Introduction

Machine Learning Theory (MLT)
Edinburgh
Rik Sarkar

Machine Learning Theory – Course info

- Course code: INFR11202/INFR11224, Shorthand: MLT
  - Web page: https://opencourse.inf.ed.ac.uk/mlt
- Lecturer : Rik Sarkar (rik.sarkar@ed.ac.uk)
- Schedule
  - Tuesdays 16:10 – 18:00 (7 George Square)
- Three Tutorial sessions
  - Weeks 5, 8, 10 (may change)
- 1 Coursework 30% (written analysis, proofs) (Given Feb 16, Due march 13)
- 1 Exam (April/May) : 70%

Resources

- Book: Understanding Machine Learning: From Theory to Algorithms. Shai Shalev-Shwartz and Shai Ben-David
  - Available in library, Book retailers, free pdf Online
- Other notes and papers to be given out as we go.
- Exercises given in tutorials and notes.
- Piazza forum active
- Sample Exam: Last year’s exam available online
- Forms of feedback:
  - Coursework.
  - Tutorials: Attempt tutorial exercises. Attend tutorials. Ask questions.
  - Exercises in notes: Some exercises available in notes. Attempt them and check solutions.
  - Piazza available. Ask your questions!

What is machine learning

- What is learning?
- When is learning possible?
- When is learning needed?

- Do we need to “learn”
  - Tic Tac Toe?
  - The 2 times table upto 2x10?

Learning is useful when

- Available data is small compared to possible inputs/questions
  - If answers to all relevant questions are available, then it is just a matter of memorization
- Data possibly contains noise
- We have some idea (hypothesis class) of what the learned model could be

- Ideally the smaller quantity of data we learn from, the better
  - But what is the definition of “small”? How little data is sufficient?

Why theory

- We are interested in mathematics of ML
  - Define exactly what different metrics, models, methods are
  - Gain better understanding of their strengths and weaknesses – where they work where they do not. What is understood/not understood
- Do better ML in the future
  - Accuracy, generalization
  - Privacy
  - Fairness
  - Explainability
  - Other desirable properties….
- Similar to learning algorithms and data structures to improve programming

Have you taken an ML course before?

- Raise your hand if you have never taken an ML course
  - (I never studied formally till I started teaching this course, so don’t be shy!)

What the course is for

- Learn to precisely define and analyse ML models and algorithms
- Learn to think about and analyse their properties, know what they are good for
- Learn to read ML – the field is continuously evolving, it is not easy to read the latest paper
- Eventually develop better models, metrics and algorithms
- Make ML better in other ways beyond accuracy
- The course is suitable for two types of students
  - You have learned various ML models and algorithms in courses and would like to have a unified view and understand them at a deeper level
  - You have studied maths/stats and would like to know how to think about ML
- If neither description suits you

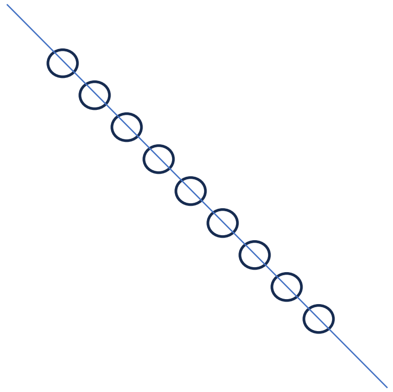
What to expect in the course

- Reading and writing precise definitions, using symbols and equations
  - In class, notes, possibly papers..
- Proofs. And intuitions
- We will **not** study new models, types of neural networks etc
- We will study ideas that broadly apply to all (or many) types of models
  - What helps generalization, reduces overfitting etc
- We will focus on understanding why models behave the way they do.
- How to think about privacy, fairness etc, How to define them mathematically. What is possible/impossible

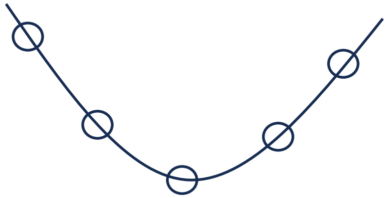
Today

- A very quick introduction to machine learning
  - Regression
  - Classification
  - Neural network
  - ML process and pipeline
- Introduction to Learning theory
  - Notations and space of models
  - Loss and empiricial risk minimization
  - Introduction to PAC learning, sample complexity, loss functions
- Brief discussion of other topics: Privacy, fairness, explainability
- Homework: Analysis of simple ML problem: Classifying ripeness of papayas from color
  - How many papayas do we need to have a good prediction threshold?

