COMS 4771
Machine Learning

Nakul Verma

Machine learning: what?

Study of making machines **learn** a concept ***without*** having to explicitly program it.

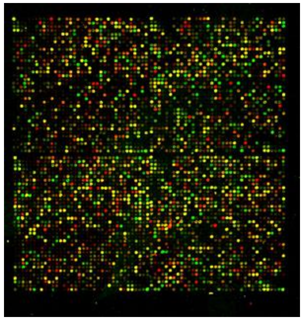
- Constructing algorithms that can:
  - learn from input data, and be able to make predictions.
  - find interesting patterns in data.
- Analyzing these algorithms to understand the limits of ‘learning’

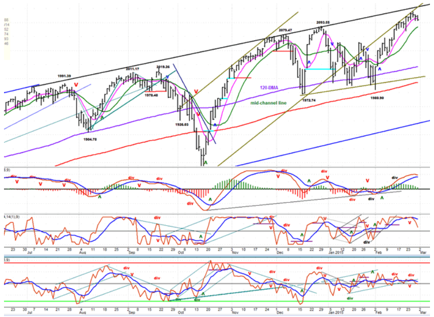
Machine learning: why?

We are smart programmers, why can't we just write some code with a set of rules to solve a particular problem?

Write down a set of rules to code to distinguish these two faces:

What if we don't even know the explicit task we want to solve?





Machine learning: problems in the real world

- Recommendation systems (Netflix, Amazon, Overstock)
- Stock prediction (Goldman Sachs, Morgan Stanley)
- Risk analysis (Credit card, Insurance)
- Face and object recognition (Cameras, Facebook, Microsoft)
- Speech recognition (Siri, Cortana, Alexa, Dragon)
- Search engines and content filtering (Google, Yahoo, Bing)

Machine learning: how?

so…. how do we do it?

This is what we will focus on in this class!

This course

We will learn:

- Study a prediction problem in an **abstract manner** and come up with a solution which is **applicable to many problems** simultaneously.

- Different types of paradigms and algorithms that have been successful in prediction tasks.

- How to systematically **analyze** how good an algorithm is for a prediction task.

Let’s get started!

Machine Learning: the basics…

A closer look at some prediction problems…

- Handwritten character recognition:

{ 0, 1, 2, …, 9 }

- Spam filtering:

murdered during the recent civil war in Libya in March 2011, before his death my late father was a strong supporter and a member of late Moammar Gadhafi Government in Tripoli. Meanwhile before the incident, my late Father came to Cotonou Benin republic with the sum of USD4, 200,000.00 (US$4.2M) which he deposited in a Bank here in Cotonou Benin Republic West Africa for safe keeping.

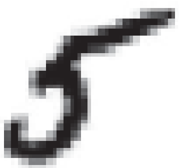
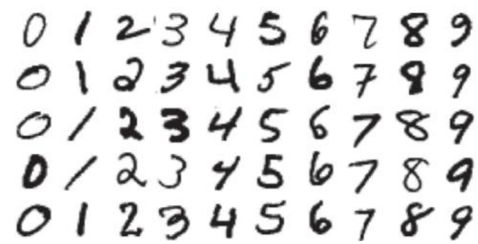
{ spam, not spam }

- Object recognition:

{ building, tree, car, road, sky,... }

Machine Learning: the basics...

Commonalities in a prediction problem:

Input:  = $\vec{x}$ = $\begin{pmatrix} x^{(1)} \\ \vdots \\ x^{(d)} \end{pmatrix} \in \mathcal{X}$ 

To learn: f

Output: **5** = $y \in \mathcal{Y}$ `{ 0, 1, 2, …, 9 }`

Machine Learning: the basics...

