

Foundations of Machine Learning

Introduction

1

About this class

- Expect a mixture of theoretical principles and practical experience on real data
- Two-pronged approach, with an emphasis on
 - Theory and foundations for solid understanding
 - Will derive principles from very basic assumptions
 - Will develop algorithms from scratch
 - But also, a discussion of tools, implementation details, and practical advice
 - Will program, test, and optimize many of the learning algorithms
 - Will discuss practical concerns and provide useful implementation tips
 - Will illustrate the use of popular ML libraries
- Concepts and algorithms that *every* ML practitioner should know
- Running themes
 - What does it mean to learn something
 - How can machines learn something
 - When can machines learn something and at what cost
 - How do we know that the learning process succeeded

} *quantify learnability*

2

2

Topics covered

• Models

- Linear regression: simple, multiple, polynomial, locally weighted
- Linear models
- Logistic regression
- Boolean-based concept learning
- Naïve Bayes classifiers
- k -nearest neighbors classifiers
- Neural networks
- Decision trees
- Ensemble methods
- Support vector machines
- Cluster identification
- Dimensionality reduction
- Kernel methods

• Algorithms

- Gradient descent: batched, incremental, stochastic
- Coordinate descent
- Boosting
- k -means clustering
- Principal component analysis

• Concepts

- Loss functions
- Bias and overfitting
- Bias-variance tradeoff
- Regularization
- Test, Train, validation, cross-validation
- Accuracy, precision-recall, ROC, F1
- Entropy

3

3

Why these topics?

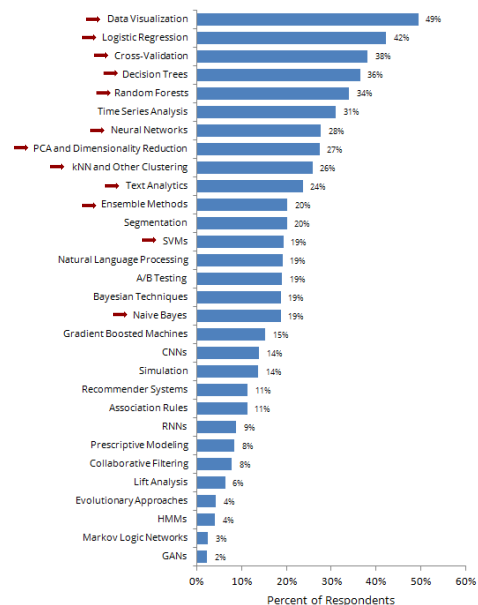
- The following criteria were used in choosing topics:

1. *Universality*. The principles and methods presented should be shared by many different learning algorithms
2. *Simplicity*. Lofty theories are ok for seasoned professionals; for an intro class, excessive sophistication is discouraging
3. *Programmability*. Each algorithm should be fairly easy to implement in a general-purpose programming language that does not rely on fancy libraries
4. *Popularity*. The methods presented should be widely used in practice
5. *Consistency*. All material should be of similar level of difficulty, and it should rely on similar concepts and mathematical abstractions

4

4

What methods do ML practitioners use at work?



Data are from the Kaggle 2017 The State of Data Science and Machine Learning study. You can learn more about the study and download the data here: <https://www.kaggle.com/surveys/2017>. Respondents were asked to indicate which data science methods they use at work. A total of 10153 respondents answered the question.



[Kaggle Survey 2017](https://www.kaggle.com/surveys/2017)

Copyright 2018 Business Over Broadway

5

5

Expectations

- **Mathematical background¹**
 - Calculus (differentiation, partial derivatives, chain rule, gradient)
 - Analytic geometry (equations of lines, planes, parabolas, circles, ellipses)
 - Linear algebra (vectors, matrices, linearity, independence, eigen-things)
 - Probability (expectation, conditional probability, independence, max likelihood)
 - Discrete math (permutations, combinations, counting techniques, relations, graphs)
- **Programming experience**
 - All examples and homework will be done in Python
 - Willingness to learn and explore the functionality of various Python libraries on your own (NumPy, Pandas, Matplotlib, Scikit-learn)
- **Background in algorithms**
 - Algorithm design techniques
 - Data structures
 - Complexity analysis

¹ You can find review videos at <https://www.khanacademy.org/>

6

6

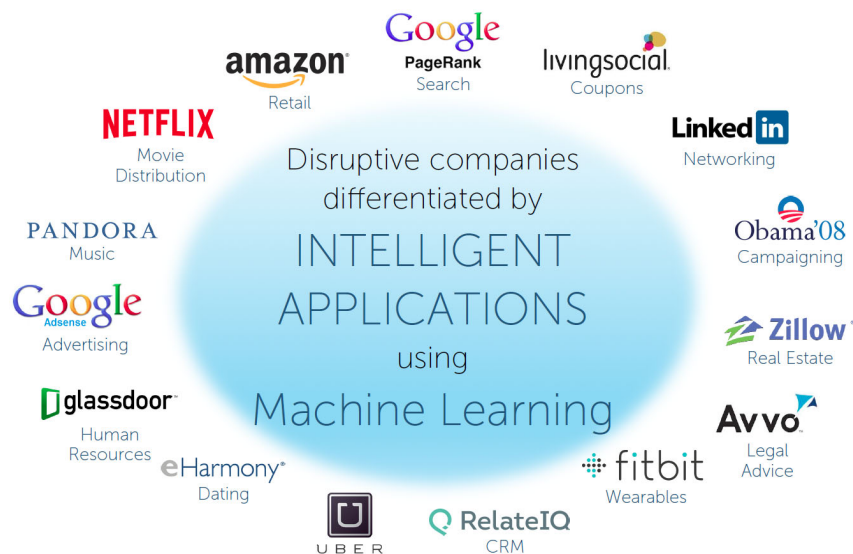
The hype

- *"A breakthrough in machine learning would be worth ten Microsofts"*
Bill Gates, Microsoft
- *"Machine learning is the new electricity"*, Andrew Ng, Stanford U.
- *"Machine learning is the next internet"*, Tony Tether, Director, DARPA
- *"Machine learning is going to result in a real revolution"*, Greg Papadopoulos, Former Chief Technology Officer, Sun Systems
- *"Machine learning is today's discontinuity"*, Jerry Yang, Yahoo Founder
- *"Machine learning today is one of the hottest aspects of computer science"*, Steve Ballmer, Chief Executive Officer, Microsoft

