

COMS 4771

Introduction to 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’

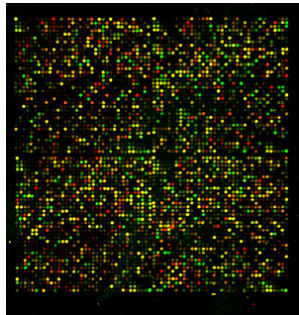
Machine learning: why?

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.

Prerequisites

Mathematical prerequisites

- Basics of probability and statistics
- Linear algebra
- Calculus

Computational prerequisites

- Basics of algorithms and datastructure design
- Ability to program in a high-level language.

Administrivia

Website:

`http://www.cs.columbia.edu/~verma/classes/ml/`

The team:

Instructor: Nakul Verma (me)

TAs

Students: you!

Evaluation:

- Homeworks (40%)
- Exam 1 (30%)
- Exam 2 (30%)

Policies

Homeworks:

- No late homework
 - **Must** type your homework (no handwritten homework)
 - Please include your name and UNI
 - Submit a pdf copy of the assignment via gradescope
-
- We encourage discussing the problems (piazza/groups/etc), but please don't copy.

Announcement!

- Visit the course website
- Review the basics (prerequisites)
- HW0 is out!
- Sign up on Piazza & Gradescope

Let's get started!

Machine Learning: the basics...

A closer look at some prediction problems...

- Handwritten character recognition:

5 6 7 8

{ 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 }

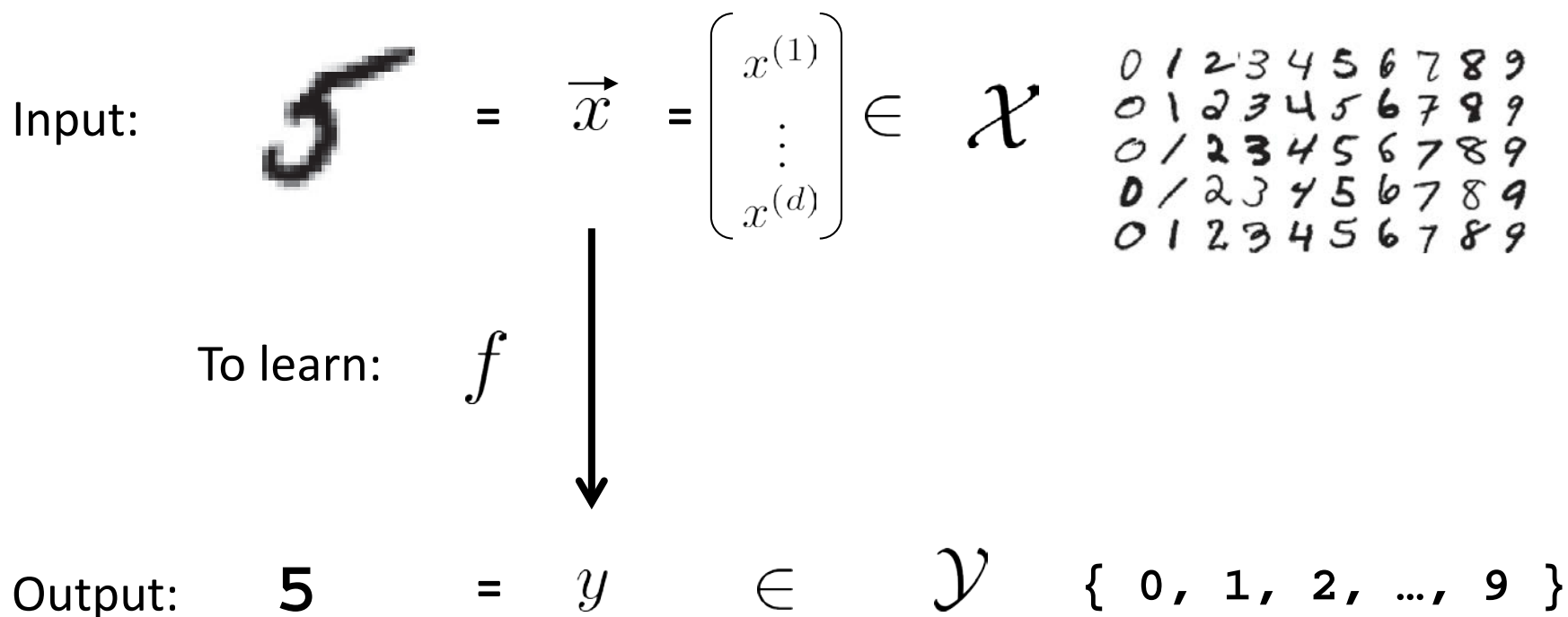
- Object recognition:



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

Machine Learning: the basics...

Commonalities in a prediction problem:



Machine Learning: the basics...

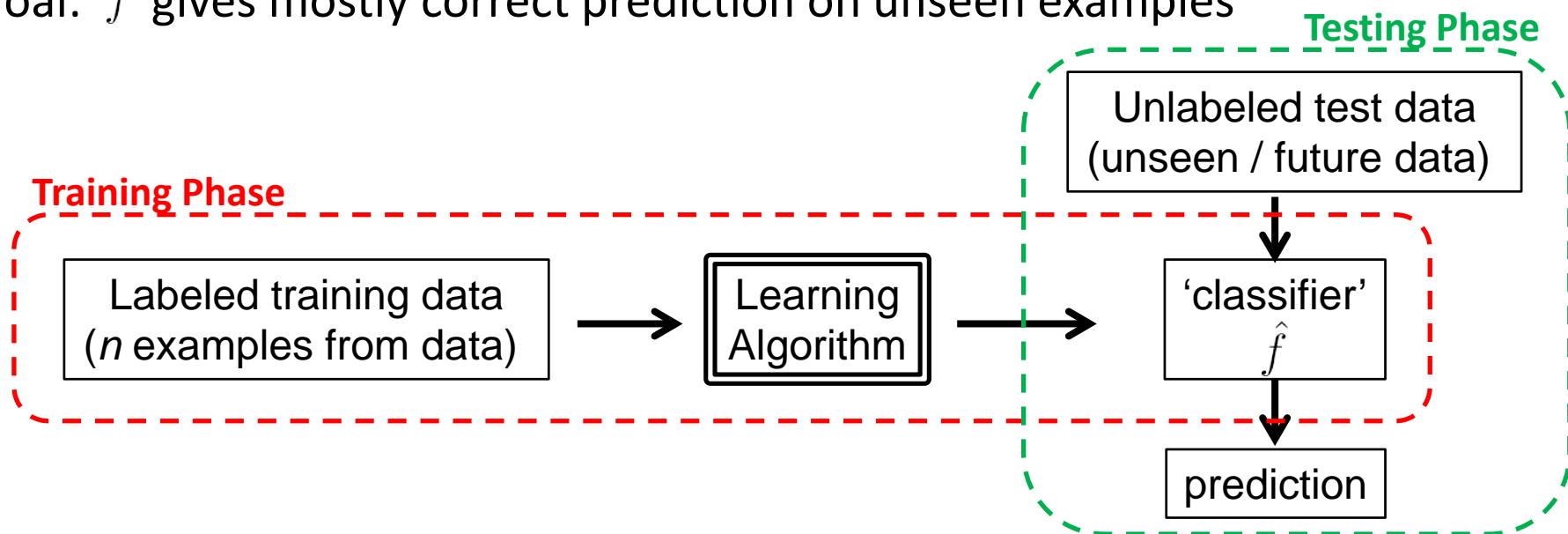
Data: $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots \in \mathcal{X} \times \mathcal{Y}$

Supervised learning

Assumption: there is a (relatively simple) function $f^* : \mathcal{X} \rightarrow \mathcal{Y}$
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...

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

Unsupervised learning

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}

$(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_n, y_n)$
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)
- ...

Maximum Likelihood Estimation (MLE)

Given some data $\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n \in \mathcal{X}$ i.i.d. *(Let's forget about the labels for now)*

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***!