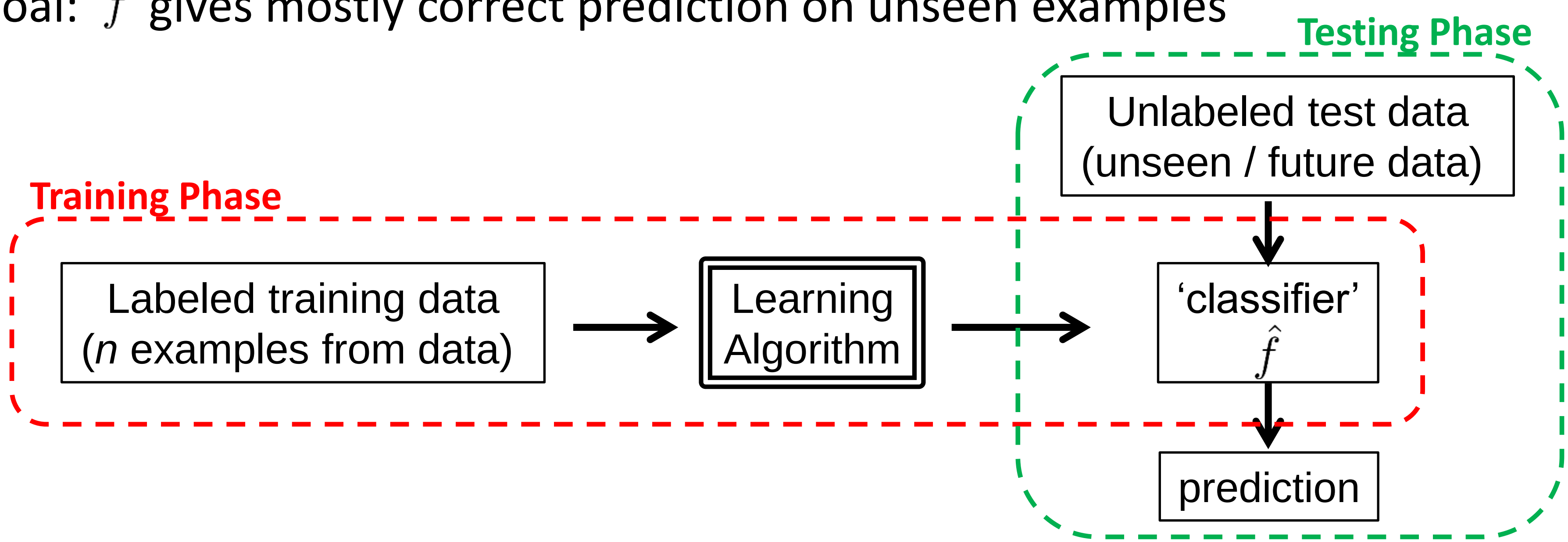
Supervised learning

Data: $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots \in \mathcal{X} \times \mathcal{Y}$

Assumption: there is a (relatively simple) function $f^* : \mathcal{X} \to \mathcal{Y}$
such that $f^*(\vec{x}_i) = y_i$ for most *i*

Learning task: given *n* examples from the data, find an approximation $\hat{f} \approx f^*$

Goal: $\hat{f}$ gives mostly correct prediction on unseen examples



Machine Learning: the basics…

Unsupervised learning

Data: $\vec{x}_1, \vec{x}_2, \ldots \in \mathcal{X}$

Assumption: there is an underlying structure in $\mathcal{X}$

Learning task: discover the structure given *n* examples from the data

Goal: come up with the summary of the data using the discovered structure

More later in the course…

Supervised Machine Learning

Statistical modeling approach:

Labeled training data
(*n* examples from data) → Learning Algorithm → 'classifier' $\hat{f}$

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots (\vec{x}_n, y_n)$$

drawn **independently** from
a fixed underlying distribution
(also called the *i.i.d.* assumption)

select $\hat{f}$ from…?

from a pool of ***models*** $\mathcal{F}$

that **maximizes**
label agreement of the
training data

How to select $\hat{f} \in \mathcal{F}$?

- Maximum likelihood (best fits the data)
- Maximum a posteriori (best fits the data but incorporates prior assumptions)
- Optimization of 'loss' criterion (best discriminates the labels)
- …

The Classifier

ok, so given a joint input/output space $\mathcal{X} \times \mathcal{Y}$

where the data is distributed according to the distribution $\mathcal{D}(\mathcal{X} \times \mathcal{Y})$

a classifier is simply a (measureable) function of the type $f : \mathcal{X} \to \mathcal{Y}$

there are plenty of functions of such kind, so which *f* to pick?

Examples

- $f_1(\vec{x}) := y$ for some fixed $y \in \mathcal{Y}$ for all inputs $\vec{x} \in \mathcal{X}$ *'Constant' classifier*

- $f_2(\vec{x}) := \begin{cases} 0 & \textit{if } x \geq 5 \\ 1 & \textit{otherwise} \end{cases}$ *say* $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \{0, 1\}$ *'threshold' classifier*

- $f_3(\vec{x}) := \underset{y \in \mathcal{Y}}{\arg\max}\ P[Y = y]$ *'Majority class' classifier*

The Classifier

So… there are plenty of functions $f : \mathcal{X} \rightarrow \mathcal{Y}$ to choose from.

What might be a reasonable/good function *f* ?

Afterall, we want to do prediction. It better be the case that we select an *f* which does well in terms of prediction.