Regression (curve fitting)



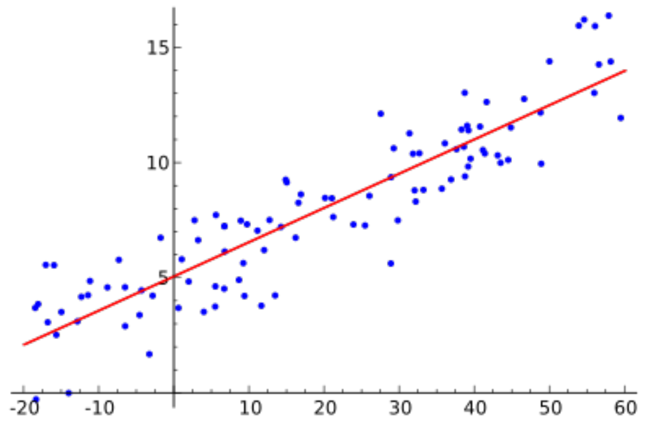
- Suppose points lie in a line
  - $y = mx + c$
- How many points do we need to “learn” the line model?



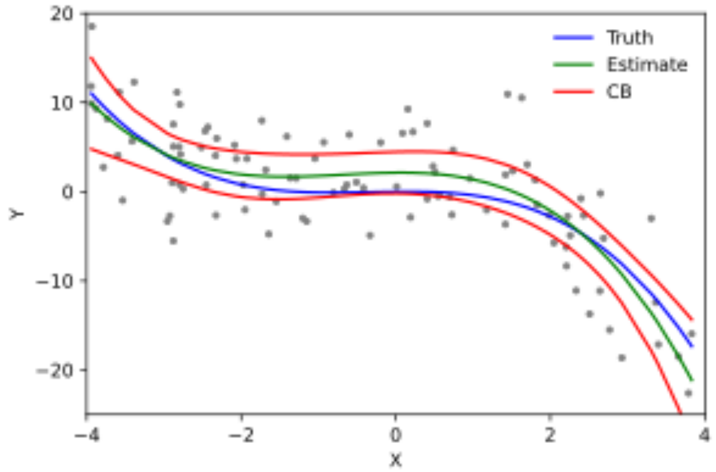
- Suppose the points were on a 2^{nd} degree curve
  - $y = ax^2 + bx + c$
- How many points do we need to learn it?

Regression in reality

- Points may not be exactly in the line
  - They may contain noise (real world issues may cause them to deviate slightly from exact coordinates)



- We do not know the right “degree”
  - Visual observation is unreliable
  - In high dimensions, we can’t even visualize
  - Thus, we do not know the right class/type of model

