	0.5	0.2
Y=C2	0.50	0.1	0.3	0.3	0.4
Y=C3	0.25	0.7	0.3	0.2	0.4
Best	**C2**	**C3**	**C1**	**C1**	**C2/C3**

$$\hat{y}(x) = \arg\max_{y} P(Y = y \mid X = x)$$

Statistical Decision Theory

The **optimal prediction rule for classification** is the one that outputs the highest probability option P(y|x) for each *x*:

$$f(x) = \arg\max_{y} P(y \mid X = x)$$

Such prediction rule in classification is known as the **Bayes classifier**. Bayes classifier is the optimal classifier in the **probability-of-error** sense.

QUESTIONS FOR SELF-CONTROL

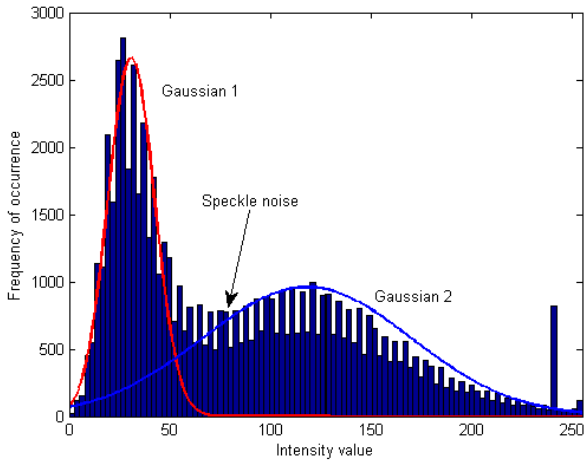
- Explain in your words what is mean by “Data=P(X)” statement
- Explain in your words how learning P(X) can be used to discover entities from data
- Explain in your words how learning P(X,Y) can be used to describe functional relationships and prediction
- Describe what a Gaussian Mixture Model is
- Write the formula for RSS of linear model
- Define a linear model
- Derive the θ^ equation for linear model
- Describe what a feature matrix is. Suppose you have 1000 examples of 100-long feature vectors Xi. What the feature matrix **X** be then? What the output vector **y** be then?
- How can we use linear model for classification?
- Describe the KNN algorithm
- Which models are more complex, in the sense discussed in the class: K=1 NN, K=10 NN, K=100 NN?

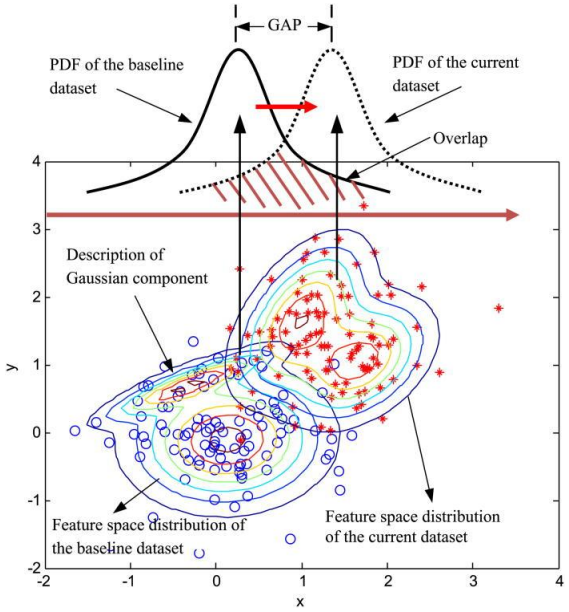
- Explain in your words why linear model allows predictions to be carried out more accurately than KNN.
- By the same principle, if you have a p=100 parameter linear model or p=10,000 neural network, which model will be more accurately estimated from n=10,000 data points.
- Define square loss function.
- In statistical decision theory, what is the optimal prediction rule for regression?
- How KNN is related to the optimal prediction rule in regression $\hat{y}(x)=E[Y|X=x]$?
- What is meant by “local averaging” in KNN or regression prediction context?
- In statistical decision theory, what is the optimal prediction rule for classification?
- Define is 0-1 loss function.
- Define Expected Prediction Error and Expected Loss. Why in practice it is impossible to compute EPE?

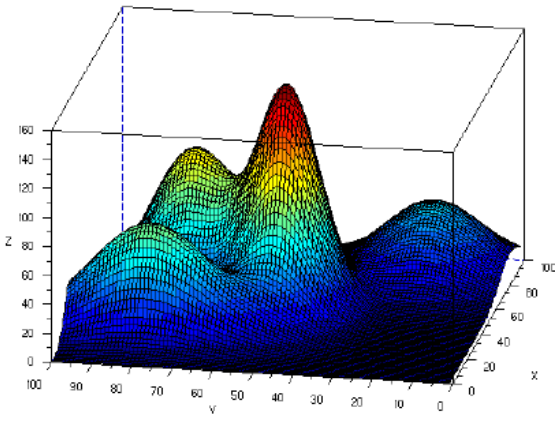
ADVANCED

Gaussian mixture models as kernel density estimator

Any P(X) can be described with an arbitrary accuracy by a GMM with the number of components in the mixture $K \rightarrow \infty$.