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

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

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

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

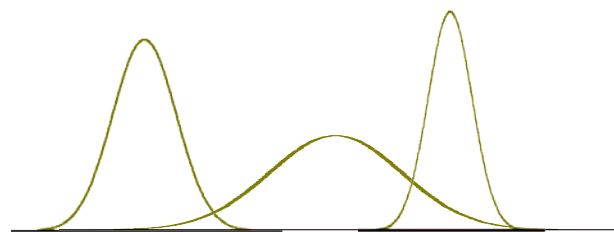
MLE Example

Fitting a statistical probability model to heights of females

Height data (in inches): $60, 62, 53, 58, \dots \in \mathbf{R}$

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

Model class: Gaussian models in \mathbf{R}



$$p_{\theta}(x) = p_{\{\mu, \sigma^2\}}(x) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) \quad \begin{array}{l} \mu = \text{mean parameter} \\ \sigma^2 = \text{variance parameter} > 0 \end{array}$$

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

MLE Example (contd.)

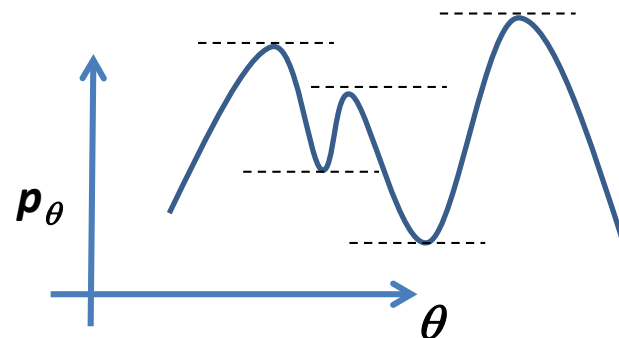
Height data (in inches): $x_1, x_2, \dots, x_n \in \mathcal{X} = \mathbf{R}$

Model class: Gaussian models in \mathbf{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 $\theta = \{\mu, \sigma^2\}$

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

Maximizing μ :

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

Maximizing σ^2 :

$$\sigma_{\text{ML}}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

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, ... $\in \mathbf{R}$

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

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

Most machine learning data is vector valued!

Multivariate Gaussian Model

Vector valued

Multivariate version available of other scalar valued models

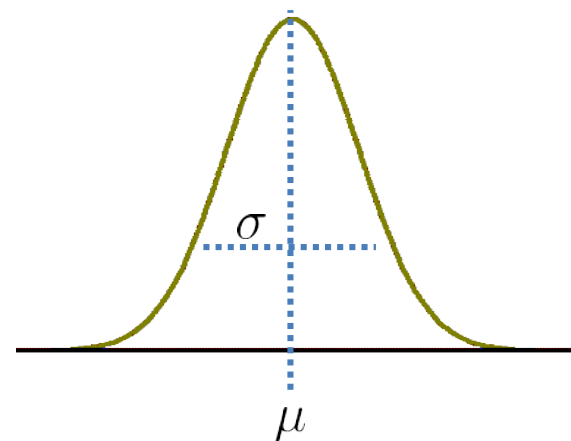
Multivariate Gaussian

Univariate \mathbf{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

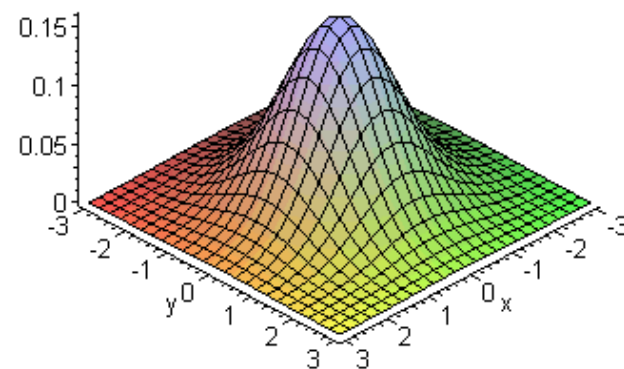


Multivariate \mathbf{R}^d

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

$\vec{\mu}$ = mean vector

Σ = Covariance matrix (positive definite)



From MLE to Classification

MLE sounds great, how do we use it to do **classification** using **labelled** data?