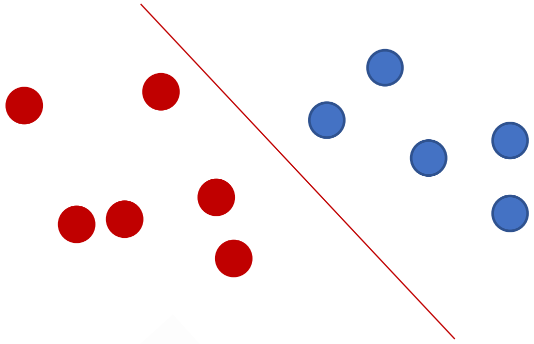


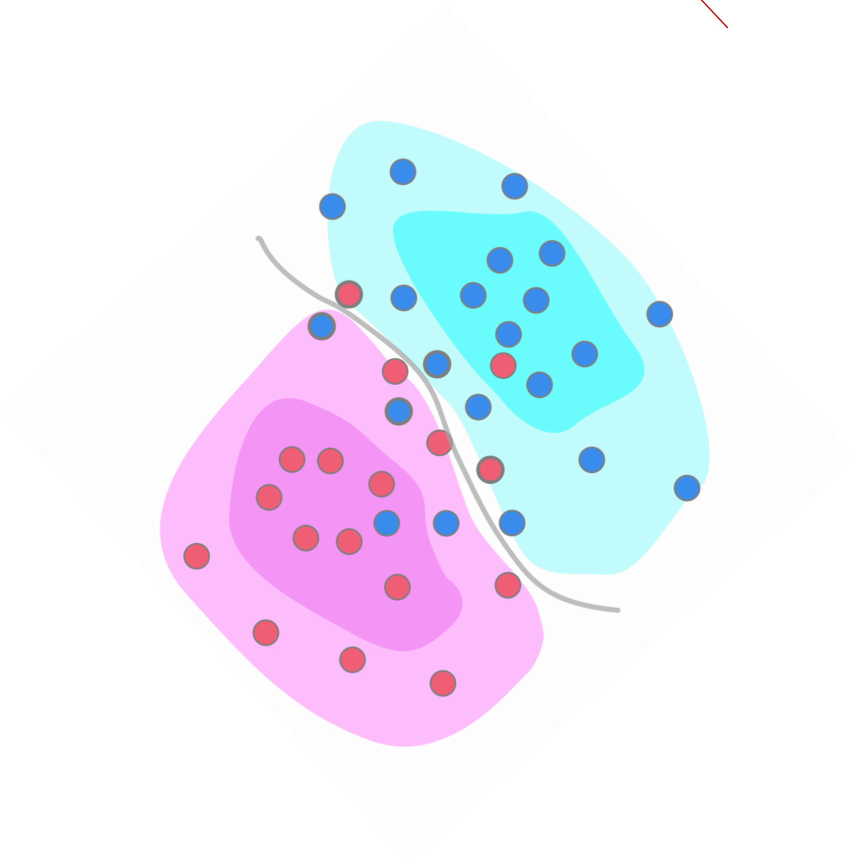
Images from wikipedia

Classification

- Could be simple
  - Linear separation
  - Red: $y \leq mx + c$

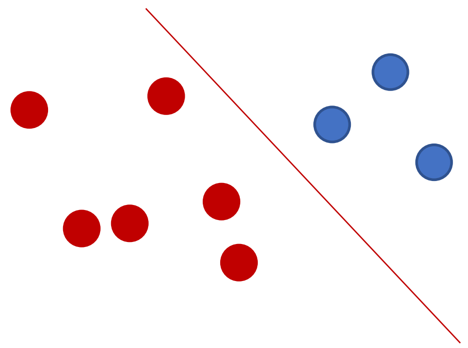
- Or more complex
  - Red $y \leq ax^3 + bx^2 + cx + d$