Ideally we want: on any example pair (x,y) from *D* we want *f* such that *f*(x) = y

That is, define **accuracy of a classifier *f*, *ie* acc(*f*)** as

$$\mathrm{acc}(f) := P_{(\vec{x},y)}\left[f(\vec{x}) = y\right] \quad = \quad \mathbb{E}_{(\vec{x},y)}\Big[\mathbf{1}\left[f(\vec{x}) = y\right]\Big]$$

we want a classifier *f* which maximizes acc.

A Good Classifier

Let's study the following classifier

$$f(\vec{x}) = \arg\max_{y \in \mathcal{Y}} \; P[Y = y | X = \vec{x}]$$

aka 'Bayes' classifier

this classifier is remarkably good!

Why the particular f = argmax$_y$ P[Y|X] ?

Accuracy of a classifier f : $$P_{(\vec{x},y)}\left[f(\vec{x})=y\right] \;=\; \mathbb{E}_{(\vec{x},y)}\left[\mathbf{1}\left[f(\vec{x})=y\right]\right]$$

Assume binary classification (for simplicity) : $$\mathcal{Y}=\{0,1\}$$

Let: $$f(\vec{x})=\arg\max_{y\in\{0,1\}} P[Y=y|X=\vec{x}]$$ Bayes classifier

$$g(\vec{x})=\mathcal{X}\rightarrow\{0,1\}$$ any classifier

Theorem: $$P_{(\vec{x},y)}\left[g(\vec{x})=y\right] \;\leq\; P_{(\vec{x},y)}\left[f(\vec{x})=y\right]$$

!!! Bayes classifier is optimal !!!

Optimality of Bayes classifier

Theorem: $P_{(\vec{x},y)}\big[g(\vec{x}) = y\big] \;\leq\; P_{(\vec{x},y)}\big[f(\vec{x}) = y\big]$

Observation: Fix any $\vec{x} \in \mathcal{X}$, then for any classifier *h*

$$
\begin{aligned}
P\big[h(\vec{x}) = y | X = \vec{x}\big] &= P\big[h(\vec{x}) = 1, Y = 1 | X = \vec{x}\big] + P\big[h(\vec{x}) = 0, Y = 0 | X = \vec{x}\big] \\
&= \mathbf{1}[h(\vec{x}) = 1] \cdot P[Y = 1 | X = \vec{x}] + \mathbf{1}[h(\vec{x}) = 0] \cdot P[Y = 0 | X = \vec{x}] \\
&= \mathbf{1}[h(\vec{x}) = 1]\eta(\vec{x}) + \mathbf{1}[h(\vec{x}) = 0]\big(1 - \eta(\vec{x})\big)
\end{aligned}
$$

$$\Big\{\ \eta(\vec{x}) := P[Y = 1 | X = \vec{x}]\ \Big\}$$

So:

$$
\begin{aligned}
&P\big[f(\vec{x}) = y | X = \vec{x}\big] - P\big[g(\vec{x}) = y | X = \vec{x}\big] \\
&\quad = \eta(\vec{x})\Big[\mathbf{1}[f(\vec{x}) = 1] - \mathbf{1}[g(\vec{x}) = 1]\Big] + \big(1 - \eta(\vec{x})\big)\Big[\mathbf{1}[f(\vec{x}) = 0] - \mathbf{1}[g(\vec{x}) = 0]\Big] \\
&\quad = \big(2\eta(\vec{x}) - 1\big)\Big[\mathbf{1}[f(\vec{x}) = 1] - \mathbf{1}[g(\vec{x}) = 1]\Big] \\
&\quad \geq 0 \qquad \text{By the choice of } f
\end{aligned}
$$

Integrate over X to remove the conditional ∎

So… is classification a solved problem?

We know that Bayes classifier is optimal.

So have we solved all classification problems?

$\boxed{\text{Not even close!}}$

Why?

$$f(\vec{x}) = \arg\max_{y \in \mathcal{Y}} \; P[Y = y | X = \vec{x}]$$

The true data distribution is unknown,
so P[Y|X] is unknown

- Since P[Y|X] is unknown, need to **estimate** it from observed data samples
- What about the quality of this estimation?
- There are numerous x $\in$ *X* (usually infinite), so how many P[Y|X=x] do we need to construct a practically viable approximation to the Bayes classifier?