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

Why? More later...

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

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

Class prior:

Simply the probability of data sample occurring from a category

Class conditional:

Use a separate probability model individual categories/class-type

We can find the appropriate parameters for the model using MLE!

Classification via MLE Example

Task: learn a classifier to distinguish **males** from **females**
based on say height and weight measurements

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

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

Learning **class priors**:

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

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

Learning **class conditionals**:

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

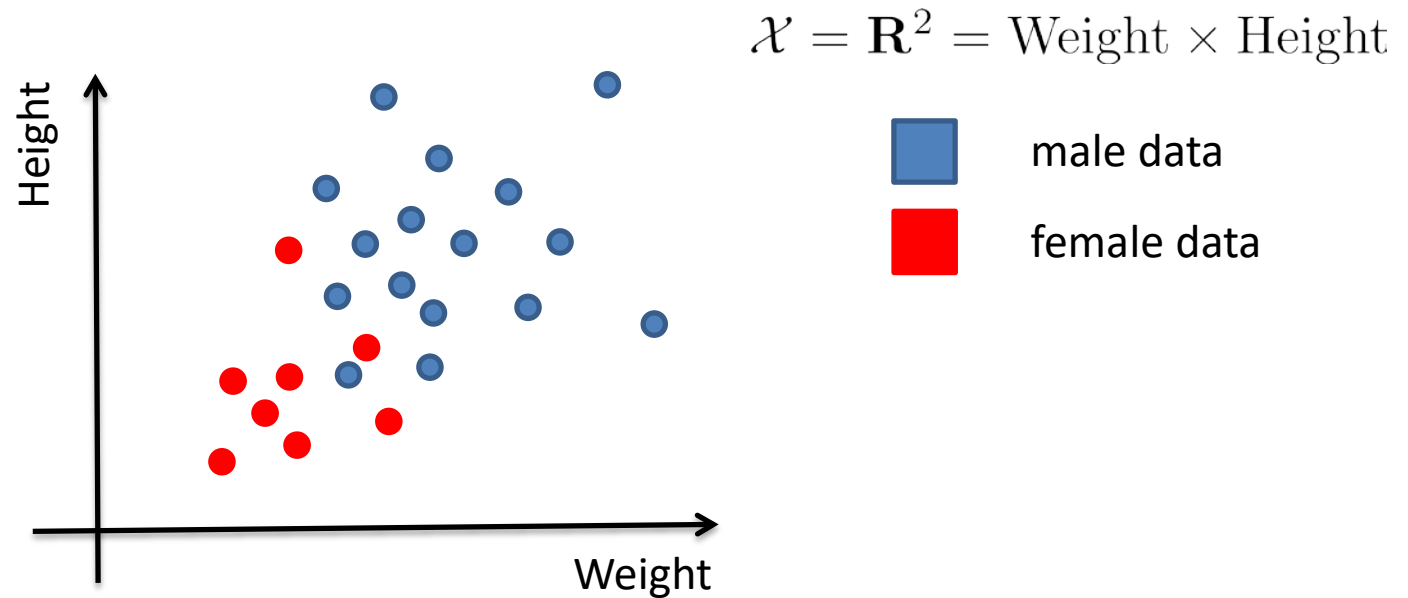
$$P[X | Y = \text{female}] = p_{\theta(\text{female})}(X)$$