7

7

Societal impact



© Emily Fox & Carlos Guestrin

8

8

Ethical considerations

- ML is a transformative technology that has permanently changed society
 - Other such technologies: electricity, the combustion engine, the internet, etc.
- The potential benefits in healthcare, entertainment, education, commerce, etc. are enormous, but the potential for harm is just as great, including surveillance, disinformation, manipulation of financial markets, and more
 - *Bias*. A system trained on historical data will reproduce historical biases
 - Careless application of ML algorithms will likely entrench or aggravate biases

Examples. A system to predict salaries, systematically gave women smaller salaries, a system to generate faces of executives, generated mostly males, etc.
 - *Explainability*. ML systems that learn billions of parameters are used routinely to make decisions rarely understood by the creators or users
 - *Concentration of power*. All the largest companies are investing heavily in ML. Job automation will likely affect more severely workers with fewer skills
 - *Existential risks* from misuse of technology (climate change, nuclear weapons)

9

9

What is machine learning?

- Machine learning is the study of algorithms that learn from observed data but...what is learning?
- *Learning is any process by which a system acquires a skill or improves performance from experience* (Herbert Simon)
- *Machine learning is a set of techniques that give computers the ability to learn from data, without being explicitly programmed* (Arthur Samuel)
 - Samuel coined the term *machine learning*. He wrote the first checkers program that improved by playing against itself thousands of times
- *Machine learning is a field of study in artificial intelligence concerned with the development of statistical algorithms that can effectively generalize and thus perform tasks without explicit instructions* (Wikipedia)

10

10

A working definition

- *Machine learning is the study of algorithms whose performance P in some task T improves with experience E (Tom Mitchell 1998)*
 - A well-defined learning task is given by the triple $\langle T, E, P \rangle$
- Example.* Your software records which emails you mark as spam, and based on that, learns to better filter true spam
 - T : filtering out spam emails
 - E : analyzing database of emails already marked as spam or not spam
 - P : percentage of emails correctly classified as spam or not spam, computed on an independent set of emails
- Machine learning is *not* learning in the traditional sense
 - If you slightly distort the input, the output may become entirely wrong!
- The goal of the learning task is to fit a *mathematical model* from the data

11

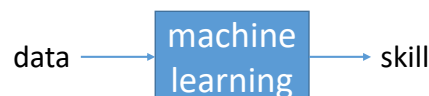
11

Learning vs. machine learning

- **Learning** is the process of acquiring a **skill** through **experience** accumulated from repeated **observations**



- **Machine learning** is the process of acquiring a **skill** through **experience** computed from a substantial amount of **data**



- Skill refers to making accurate predictions for task T , evaluated by some quantitative measure P of performance, e.g., prediction accuracy

12

12

The learning task T

Common types of learning tasks include:

- *Regression*. Predict a numeric value associated with an input instance, e.g., the expected claim that a person ensuring their auto will make
- *Classification*. Predict to which of k classes a given input belongs to, e.g., diagnosing an ailment from a set of symptoms
- *Denoising*. Given a corrupted sample, suggest the clean sample that the noisy sample came from, e.g., clean up a garbled audio clip
- *Transcription*. Compute a textual description of an object from a description given in non textual format, e.g., image captioning
- *Recommendation*. Identify products that a customer may like based on purchase history
- *Clustering*. Identify natural groupings of similar objects based on their attributes, e.g., clustering genes to determine new functions for unknown genes
- *Anomaly detection*. Detect unusual/atypical events, e.g., fraudulent transactions
- *Synthesis*. Generate examples similar to a given one, e.g., a musical fragment in the style of Mozart, a painting in the style of Dali, etc.

13

13

The performance P

- To evaluate a learning algorithm, need a quantitative measure P of performance
 - A *loss function* measures how far a prediction is from the true value
 - A *cost function* is a cumulative measure of the total loss
- The specific measure depends on the task T
 - For classification tasks, a common measure of performance is *accuracy*, defined as the percentage of samples classified correctly, but several other measures are commonly used (can you think of some?)

Question. Is accuracy a good measure of performance when predicting if a person has tuberculosis?

Question. What is a good measure of performance for a regression task, such as predicting the sale price of a house?

 - What is worse? a small number of medium mistakes or a few large mistakes?
- For complex tasks (e.g., transcription) the right choice of P is not obvious
- P should always be measured on an independent set of test data

14

14

The experience E

- The “experience” of a learning algorithm derives from training data S available for learning
 - The data may in S be *labeled* or *unlabeled* or *mixed*
- The goal is to use E , i.e., the data in S , to compute a model that accurately reflects the structure or predicts the labels of the data in S

Example. S is a collection of music clips each labeled with the correct genre
- For labeled data, the model needs to *generalize*, i.e., predict accurate results for observations not in S
- Most often, the model has a “fixed form”, i.e., must selected a priori from a collection \mathcal{H} of similar candidate models, defined by a set \mathbf{w} of parameters
 - Each value of \mathbf{w} corresponds to one model in \mathcal{H}

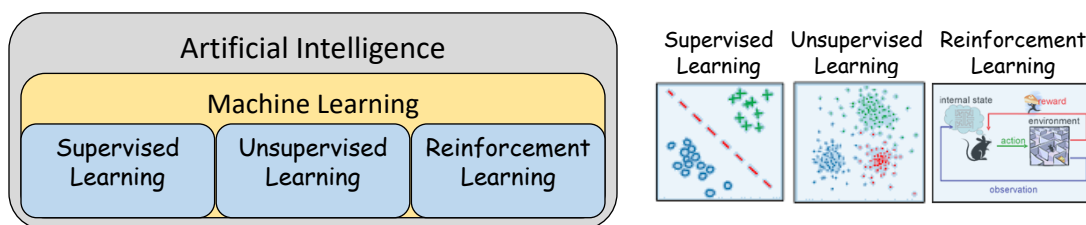
Example. Equation of a line, plane, ellipsoid, A tree of certain depth, etc.