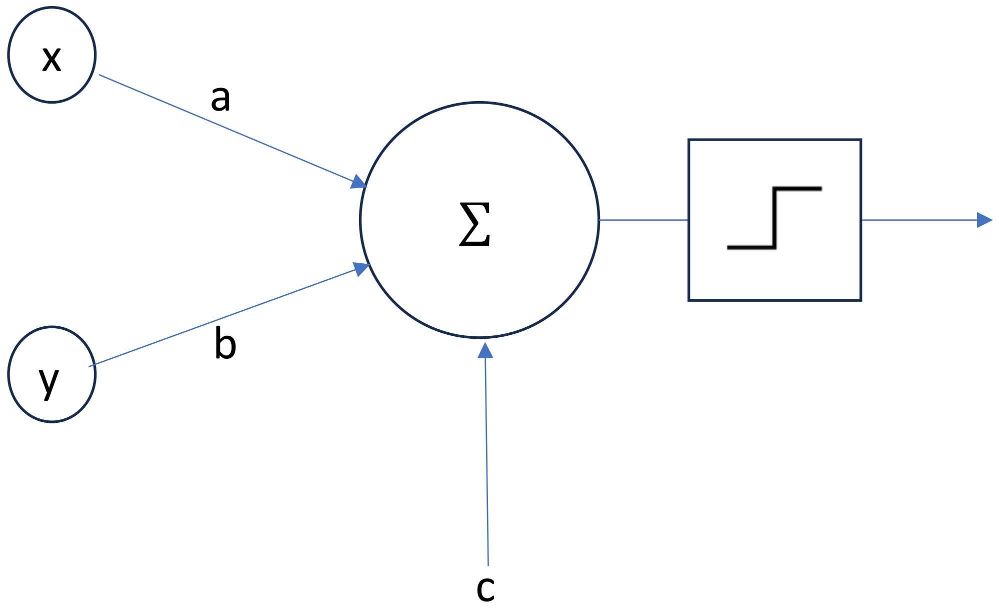




Perceptrons

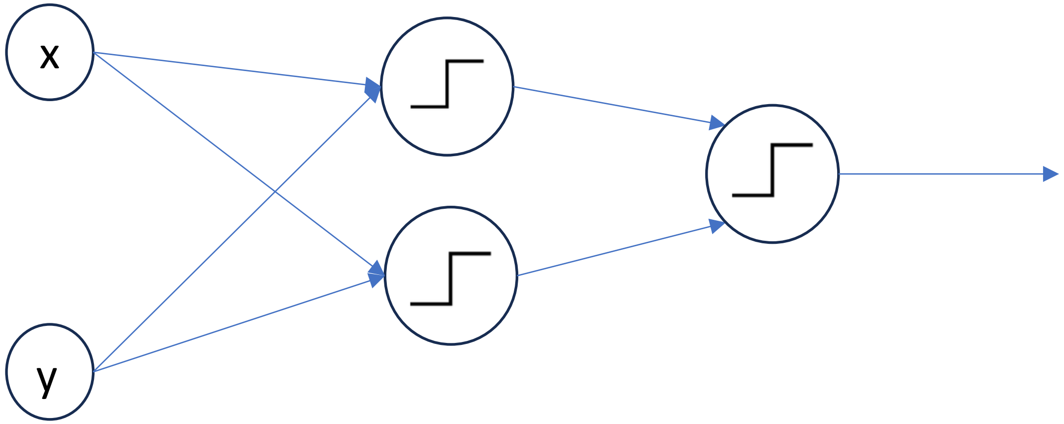
- Straight line separators
  - $ax + by + c \geq 0$
- Can be drawn diagrammatically
- Often the summation sign is omitted and activation function put in place
  - Summation is assumed
  - c is assumed and ommited

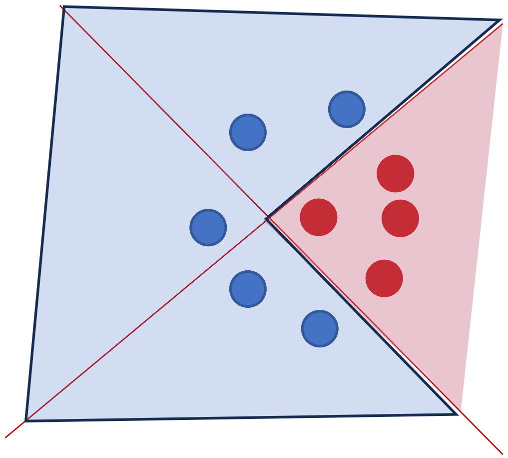




What do we get with more perceptrons?

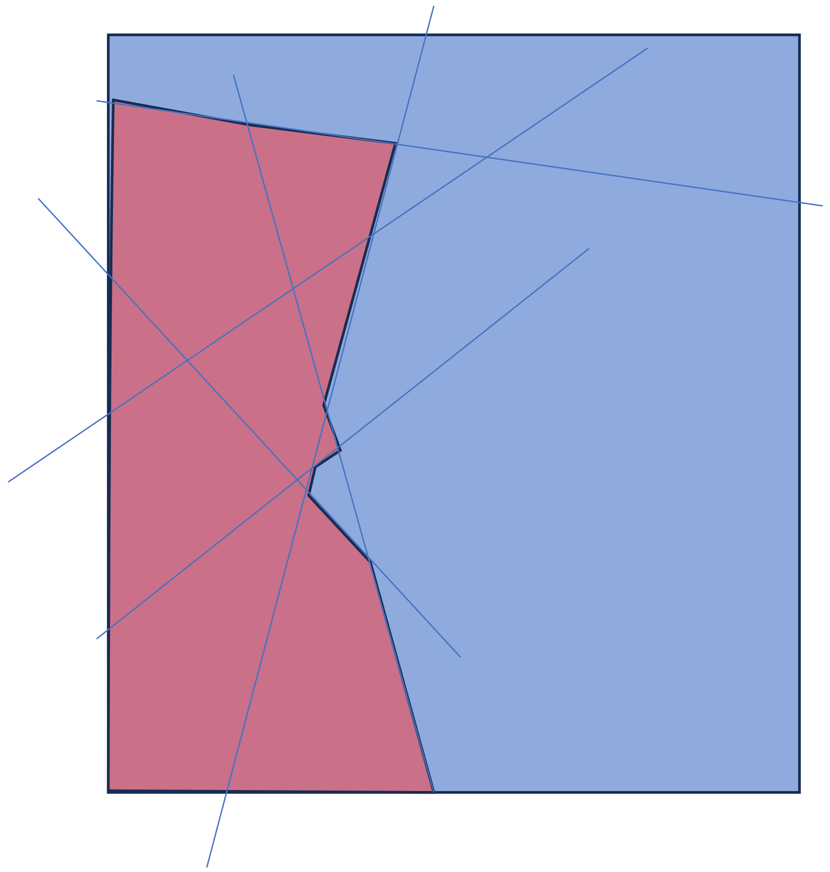
- Find out what weights would be needed to achieve something like this





With more complicated networks:

- We can achieve more complicated decision boundaries



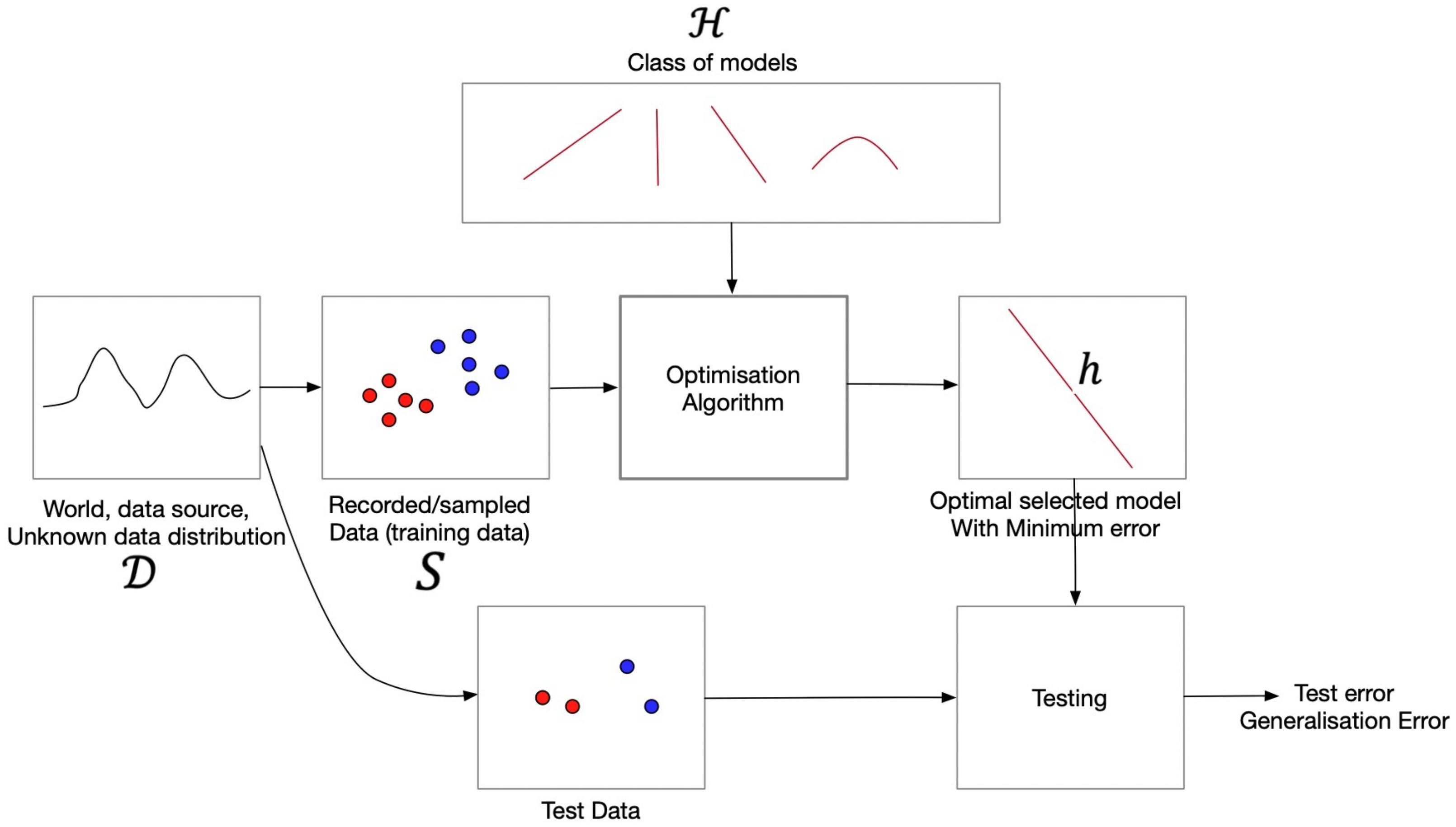
Machine learning as a function to describe the data

- Computation of mean
  - Is a very simple description of the data
- Mean + Standard deviation
  - A more detailed description
  - We can test if a data point is likely from the same source
- Mean + standard deviation + a class of distribution, e.g Gaussian
  - An even more detailed description
  - We can generate similar data

- Other ML models: Neural nets, SVM etc
  - Other types of multi-value functions describing the data
  - The task is to find the different aggregate values, i.e. the model parameters

The machine learning pipeline

- Assume there is a distribution $\mathcal{D}$ from which data is drawn
  - S is a sample of m data points used as training data
  - Written as $S \in \mathcal{D}^m$
- $\mathcal{H}$ is a hypothesis class: a set of possible models
- $h \in \mathcal{H}$ is a model E.g. a model selected by an algorithm
- An optimization algorithm $\mathcal{A}$ takes in $S, \mathcal{H}$ and produces a model h that it thinks has lowest errors on S
- h is tested on test data also taken from $\mathcal{D}$ to estimate generalization of h


$\mathcal{H}$
Class of models
Optimisation
Algorithm
h
World, data source,
Unknown data distribution
$\mathcal{D}$
Recorded/sampled
Data (training data)
S
Optimal selected model
With Minimum error
Test Data
Testing
Test error
Generalisation Error

Question:

- Can the model class be all possible models?