$\theta(\text{male})$ = **MLE** using only male data

$\theta(\text{female})$ = **MLE** using only female data

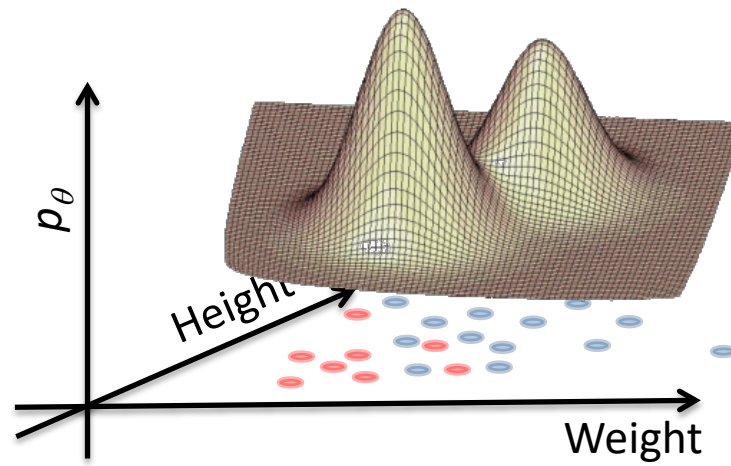
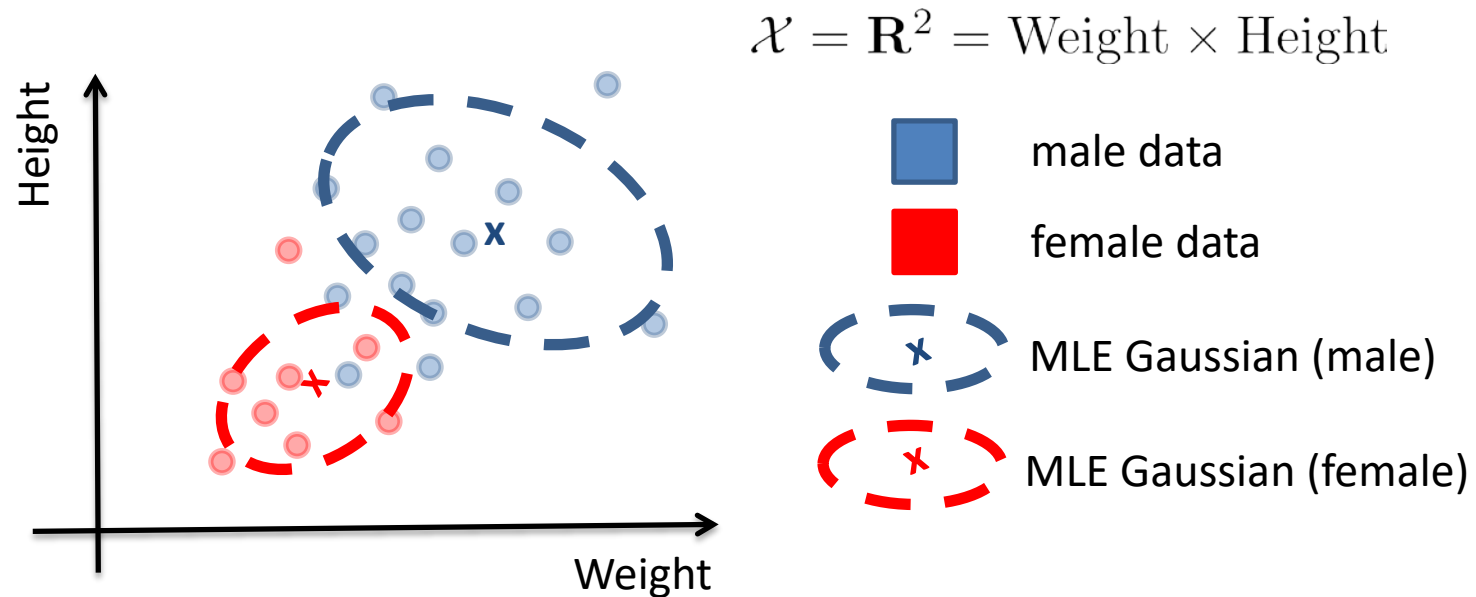
What are we doing geometrically?

Data geometry:



What are we doing geometrically?

Data geometry:



Classification via MLE Example

Task: learn a classifier to distinguish **males** from **females**
based on say height and weight measurements

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

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

Learning **class priors**:

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

Learning **class conditionals**:

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

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

$\theta(\text{male})$ = **MLE** using only male data

$\theta(\text{female})$ = **MLE** using only female data

Classification via MLE Example

We just made our first predictor \hat{f} !

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

Why the particular $f = \operatorname{argmax}_y P[Y|X]$?

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

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

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

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

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

!!! Bayes classifier is optimal !!!