15

15

Learning methods

- ML methods can (coarsely) be divided into 3 types:



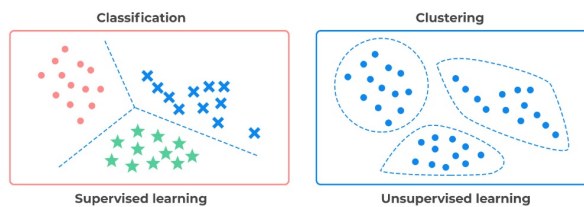
1. *Supervised learning* computes a mapping from input data to output prediction. To this end, a set of correctly labeled data is provided in order to learn a good mapping. Output is either a number (*regression*) or a category (*classification*)
2. *Unsupervised learning* discovers the latent structure of unlabeled data. The main types include *clustering* similar items, *generation* of new ones, and *outlier detection*
3. *Reinforcement learning* introduces the idea of an agent that interacts with the world by performing actions at each time step. Each action carries a *reward*. The goal is for the agent to learn to choose the actions that maximize the average reward

16

16

Supervised vs. Unsupervised: Example

- For illustration, consider the problems of *classification* and *clustering*. In both, each object x belongs to a class $f(x) \in \{1, \dots, k\}$
 - In classification, the goal is to learn f , i.e., find $h \approx f$
 - In clustering, the goal is to partition S into k subsets of similar objects
- A machine learning algorithm is:
 - *Supervised* if the exact f is known on the whole training set S (the data is labeled)
 - *Unsupervised* if the exact f is unknown on any $x \in S$ (the data is unlabeled)
 - *Mixed* if the exact f is known on a strict subset $S_0 \subset S$ (the data is partially labeled)



17

17

Exercise

1. Consider 3 points in \mathbb{R} : $x_1 = -1, x_2 = 1, x_3 = a$, for some $-1 < a < 1$
 - a. Suppose you know that x_2 and x_3 belong to the same class, while x_1 belongs to a different class. Propose a classifier that fits the data
 - b. Assume instead that we do not know the classes but know there are two classes so that instead try to cluster the points into two groups. Propose a clustering algorithm
2. Discuss how to solve problems 1-a and 1-b for the case of $n > 3$ points

18

18

Sample application: real state valuation

- A real state company wants to automate the computation of the estimated price for various properties
- Given:
 - Sale records for 49,000 houses sold since 2015
- Predict a reasonable sale price for a house about to go on the market



19

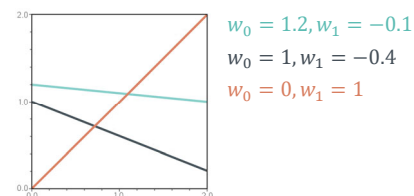
19

Warmup: Predicting the price of a house

- The task T : predict the price of a house as a function of size in square feet
- The model h : a straight line \Rightarrow your model has 2 parameters $\mathbf{w} = (w_0, w_1)$:

$$h(x; \mathbf{w}) = w_0 + w_1 x$$

- The experience E : a list $(x_1, y_1), \dots, (x_N, y_N)$ of actual sizes and prices from past sales, where $y_i = f(x_i)$ is the actual sale price of the i^{th} sale



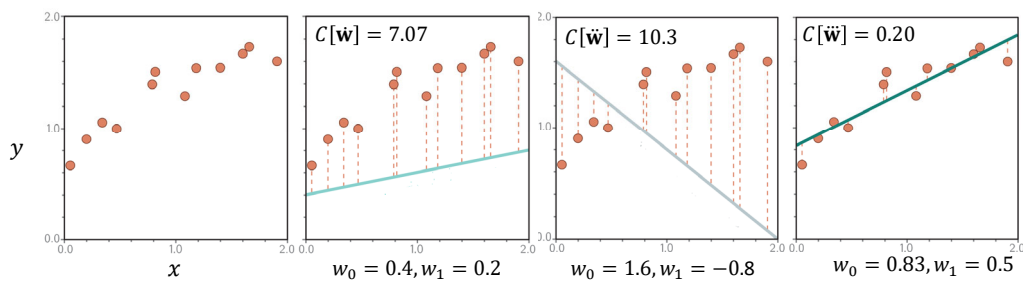
- The performance P is measured by loss and cost functions:
 - The *loss function*, say $(y_i - h(x_i))^2$, captures the mismatch between $f(x_i)$ and $h(x_i)$
 - The *cost function*, say $C[\mathbf{w}] = \sum_{i=1}^N (y_i - h(x_i))^2$
- The learning objective is to find $\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}}\{C[\mathbf{w}]\}$

Exercise. Can you think of other reasonable loss and cost functions?

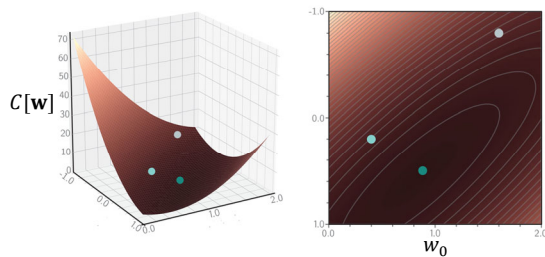
20

20

Warmup...