Estimating Bayes classifier from observed data

We know there are numerous P[Y|X] (one for each X=x), but observe

$$f(\vec{x}) = \arg\max_{y \in \mathcal{Y}} \; P[Y = y | X = \vec{x}]$$

$$= \arg\max_{y \in \mathcal{Y}} \; \frac{P[X = \vec{x} | Y = y] \cdot P[Y = y]}{P[X = \vec{x}]}$$

Bayes rule

indep. of y

$$= \arg\max_{y \in \mathcal{Y}} \; P[X = \vec{x} | Y = y] \; \cdot \; P[Y = y]$$

Class conditional probability model

Class Prior

- We STILL need to estimate P[X|Y] and P[Y] from data, but the number of distributions that need to be estimated is small, since |Y| is tiny (usually 2)

Estimating Densities

Ok so how do we estimate conditional densities like P[X|Y] or densities (like P[X] or P[Y] in general?

Multiple ways of estimating densities

- If the particular form of the data-density is known/assumed, we can do some sort of parametric density estimation

  [e.g. likelihood estimation, max a-postirori estimation, mixture models, etc.]

- If one doesn't want to/ if it is inappropriate to assume a specific distribution, we can do non-parametric density estimation

  [e.g. kernel density estimation, parzen windows, histograms, etc.]

(each technique has its pros and cons)

Maximum Likelihood Estimation (MLE)

Given some data $\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n \in \mathcal{X}$ i.i.d.

Say we have a model class $\mathcal{P} = \{p_\theta \mid \theta \in \Theta\}$

ie, each model *p* can be described by a set of parameters θ

find the parameter settings θ that **best fits** the data.

If each model *p*, is a **probability model** then we can find the best fitting probability model via the ***likelihood estimation***!

Likelihood $\mathcal{L}(\theta|X) := P(X|\theta) = P(\vec{x}_1, \ldots, \vec{x}_n|\theta) \overset{i.i.d.}{=} \prod_{i=1}^{n} P(\vec{x}_i|\theta) = \prod_{i=1}^{n} p_\theta(\vec{x}_i)$

Interpretation: How **probable** (or how likely) is the data given the model p_θ?

Parameter setting θ that maximizes $\mathcal{L}(\theta|X)$

$$\arg\max_\theta \mathcal{L}(\theta|X) = \arg\max_\theta \prod_{i=1}^{n} p_\theta(\vec{x}_i)$$

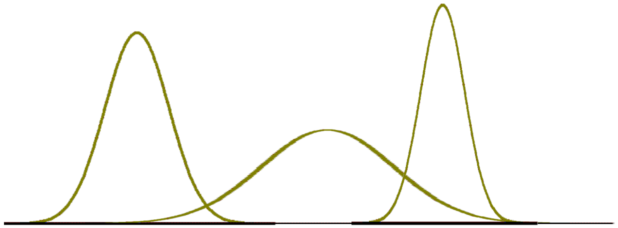
MLE Example

Fitting a statistical probability model to heights of females

Height data (in inches): 60, 62, 53, 58, … $\in$ **R**

$$x_1, x_2, \ldots x_n \in \mathcal{X}$$

Model class: Gaussian models in **R**



$$p_\theta(x) = p_{\{\mu,\sigma^2\}}(x) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right)$$

μ = mean parameter

σ^2 = variance parameter > 0

So, what is the MLE for the given data X ?

MLE Example (contd.)

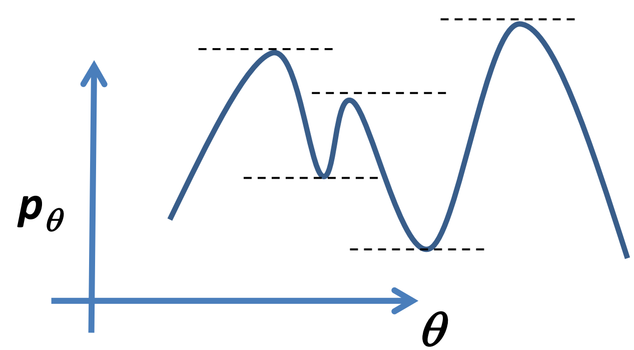
Height data (in inches): $x_1, x_2, \ldots x_n \in \mathcal{X}$ = **R**

Model class: Gaussian models in **R** $p_{\{\mu,\sigma^2\}}(x) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

MLE: $$\arg\max_{\theta} \mathcal{L}(\theta|X) = \arg\max_{\mu,\sigma^2} \prod_{i=1}^{n} p_{\{\mu,\sigma^2\}}(x_i)$$ *Good luck!*

Trick #1: $$\arg\max_{\theta} \mathcal{L}(\theta|X) = \arg\max_{\theta} \log \mathcal{L}(\theta|X)$$ *"Log" likelihood*

Trick #2: finding max (or other extreme values) of a function is simply analyzing the '**stationary points**' of a function. That is, values at which the **derivative** of the function is zero !