Models as vector

- For a known class of models, we can represent them by a vector of numbers (parameters):
  - Neural networks: A vector of edge weights
  - Regression: Coefficients of polynomials
- Observe: The vector of numbers does not say the type (class) of model. That is up to us.

- Usually, this vector is written as a weight vector $\boldsymbol{w} = (w_1, w_2, w_3, \dots)$
- The size or dimension of is the size of the model
  - Larger models are likely to be more *complex.*

The *space* of models

- If each model is a vector
- We can imagine a vector space or Euclidean space
  - Where each point is a model
  - The dimension of the space (number of weights, coordinates) is the dimension of $\mathcal{H}$

- Optimisation Algorithm: Search over all possible (a, b) to find the best model

Notations

- **Domain set $\mathcal{X}$**. Form which data is sampled
- **Label Set $\mathcal{Y}$.** From which labels are drawn. Eg. {0,1} or {-1, +1} red or blue.
- **Training data (sample set): $\boldsymbol{S} = \{(x_1, y_1), \ldots (x_m, y_m)\}$** (we assume random sample)
- **Model, hypothesis, classifier, predictor $\boldsymbol{h}$**:
  - A function $h: \mathcal{X} \rightarrow \mathcal{Y}$. That is, $h(x)$ returns a predicted label y
- **Hypothesis or model class $\boldsymbol{\mathcal{H}}$**: The set of functions from which h is chosen
- **Algortihm A:** Chooses hypothesis h based on S

Notations

- **Data generating distribution $\mathcal{D}$:** An unknown probability distribution over $\mathcal{X}$. The training data is assumed to be sampled from $\mathcal{D}$
  - We also assume there is a function f giving true labels of data
  - Both $\mathcal{D}$ and f were unknown to us (and to the learning algorithm).
- **Success measure: Loss/error function *L*:** The learning algorithm gives a hypothesis h
  - The true loss of h is defined $L_{\mathcal{D},f}(h) \stackrel{\text{def}}{=} \underset{x\sim\mathcal{D}}{\mathbb{P}}[h(x) \neq f(x)]$
  - When drawn from $\mathcal{D}$, L is the probability that label predicted by h will *not* match the true label

Searching for the best model: Empirical risk minimization (ERM)

- Given a dataset S of size m,
- The empirical loss of hypothesis h is defined as
  - The average loss over all data points

$$L_S(h) \stackrel{\text{def}}{=} \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m}$$

- This is called the empirical risk or empirical error or empirical loss
- ERM is finding h with minimum $L_S(h)$

Observation 1

- Finding the true minimum-loss L_{min}
- may be difficult
  - E.g. Searching in an infinite model class is not easy
  - And we do not know what the min loss is

- What we can hope is to ensure that loss is not much higher than minimum
- That is, loss is approximately minimum
- It is not higher than $L_h \leq (1+\epsilon)L_{min}$
- Note: we do not know L_{min}

Observation 2

- Our eventual goal is a good classifier for $\mathcal{D}$
- But we work on S : a random sample of $\mathcal{D}$
- We cannot guarantee that S is a good representative of true $\mathcal{D}$
- But, with enough samples, we are *likely* to get close
  - But not for sure. It is still probabilistic

- So, with enough data:
  - Probably, we can approximate the minimum loss model!

PAC learning

- Probably Approximately Correct learning
  - If true min-loss is L_{min} and L_h is loss of h, then
  - PAC learnability means we can get:
  - $\Pr[(L_h - L_{min}) \leq \epsilon] \geq 1 - \delta$
- (Hopefully for small ϵ, δ)