- **Goal:** minimize $C[\mathbf{w}] = (w_0 + w_1x_1 - y_1)^2 + (w_0 + w_1x_2 - y_2)^2 + \dots + (w_0 + w_1x_N - y_N)^2$



The process of finding \mathbf{w} that minimizes the cost is called *training*

21

21

Exercise

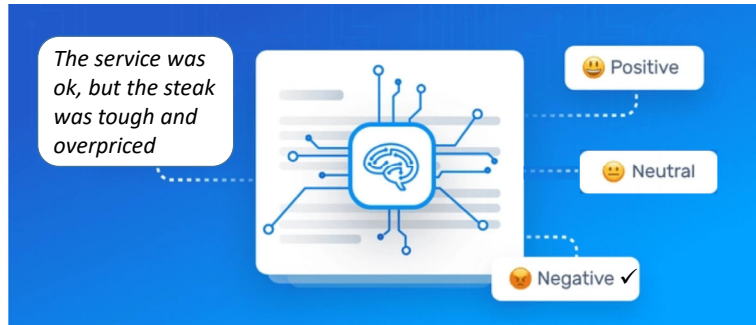
- You are given the following accurate data for training:
(3,12), (4,12), (5,13), (6,18)
Note: in a realistic scenario you would have much more data
- You decide to learn a model $y = w_0 + w_1x$, where $w_0 = A = 5$ is known
- Describe a learning algorithm to minimize $C(A, w_1)$
 1. Express and plot $C(A, w_1)$ as a function of w_1
 2. Find an optimal value of w_1
 3. What are the resulting losses and cost?

22

22

Sample application: sentiment analysis

- A regional manager wants to track the success of a restaurant branch
- Given: a body of textual reviews with accompanying scores (1-5) ☆☆☆
- Learn to predict the sentiment (positive, negative, or neutral) of online text-only reviews appearing in Yelp, GrubHub, etc.



23

23

Sample application: identifying species of plants

- A software assistant learns from botanical records and pictures to identify plant species
- Given:
 - A database of over 650,000 plants and over 475 million images and plant characteristics
- Learn to predict:
 - The scientific name as well as additional information about a plant from a picture of a plant picture taken with your phone



24

24

Sample application: attendance to an event or place

- A manager needs to predict the number of visitors to a national park, museum, etc. in order to plan how many staff members to assign
- Given:
 - Thousands of past records, each describing weather conditions, day of the year, various economic indicators, etc.
- Learn to predict:
 - Expected number of visitors



25

25

Sample application: adjusting drug dosage

- A medical researcher wants to understand the impact of a new drug on a person's blood sugar level
- Given:
 - A set of records that include information about individuals, such as age, body mass index (BMI), physical activity level, diet, family medical history, drug dosage, and the corresponding blood sugar levels
- Learn to predict:
 - Expected blood sugar level of new unseen patients on a given dosage

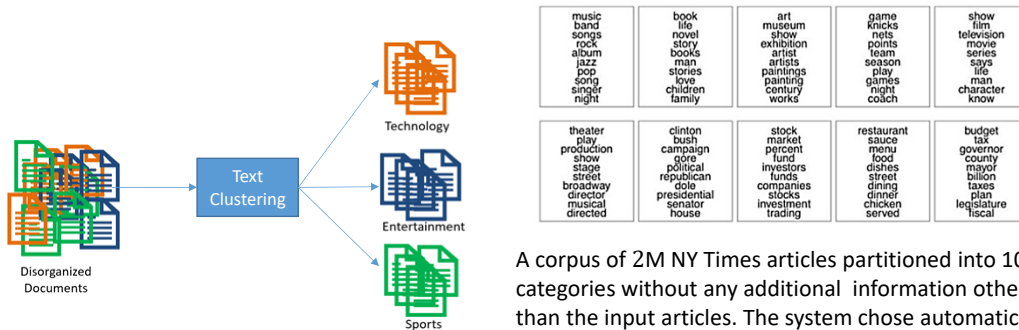


26

26

Sample application: indexing documents by content

- A software system that groups articles about the same topic (science, politics, sports, technology, entertainment, etc.)
- Goal: given thousands of news articles generated by multiple sources in the last few hours, group them into clusters by content type

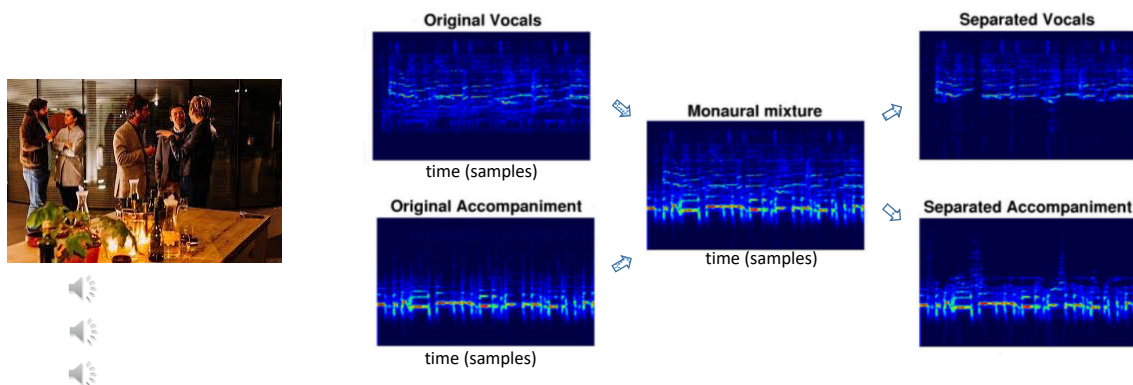


27

27

Sample application: the cocktail party problem

- *Deep Karaoke* is a software application that separates vocals from instruments mixed in a single audio source
- Goal: Given an audio stream recorded from multiple sound sources (speakers, instruments, background noise), separate them into independent audio tracks