MLE Example (contd. 2)

Let's calculate the best fitting $\theta = \{\mu, \sigma^2\}$

$$\begin{aligned}
\arg\max_{\theta} \mathcal{L}(\theta|X) &= \arg\max_{\theta} \; \log \mathcal{L}(\theta|X) \\
&= \arg\max_{\mu,\sigma^2} \; \log \left(\prod_{i=1}^{n} p_{\{\mu,\sigma^2\}}(x_i) \right) \\
&= \arg\max_{\mu,\sigma^2} \; \sum_{i=1}^{n} \log \left(p_{\{\mu,\sigma^2\}}(x_i) \right) \\
&= \arg\max_{\mu,\sigma^2} \; \sum_{i=1}^{n} \Bigg[\underbrace{-\frac{1}{2}\log(2\pi\sigma^2) - \frac{(x_i-\mu)^2}{2\sigma^2}}_{g_i(\mu,\sigma^2)} \Bigg]
\end{aligned}$$

"Log" likelihood

i.i.d.

Maximizing μ :

$$0 = \nabla_\mu \left(\sum_{i=1}^{n} g_i(\mu, \sigma^2) \right) \implies \mu_{\mathrm{ML}} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Maximizing σ^2:

$$\sigma^2_{\mathrm{ML}} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2$$

MLE Example

So, the best fitting **Gaussian model** $p_{\{\mu,\sigma^2\}}(x) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right)$

Female height data: 60, 62, 53, 58, … $\in$ **R**

$$x_1, x_2, \ldots x_n \in \mathcal{X}$$

Is the one with parameters: $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$

What about other model classes?

Other popular probability models

Bernoulli model (coin tosses) *Scalar valued*

Multinomial model (dice rolls) *Scalar valued*

Poisson model (rare counting events) *Scalar valued*

Gaussian model (most common phenomenon) *Scalar valued*

$\boxed{\text{Most machine learning data is vector valued!}}$

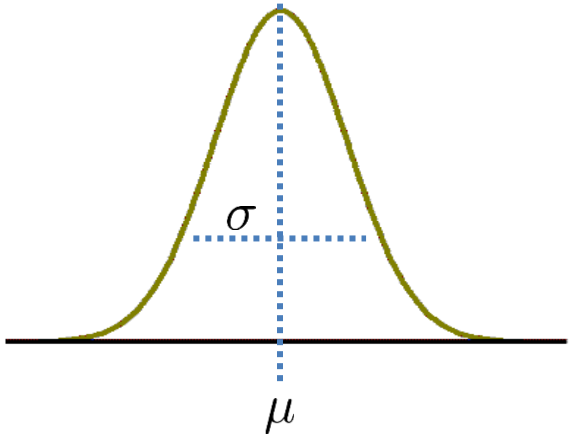
Multivariate Gaussian Model ***Vector valued***

Multivariate version available of other scalar valued models

Multivariate Gaussian

Univariate **R**

$$p_{\{\mu,\sigma^2\}}(x) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right)$$



μ = mean parameter

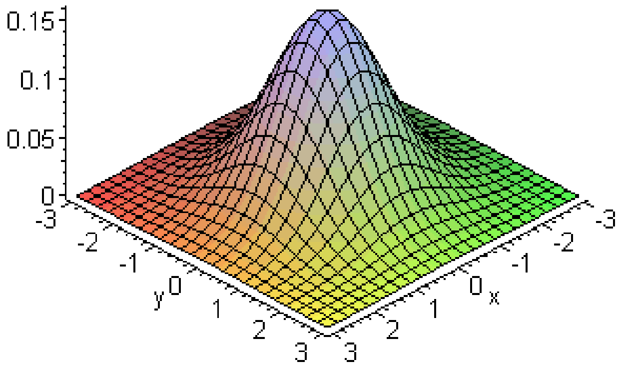
σ^2 = variance parameter > 0

Multivariate **R**^{d}

$$p_{\{\vec{\mu},\Sigma\}}(\vec{x}) := \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^{\mathsf{T}}\Sigma^{-1}(\vec{x}-\vec{\mu}) \right)$$

$\vec{\mu}$ = mean vector

Σ = Covariance matrix (positive definite)



Back to Classification

So how can we construct a practical classifier inspired from Bayes classifier?

$$f(\vec{x}) = \arg\max_{y \in \mathcal{Y}} \; P[Y = y | X = \vec{x}]$$

$$= \arg\max_{y \in \mathcal{Y}} \; \underbrace{P[X = \vec{x} | Y = y]}_{\text{Class conditional probability model}} \cdot \underbrace{P[Y = y]}_{\text{Class Prior}}$$

Class prior:

Since Y is finite, we can model P[Y] as a generalized Bernoulli distribution and estimate the parameters from observed data using MLE!