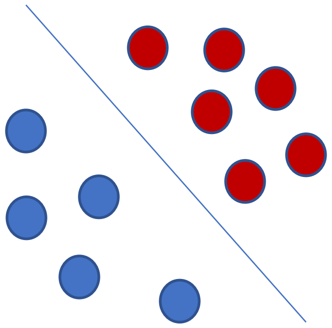
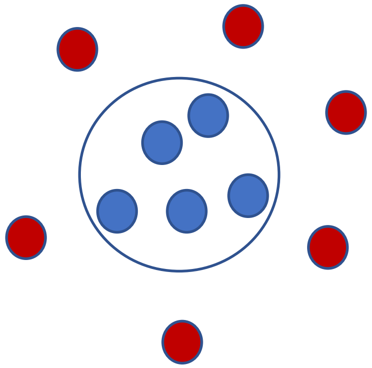
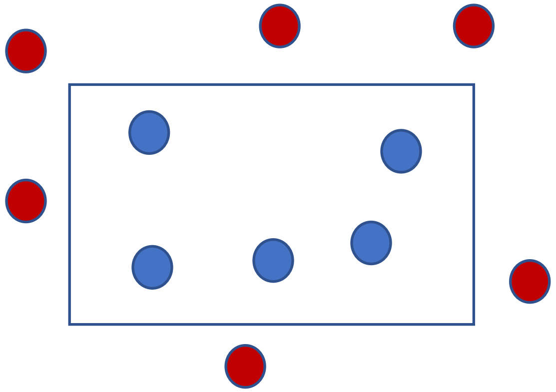
Sample complexity

- How many data points do we need?
- We will show that finite hypothesis classes need:

$$m_{\mathcal{H}}(\epsilon, \delta) \;\leq\; \left\lceil \frac{\log(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil$$

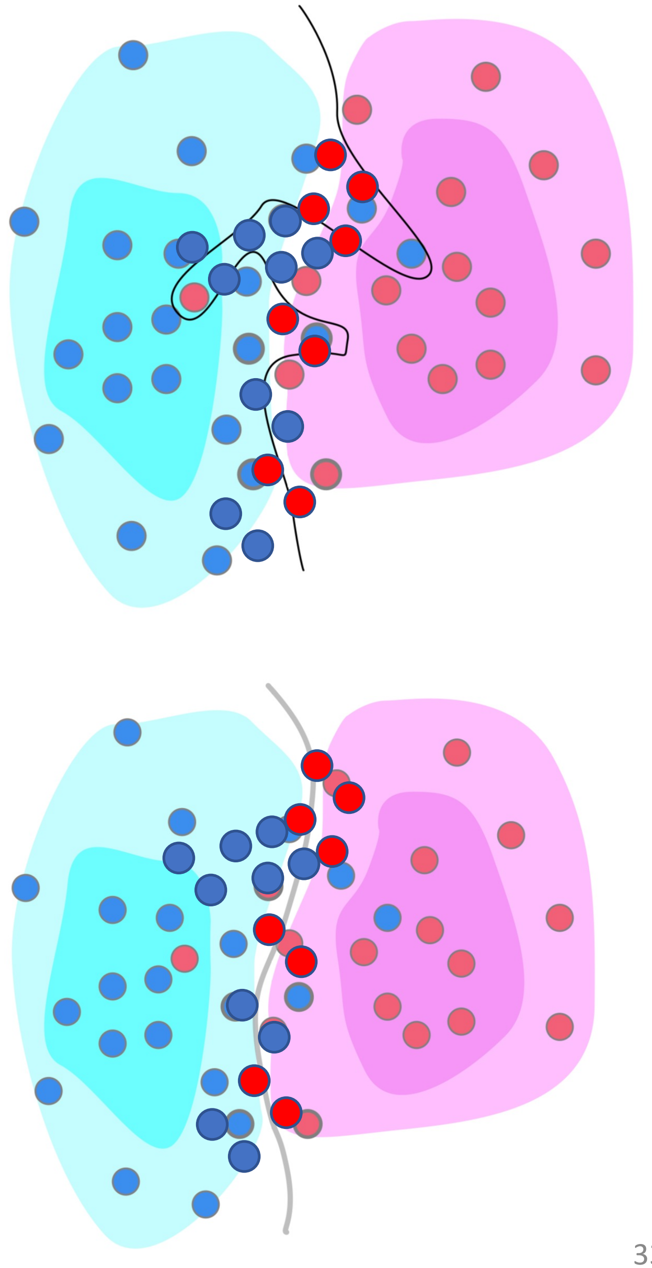
What happens for infinite classes?