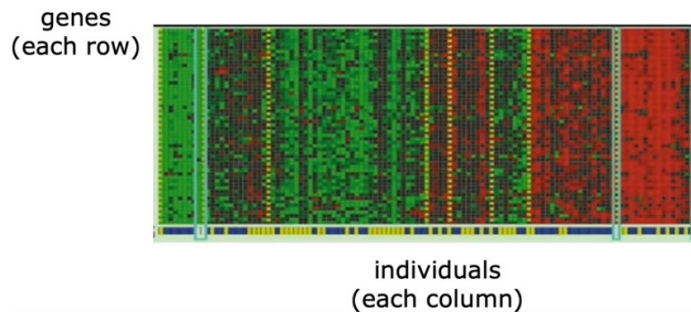


28

28

Sample application: gene expression profiling

- A DNA microarray is a tool in genetics that enables the simultaneous analysis of the expression levels of thousands of genes in a single experiment
- *Goal:* Group individuals by which active genes they have in common for early disease diagnosis, drug effectiveness discovery, etc.



29

29

Sample application: recommender systems

- A software system that makes recommendations of what products to buy or movies to watch
- Given:
 - Your purchase history and reviews, as well as the purchase history and reviews of other consumers, make recommendations of which products to buy

					
A					
B					
C					
D					
E					

30

30

Exercise: rating loan applications

- A software system that helps with approving risky financial transactions
- Given:
 - 10,000 financial records, each describing years of credit, loan balance, annual income, home ownership, past delinquent accounts, maximum number of delinquent billing cycles
- Describe different learning tasks, supervised or not, for this data



31

31

Sample application: Large language models

- ChatGPT is a large language model trained on a large corpus of text data to generate human-like responses to natural language questions
 - Provides explanations and suggestions while engaging in human-like conversations

MA Give me a story based on the following premise: "Biologists visiting an unexplored region of the Andes mountains encountered a talking blue unicorn".



In the heart of the Andes mountains, there lay an uncharted region, untouched by human exploration for centuries. This land was shrouded in an air of mystery, with its lush rainforests, towering peaks, and an ecosystem that had evolved in isolation for millennia. A team of intrepid biologists embarked on an expedition to this remote region, driven by the desire to uncover the secrets hidden within this pristine wilderness. Led by Dr. Isabella Rodriguez, an accomplished biologist with a passion for rare and endangered species, the team braved the treacherous terrain, crossing dense jungles and scaling imposing mountain ranges. As they ventured deeper into this uncharted territory, the biologists encountered species that were not only unique but also utterly astonishing. They discovered vibrant orchids the size of a human head, butterflies with iridescent wings that seemed to shimmer with every hue of the rainbow, and frogs that glowed in the dark. However, their most remarkable discovery was yet to come. One morning, while conducting research in a lush meadow surrounded by emerald-green hills, the team encountered a creature that defied all biological logic. There, standing before them, was a majestic, talking blue unicorn. Its coat was a vibrant shade of cobalt, and its eyes sparkled like sapphires. The unicorn had a kind and gentle demeanor, and it spoke to the astonished biologists in a melodious voice that was both soothing and surreal. "I am Luna," said the unicorn, "the guardian of this sacred land." Dr. Rodriguez and her team were spellbound. They quickly learned that Luna was not only a remarkable creature but also a repository of knowledge about the unique flora and fauna of the region. Luna guided them through the jungles, helping them identify new species and understand the complex relationships within this ecosystem.

...

32

Sample application: autonomous robot locomotion

- Teach a quadruped robot to navigate over uneven terrain
- Given:
 - Information from various sensors
 - No training data
- Get the control system to enable the robot to go from point *A* to point *B* without losing its balance



[terrain before](#)

[terrain after](#)

[stairs](#)

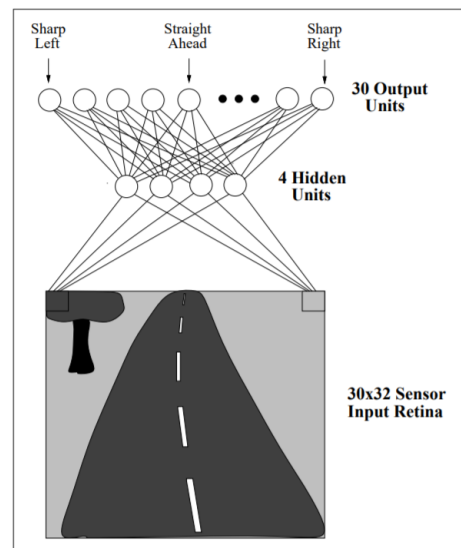
33

33

Sample application: Self-driving cars



- On-board system in a self-driving car uses data from sensors to:
 - stay on the road
 - recognize and avoid nearby objects [link](#)
 - combines many techniques all at once

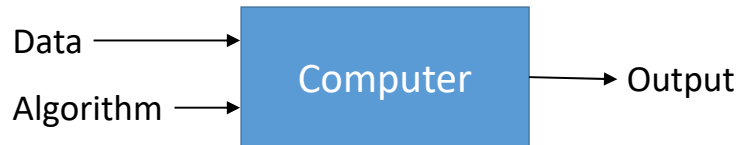


34

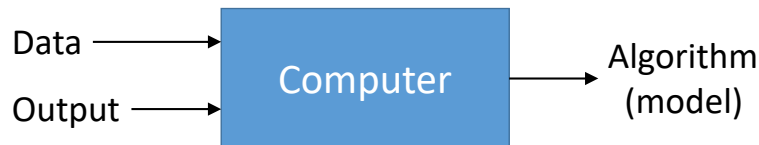
34

Traditional algorithm design vs. machine learning

Traditional Algorithms: focus is on code



Machine Learning: focus is on data



Disclaimer: There is no such thing as machine “learning”. All ML algorithms do is create fancy statistical models.