Class conditional:

Use a separate probability model P[X|Y=y] for individual categories/class-type y

Again, we can find the appropriate parameters for the models using MLE!

Classification via MLE Example

Task: learn a classifier to identify an individual as **`male`** or **`female`** based on collecting, say, just the height and weight measurements

$$\mathcal{X} = \mathbf{R}^2 = \text{Weight} \times \text{Height}$$
$$\mathcal{Y} = \{\text{male}, \text{female}\}$$

For a certain population of interest, the data may be distributed according to

$$\mathcal{D} = \mathcal{D}(\mathcal{X} \times \mathcal{Y})$$

which is unknown, but fixed

Observation: The following labelled data is collected/observed

(x_1, y_1) (x_2, y_2) (x_3, y_3) (x_4, y_4)

((110,5.2),f), ((185,5.11),m), ((220,6.2),m), (200,5.5),f), … $\sim_{\text{i.i.d.}} \mathcal{D}(\mathcal{X} \times \mathcal{Y})$

Classification via MLE Example

For this task, the Bayes classifier would be

$$f(\vec{x}) = \arg\max_{y \in \{\text{male},\text{female}\}} P_{\mathcal{D}|_{X|Y=y}}[X = \vec{x}|Y = y] \cdot P_{\mathcal{D}|_Y}[Y = y]$$

[Note: *D* is unknown, hence each density needs to be estimated]

Using **labelled** training data, learn all the parameters:

Estimating the **class priors**:

$\hat{P}[Y = \text{male}] =$ fraction of training data labelled as male

$\hat{P}[Y = \text{female}] =$ fraction of training data labelled as female

[using Binomial model; parameter estimation via MLE]

Estimating the **class conditionals**:

$$\hat{P}[X|Y = \text{male}] = p_{\theta(\text{male})}(X)$$

$$\hat{P}[X|Y = \text{female}] = p_{\theta(\text{female})}(X)$$

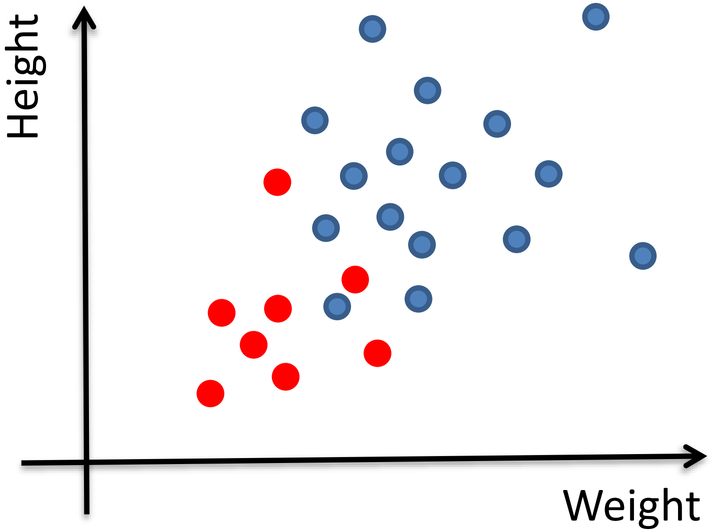
θ(male) = **MLE** using only male data

θ(female) = **MLE** using only female data

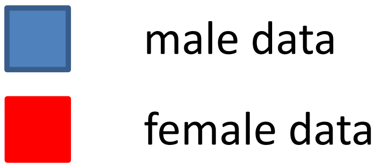
[by modelling each P[X|Y] using, say, Bivariate Gaussian; parameter estimation via MLE]

What are we doing geometrically?