- Instead of $|\boldsymbol{\mathcal{H}}|$, we use VCdim($\boldsymbol{\mathcal{H}}$)
- VC dimension is a measure of dimension (complexity) of $\mathcal{H}$
- Examples- increasingly higher vcd

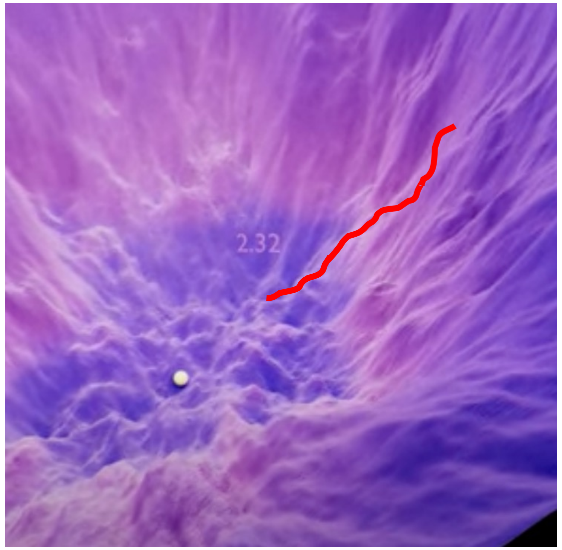
Overfitting

- More complex model classes have more flexibility
- Larger space of possible models
- Algorithm finds a model with smaller loss that works well for S
- But more likely to overfit
- Perform badly on unseen data from the same distribution



Loss functions and optimization algorithms

- Other loss functions are used to make learning easier
  - Cross entropy loss is a common one for deep learning
- We will look at cross entropy loss and what it implies for deep learning, overfitting and geenralisation
- We will study stochastic gradient descent algorithm used to train neural networks and its loss landscape



Privacy

- The problem: We need data for ML
  - Data comes from people
- It may reveal sensitive information about people
  - The data itself may get leaked
  - The ML model, or decisions made by it may reveal information

- Simple example: A company releases average salary every week. When you join the company, someone can guess your salary comparing with previous month’s average. (how can you prevent that?)
- Other model parameters or functions of the data can similarly reveal info
  - Because each new data point causes a small change in the function value/model

Privacy preserving machine learning and differential privacy

- The study of these small changes to functions and models
  - Understanding the effect of each tiny data point
- A different perspective in ML/AI
  - How can we reveal some facts and hide others?
  - What are the limits of this tradeoff?
- Differential privacy works by adding small calculate amounts of noise to models. Precise Bayesian definitions and properties

Fairness

- ML can be unfair
- E.g.
  - Most employees in a company belong to a certain community
  - The CV scanning software learns that bias
  - Even if the community is not stated explicitly (e.g. correlations with name, address etc)
- Data is likely to be biased toward the majority
- Anyone can be a minority with suitable combination of parameters (e.g. race, religion, gender, age, advanced degree..)

Fairness

- Fairness study is important for better ML
- E.g. a bank software refuses loan to a “minority” due to fairness failure
  - Bad for the person
  - Bad for the society in the long term that a deserving person did not get a loan
  - Bad for the bank as they miss a good investment

Fairness

- We will study
  - Precise mathematical definitions
  - See that everything we want may not be achievable
- Fairness is a complex topic
  - What is fair from one perspective is unfair from others
  - Many possible metrics of fairness
  - Affected by other properties like generalization, stability, privacy etc…

Explainability

- Why did the model produce a particular result?
  - E.g. a model predicts rain tomorrow
- Was a specific feature(s) were important?
- Did certain training data points play a role?
- Will a different type of model work better?

Explainability

- Giving scores to features
  - For a particular output
  - For general accuracy of the model
    - Which features contribute more to accuracy?

- Attaching value to data
  - Which data points contribute more to the accuracy?

- We will study:
  - Shapley Value from Economics – assigns Value to different items
  - Other techniques

Next:

- We will start with simple problems and models that we can analyze easily
- Build toward more complex topics
- Keep an eye on announcements and materials on Learn and Piazza.

Homework

- Read chapters 1 and 2 of the lecture notes (to be uploaded)
- Do the exercises
- Read chapters 1 and 2 of the book: Understanding Machine Learning

- Make sure you can understand these comfortably and can do exercises
- This course is mathematical and *not easy.*
- Change to a different material if you are not comfortable with the material
- Do the homework problem next

Homework: A simple classifier

- A supermarket has asked us to build a model classify ripe papayas
- Green is unripe, yellow is ripe
- A sensor reads the colour
- And returns a value in [0,1]
- Assume the supermarket sends us a random sample of labelled readings
- There is a color threshold t* of ripe papayas but we don’t know it.

Homework: Simple classifier

- Question:
  - Describe the different parameters of the ML model in this case.
  - What is the ERM objective?
  - Show that sample size $\mathrm{m} \geq \frac{1}{\epsilon} \ln \frac{2}{\delta}$ suffices to get ϵ, δ (PAC) accuracy.

  - Hint: You may need the inequality: $(1-p)^{\frac{1}{p}} \leq \frac{1}{e}$