35

35

When to use ML

- A solution is difficult to code or to describe analytically because the underlying pattern to be detected is too complex to describe formally
 - Our introspection is not sufficiently elaborate to extract well defined instructions
- A system needs to adapt to a constantly changing environment
- Need fast responses/recommendations that would require too long for a human to do or when a human is not available

all require lots of data

Exercise. Which of the following tasks are well-suited for ML?

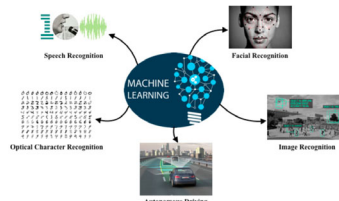
1. Decide whether a pregnancy is at high risk of requiring a cesarean C-section
2. Determine whether a graph, e.g., a social network, contains a clique of >5 people
3. Guess whether the earth might be destroyed by the misuse of nuclear power by the end of the century
4. Determine whether a loud bang came from a firearm or a firecracker
5. Predict if the next cry of an infant will happen at an even-numbered minute

36

36

Niche areas

- ✓ Medical diagnosis
- ✓ Facial recognition
- ✓ Recommendation systems: news, media, goods
- ✓ Content generation: music, stories, pictures
- ✓ High frequency stock trading
- ✓ Consumer focused marketing



- ✓ Handwriting recognition
- ✓ Speech translation
- ✓ Web searching
- ✓ Fraud/spam detection
- ✓ Autonomous driving
- ✓ Sentiment analysis
- ✓ Weather prediction
- ✓ Gaming

37

37

Related disciplines

- *Artificial intelligence*. Symbolic object representation, searching large spaces, knowledge representation
- *Data mining*. Discovery of “interesting properties” in large data sets
- *Statistics*. Bayes’ theorem for calculating probabilities of various hypotheses, error modeling, confidence intervals, statistical tests
- *Algorithmic complexity*. Theoretical bounds of inherent complexity of various tasks, efficient graph exploration, number of examples required to learn.
- *Information theory*. Entropy and information content. Minimum description length approaches to learning. Optimal codes and training sequences
- *Philosophy*. Occam’s razor, induction analysis and justification for generalization
- *Psychology and neurobiology*. Neural network models of learning, power laws to model improvements in response times,
- *Mathematics*. Multivariate calculus, linear algebra, probability, discrete math

38

38

ML vs Expert Systems

- *Expert systems* are programmed to solve problems in a specific domain, based deliberately on the knowledge of experts \Rightarrow *domain aware*
 - Knowledge encoded in thousands of *if-then* rules put together by the joint effort of software engineers and field experts
- In contrast, *machine learning* refers to a collection of algorithms that operate without contextual domain knowledge \Rightarrow *domain agnostic*
 - Requires large amounts of data and computing power to be effective
- Expert systems are fundamentally *deductive* in nature, while machine learning algorithms are *inductive*

39

39

How do algorithms learn?

- The learning task T is modeled by a **target function** f (the ground truth) that captures some property of a set \mathcal{X} of instances (the **instance space**)
 - \mathcal{X} has a well-defined but unknown statistical distribution \mathcal{D} (we write $\mathcal{X} \sim \mathcal{D}$)
Examples. Predict if email is spam (*yes* or *no*), predict the number of retweets of a tweet (an integer), partition a set of news articles into related categories (business, science, sports, health, etc.), diagnose if a patient has COVID, the flu, or a simple cold
 - Learning happens by using a finite subset $S \subset \mathcal{X}$ of instances to infer (induce) a **mathematical model** g (from a set \mathcal{H} of candidate models) that matches or approximates the unknown f
 - Each $\mathbf{x} \in S$ is drawn from distribution \mathcal{D} , i.e., $\mathbf{x} \sim \mathcal{D}$
 - Each instance $\mathbf{x} \in \mathcal{X}$ is specified as a list of d features, a vector $\mathbf{x} \in \mathbb{R}^d$
 - Model is a function (implemented as a data structure, algorithm, etc.) used to predict f
 - Often, the model is *parameterized*, i.e., specified by a set of parameters
- Example:* coefficients of a hyperplane, equation of a circle, propositional logic formula, etc.

40

40

The hypothesis space

- The set \mathcal{H} of candidate models that the learned g is chosen from is called the ***hypothesis space***
- Our choice of \mathcal{H} reflects our assumptions (*bias*) about how the real-world behaves
- There is a “best” model in \mathcal{H} , which may or may not coincide with f
- The learning task consists of using S to *induce* a “good” model $g \in \mathcal{H}$
- For each learning task T , we describe how to combine various algorithmic components, such as a *hypothesis space*, a *cost function*, an *optimization algorithm*, and a *dataset*, to build a *machine learning algorithm* that produces g

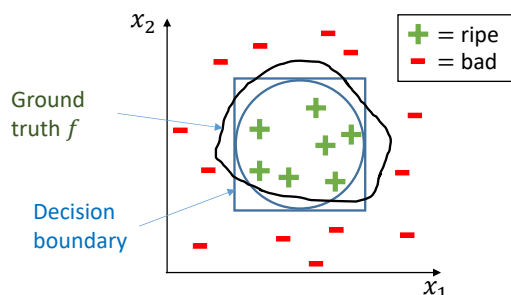
The challenge of finding a good model for S differs from the challenge of finding a model that generalizes to new data

41

41

Example: Is my avocado ready?

- The ripeness (hence, tastiness) of an avocado depends on many factors, including ‘skin softness’ (from rock solid to mushy) and ‘color gradient’ (from light green to dark), date cut from the tree, etc.
- Based on experience, you decide to use softness (x_1) and color (x_2) as predictors of ripeness and collect and correctly label a set S of avocados
- What is a reasonable hypothesis space? What parameters define $h \in \mathcal{H}$?