Data geometry:



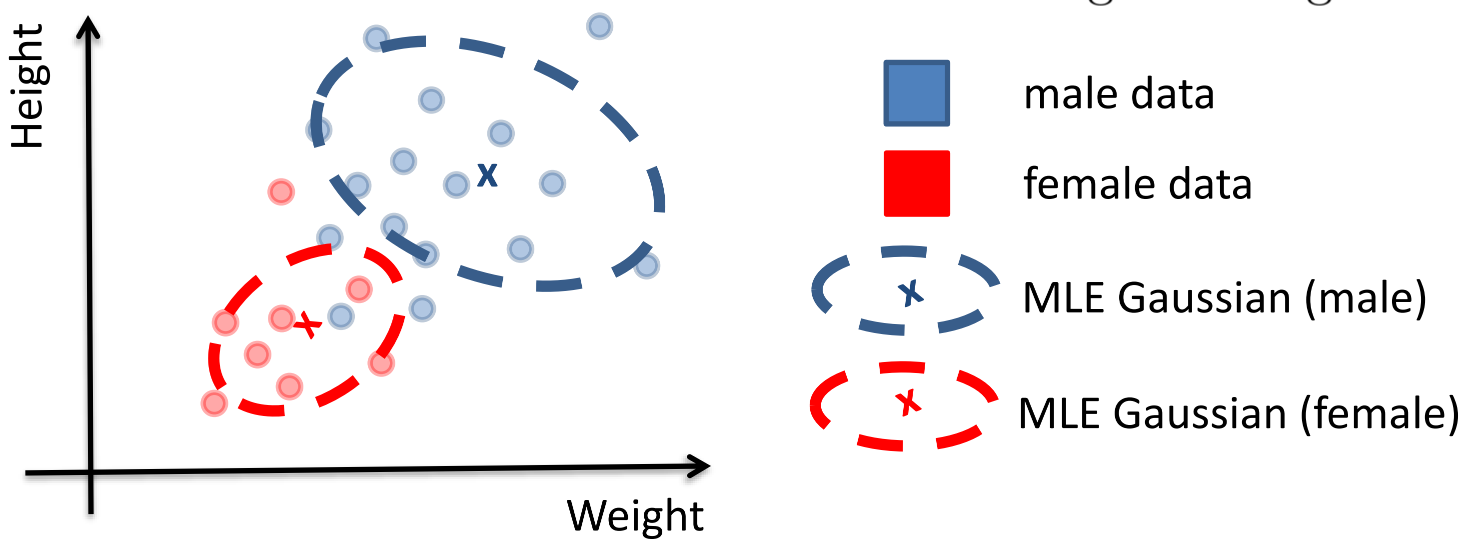
$$\mathcal{X} = \mathbf{R}^2 = \text{Weight} \times \text{Height}$$

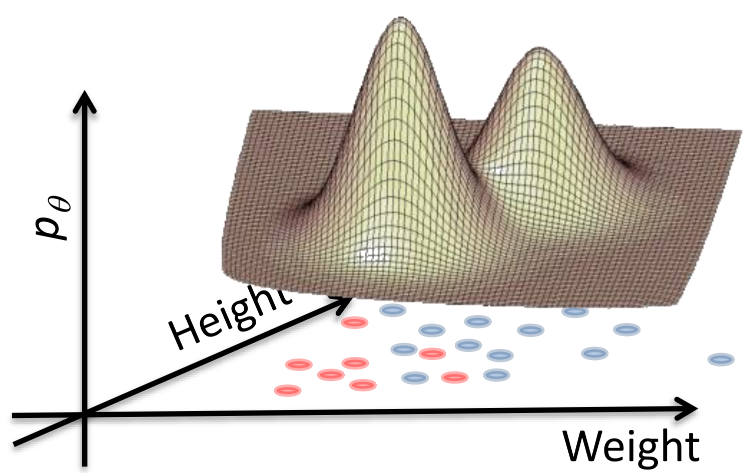


What are we doing geometrically?

Data geometry:

$$\mathcal{X} = \mathbf{R}^2 = \text{Weight} \times \text{Height}$$





Classification via MLE Example

Task: learn a classifier to distinguish **`males`** from **`females`**

based on say height and weight measurements

Classifier: $$f(\vec{x}) = \underset{y \in \{\text{male},\text{female}\}}{\arg\max} \; P[X = \vec{x} | Y = y] \cdot P[Y = y]$$

Using **labelled** training data, learn all the parameters:

Learning **class priors**:

$P[Y = \text{male}] =$ fraction of training data labelled as male

$P[Y = \text{female}] =$ fraction of training data labelled as male

Learning **class conditionals**:

$$P[X | Y = \text{male}] = p_{\theta(\text{male})}(X)$$

$$P[X | Y = \text{female}] = p_{\theta(\text{female})}(X)$$

θ(male) = **MLE** using only male data

θ(female) = **MLE** using only female data

Classification via MLE Example

$\boxed{\text{We just made our first predictor } f \text{ !}}$

Classification via Prob. Models: Variation

Naïve Bayes classifier:

$$\hat{f}(\vec{x}) = \underset{y \in \mathcal{Y}}{\arg\max} \;\; P[X = \vec{x} | Y = y] \cdot P[Y = y]$$

$$= \underset{y \in \mathcal{Y}}{\arg\max} \;\; \prod_{j=1}^{d} P[X^{(j)} = x^{(j)} | Y = y] \cdot P[Y = y]$$

$$\vec{x} = \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(d)} \end{bmatrix}$$

Naïve Bayes assumption: The individual features/measurements are **independent** given the class label

Advantages:

Computationally very simple model. Quick to code.