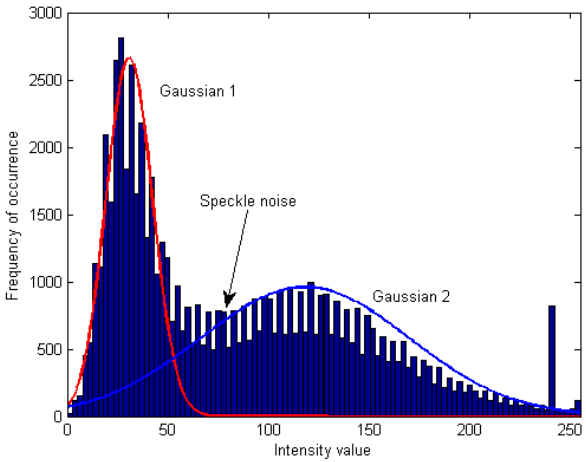


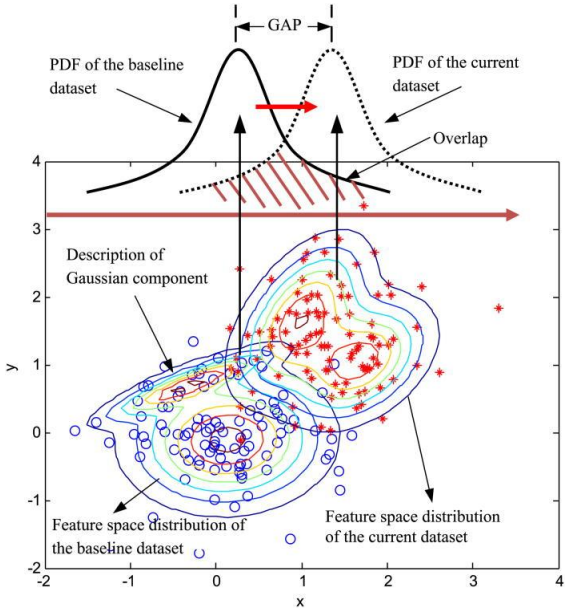


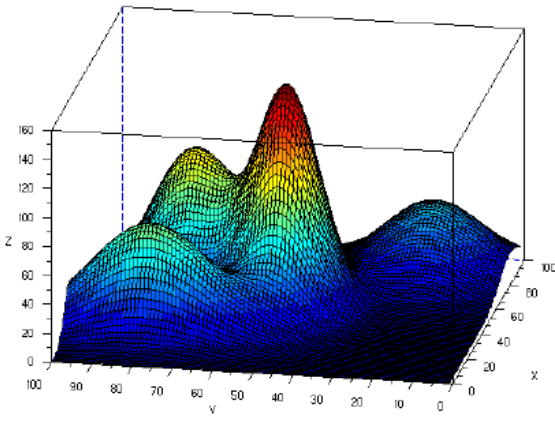


Gaussian mixture models as kernel density estimator

This can be used to develop an algorithm for directly approximating P(X) from data