- Some choices for \mathcal{H} :
 1. Set of circles with radius w_0 and center (w_1, w_2)
 $g(x_1, x_2)$: **return** $\text{sign}(w_0^2 - (x_1 - w_1)^2 - (x_2 - w_2)^2)$
 2. Set of rectangles with bounding box (w_0, w_1, w_2, w_3)
 $g(x_1, x_2)$: **return** $(w_0 < x_1 < w_1) \wedge (w_2 < x_2 < w_3)$
- How do you quantify the quality of your solution?
 - The probability of the area of the symmetric difference: $\text{area}((f \cup g) - (f \cap g))$
 - But we don't know f !
 - $\text{Error}(g, S) = 1 - \%$ of S classified correctly

42

42

Example: How good is this checkers move?

- Given a board configuration, a checkers playing program chooses a move by evaluating candidate boards on its next move
 - Task T : playing checkers
 - Experience E : set of games played against itself or against a human (the training set)
 - Performance P : % of games won against a human (measured on test set)
 - Target f : a function that assigns higher scores to better board configurations
 - Model g : board quality is measured by $g(\mathbf{x}) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$, where,
 - x_1 = # of black pieces left
 - x_2 = # of red pieces left
 - x_3 = # of black pieces under attack
 - x_4 = # of red pieces under attack
- Note that while f is assumed to exist, it may not even be possible to describe
- Arthur Samuel wrote a checkers program that computed g by playing against itself thousands of times.

A **linear model** parameterized by (w_1, w_2, w_3, w_4)
 $g(\mathbf{x}) = \mathbf{x} \cdot \mathbf{w}$, where $\mathbf{w} = (w_1, w_2, w_3, w_4)$

43

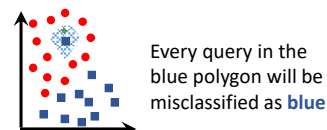
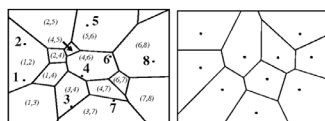
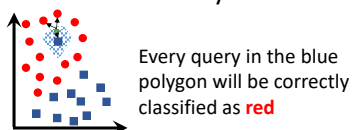
43

Nonparametric models

- Not all models are based on computing a fixed-size set of parameters
 - Such models are called *nonparametric*
- Most common *nonparametric* models are *instance-based*
 - The model is implied by the entire set S of samples
 - Model consists of a collection of *local models*
 - Prediction based on identifying which local model to use
 - Very flexible, but complex (storage and model grow with the size of the training S)

Example. Given a large set S of labeled data (label might be a number or a category), use it to predict the label of an unlabeled sample q .

- Each data sample in S is encoded as a point in \mathbb{R}^d
- Find the k points in S closest to q and report the (weighted) average or the mode
 - How do you do this efficiently?

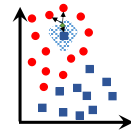
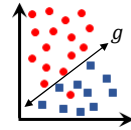


44

44

Parametric vs. nonparametric models

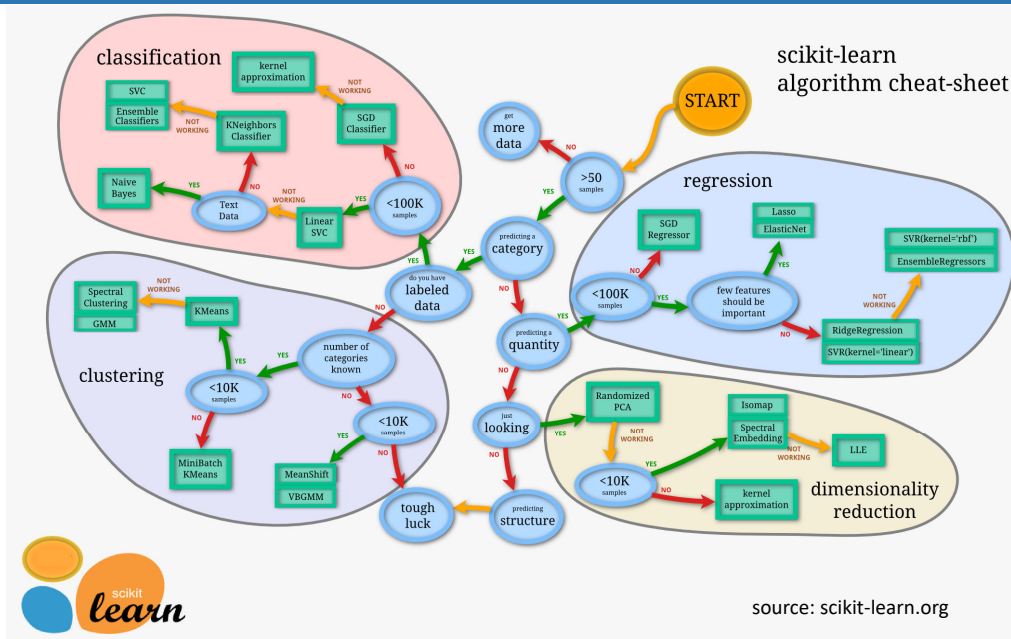
- A *parametric model* makes (strong) assumptions about the nature of f
 - The number of parameters is usually $\mathcal{O}(d)$ or $\mathcal{O}(d^2)$, where $d = \#$ of features
 - All training samples (S) affect the learned “global” model (g)
 - *Pros*: (1) Reduces the problem of learning the ground truth f to the problem of learning d values, but assumes that some $g \in \mathcal{H}$ can approximate f well, (2) the resulting model complexity is independent of the size of S
 - *Cons*: if this assumption is wrong, the model may incur huge errors
- A *nonparametric model*, on the other hand makes minimal assumptions about f
 - The only assumption is that *similar inputs have similar outputs*
 - There is no single global model, instead “local” models are estimated as needed
 - Based on finding similar (past) instances in the training set, using a suitable similarity metric, and interpolating the prediction for x from instances similar to x
 - Different models differ by the choice of similarity metric and interpolation method
 - *Pros*: flexibility, there is no need to “guess” the nature of f
 - *Cons*: complexity grows with the amount of training data