Disadvantages:

Does not properly capture the interdependence between features, giving bad estimates.

How to evaluate the quality of a classifier?

Your friend claims: “My classifier is better than yours”

How can you evaluate this statement?

Given a classifier *f* , we essentially need to compute:

$$P_{(\vec{x},y)}\left[f(\vec{x}) = y\right] \;=\; \mathbb{E}_{(\vec{x},y)}\left[\mathbf{1}\left[f(\vec{x}) = y\right]\right]$$

Accuracy of *f*

But… we **don’t know** the underlying distribution

We can use **training data** to estimate…

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}[f(\vec{x}_i) = y_i]$$

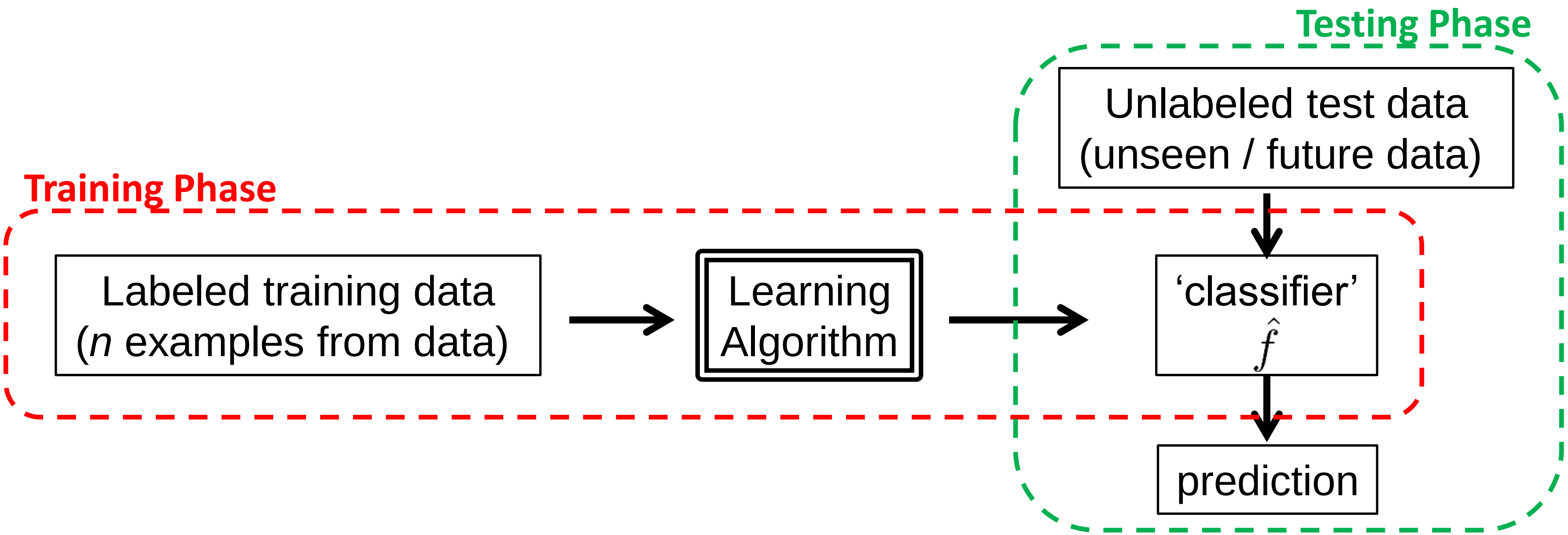
Severely overestimates the accuracy!

Why? Training data is already used to construct *f*, so it is NOT an unbiased estimator

How to evaluate the quality of a classifier?

General strategy:

- Divide the labelled data into **training** and **test** FIRST
- Only use the training data for learning f
- Then the test data can be used as an **unbiased estimator** for gauging the predictive accuracy of f



What we learned…

- Why machine learning
- Basics of Supervised Learning
- Maximum Likelihood Estimation
- Learning a classifier via probabilistic modelling
- Optimality of Bayes classifier
- Naïve Bayes classifier
- How to evaluate the quality of a classifier

Questions?

Next time…

Direct ways of finding the discrimination boundary