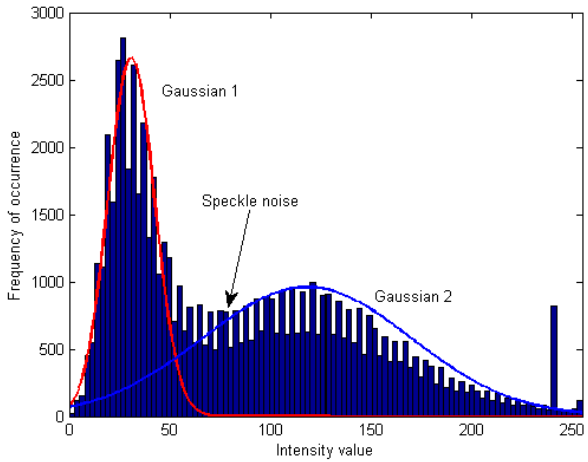


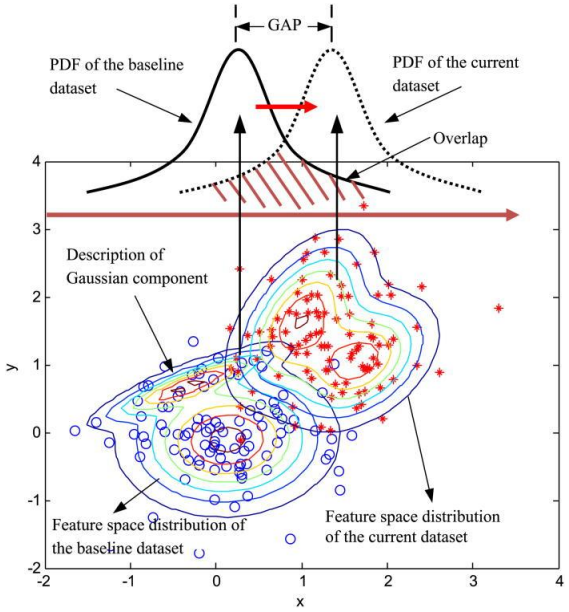


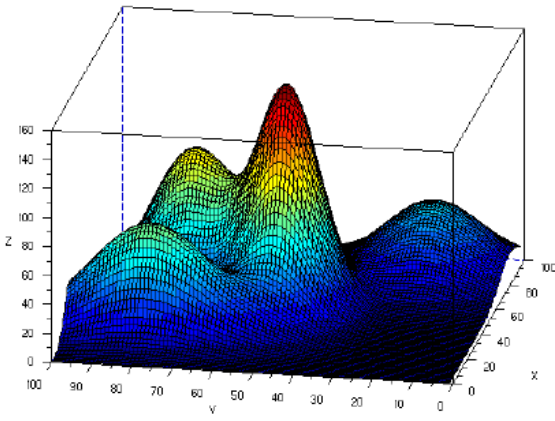


Gaussian mixture models as kernel density estimator

The parameters of GMM are usually estimated by MLE or MAP, by considering log $P(\{x_i\}|gmm)$ (see Advanced in the Probability slides)



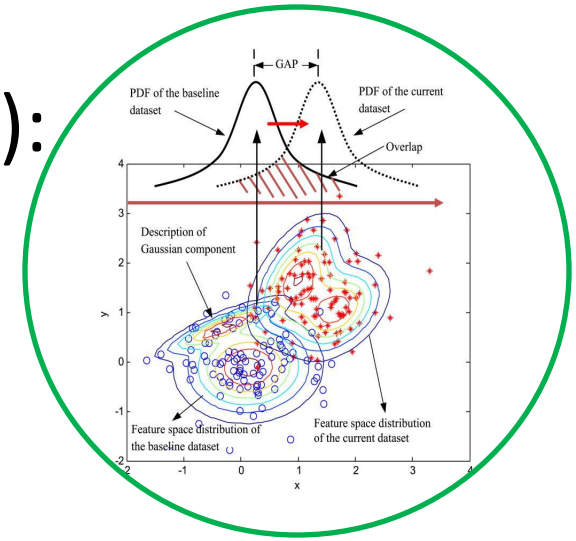




Prediction using GMM

GMM also can be used for prediction. Once a GMM distribution density $P_{gmm}(X|c)$ has been learned, one can predict class (as well as the true posterior probabilities of classes in fact) for any new x by using the Bayes rule ($\pi(c)$ are called prior belief/probability of c, see Advanced in Probability lecture):

$$P(c \mid x) = \frac{P(x \mid c)\pi(c)}{\sum_{c} P(x \mid c)\pi(c)}$$



Indicator matrix for categorical variables in models

Categorical variables can be represented in 3 different ways:

- **Direct form** - values from a list of options *city*={*Izmir*, *Istanbul*, *Ankara*}
- **Integer code** – an integer index indicating the position of option in a list: city={1:*Izmir*, 2:*Istanbul*, 3:*Ankara*}
- **Indicator matrix** aka **dummy binary variables:** *X1*={1|0 – *Izmir*}, *X2*={1|0 – *Istanbul*}, *X3*={1|0 – *Ankara*}

Example indicator matrix for city variable:

	Izmir	Istanbul	Ankara
X1	1	0	0
X2	0	1	0
X3	0	0	1

Optimal Bayes classifier

- Bayes classifier works by directly estimating the posterior probabilities of class variable c given input x and a data model P(x,c).
- By Bayes theorem,

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{\sum_{c} P(x \mid c)P(c)}$$

so if the generative model for P(x|c) is known, as is typically the case, P(c|x) can be found.

Optimal Bayes classifier

- Bayes classifier selects the class with the largest posterior probability P(c|x) in the MAP sense – easily corresponding to selecting argmax P(c|x) in the optimal classification rule in statistical decision theory considered in the class.