Optimality of Bayes classifier

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

Observation: For any classifier h

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

So:

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

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?

Not even close!

Why?

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

How to estimate $P[Y|X]$?

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

How to estimate $P[X|Y]$?

- How good is the model class?
- Quality of estimation degrades with increase in the dimension of X !
- Active area of research!

Classification via Prob. Models: Variation

Naïve Bayes classifier:

$$\begin{aligned}\hat{f}(\vec{x}) &= \arg \max_{y \in \mathcal{Y}} P[X = \vec{x} | Y = y] \cdot P[Y = y] \\ &= \arg \max_{y \in \mathcal{Y}} \prod_{j=1}^d P[X^{(j)} = x^{(j)} | Y = y] \cdot P[Y = y]\end{aligned}\quad \vec{x} = \begin{pmatrix} x^{(1)} \\ \vdots \\ x^{(d)} \end{pmatrix}$$

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:

$$P_{(\vec{x}, y)} [f(\vec{x}) = y] = \mathbb{E}_{(\vec{x}, y)} [\mathbf{1}[f(\vec{x}) = y]] \quad \text{Accuracy of } f$$

But... we **don't know** the underlying distribution

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

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

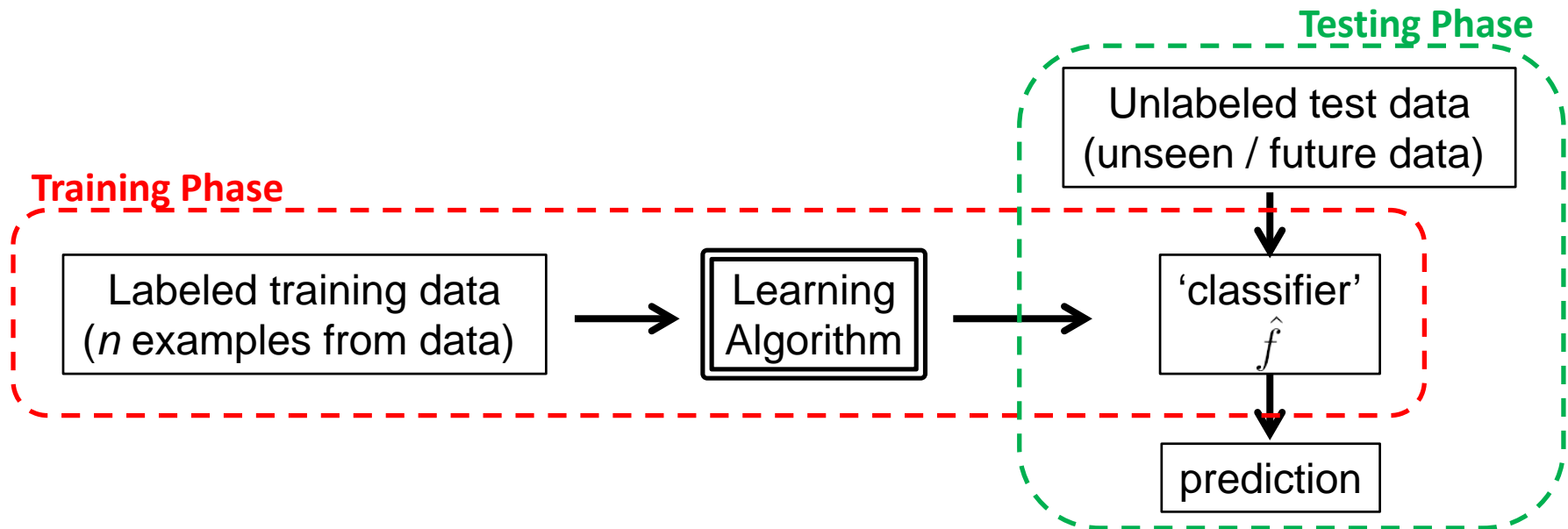
Severely overestimates the accuracy!

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

Remember

- Visit the course website

`http://www.cs.columbia.edu/~verma/classes/ml/`

- Review the basics (prerequisites)
- HW0 is out
- Sign up on Piazza & Gradescope