45

45

The ML Model Zoo



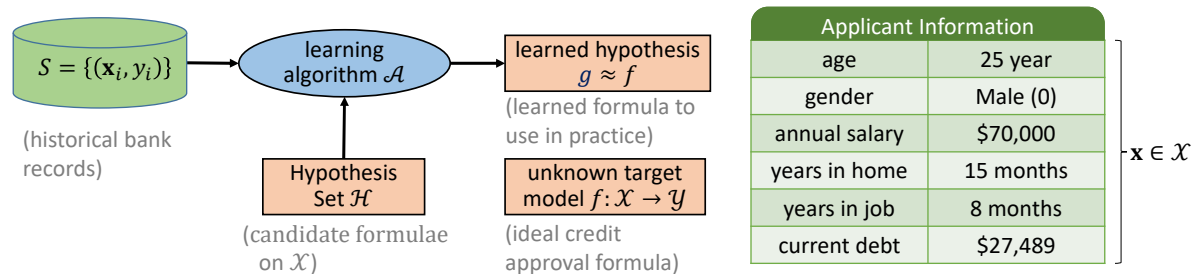
46

46

The learning problem

- Given an unknown target function f (the ground truth) to be learned
- Use a sample S to compute a hypothesis g that matches/approximates f
- A learning algorithm \mathcal{A} uses S to select best g from a set \mathcal{H} of hypotheses
- Set \mathcal{H} reflects our assumptions (bias) about the true nature of f

Example. Credit Approval Application. Given historical approval data, recommend a maximum amount of credit to approve.



47

47

Key assumptions of machine learning

- The fundamental assumption of machine learning is that the data S we train on and test on is *sampled* from a fixed but unknown distribution $S \sim \mathcal{D}$
- The aim is to learn something about reality from as small a sample as possible, and be able to *generalize* to \mathcal{X} what we learn from S
 - If all of S has property p , when we see a new sample \mathbf{x} from \mathcal{D} we expect to show property p
- Main challenge: with a finite number of observations, many hypotheses are consistent, which one do you choose?
 - Keep *all* hypotheses consistent with the data (Epicurus) and report the simplest (Occam)
- We need to assume future data is also distributed according to \mathcal{D}
 - If all you do is to predict on S , then the analysis is not very useful
 - We have S , but the real interest is to understand $\mathcal{X} \sim \mathcal{D} \Rightarrow$ *need to induct*
- Ultimate goal is to produce a succinct description of f , i.e., a model
 - Without additional assumptions, the holy grail of $g = f$ (or even $g \approx f$) is unattainable

48

48

Inductive learning hypothesis and empirical risk minimization

- Since the only information we have about the target f derives from the samples in S , without additional assumptions, *any* learning algorithm can at best guarantee that h fits the sample training data

Inductive Learning Hypothesis. *Any hypothesis found to approximate the target function well over a sufficiently large set of training examples, chosen randomly from distribution \mathcal{D} , will also approximate the target function well over other unobserved examples*
- Once a hypothesis representation is chosen, the goal of **empirical risk minimization (ERM)** is to find the hypothesis that best fits the training data
- Learning then becomes a problem of *searching* in a space of candidate hypotheses implicitly defined by the hypothesis representation
 - Search can often be efficiently organized by taking advantage of the structure of \mathcal{H}

49

49

Learning is hard

- ERM turns learning into a straightforward optimization problem: *simply look for the best fitting hypothesis*
- Unfortunately, it is not that simple, things can go wrong for two related reasons:
 - Overfitting
 - Too big a hypothesis space
- Overfitting
 - Suppose you want to learn a function $f: \mathcal{X} \rightarrow \{0,1\}$, a classification task, and have a training set $S \subset \mathcal{X}$
 - If \mathcal{H} is the set of all functions, the learning algorithm simply memorizes f on S , i.e., reports $g(\mathbf{x}) := f(\mathbf{x})$, as this minimizes the empirical risk, and fails to generalize
 - In the extreme case maybe $f(\mathbf{x}) = 1$ for all $\mathbf{x} \in S$, and ERM comfortably reports $g(\mathbf{x}) = 1$ for all $\mathbf{x} \in \mathcal{X}$, as this minimizes empirical risk. Such g may fail on *every* instance not in S
- While contrived, the example illustrates a real problem with induction
- Will deal with this by restricting the space \mathcal{H} of candidate hypotheses

50

50

An online learning task

- Goal is to learn a threshold τ , such that $f_\tau(x) = +1$ if $x > \tau$, and -1 otherwise
 - Training data comes one sample at a time, in a stream of values
- You keep an estimate $\hat{\tau}$ of τ to predict the label of each sample and pay a penalty for wrong guesses. How do you minimize the penalty? How do you update $\hat{\tau}$?
 - Your learned model is $g(x) = +1$ if $x > \hat{\tau}$ and -1 if $x \leq \hat{\tau}$

1. **for** $i = 1, 2, 3, 4, \dots$ # a stream of samples
2. your learner is presented with sample $x_i \in \mathcal{X} = \mathbb{R}$
3. your learner predicts $\hat{y}_i = g(x_i)$ and then is shown y_i
4. **if** $y_i \neq \hat{y}_i$: you pay a penalty of 1

51

51

The problem with arbitrary hypothesis spaces

- It may come as a surprise that we cannot even learn such a simple task
 - The problem is that \mathcal{H} is too flexible (rich), and the task is not simple at all!
- *Fact*: the set of real number is uncountable while the set of computable numbers is countable
- The adversary is forcing you to learn a number that cannot be computed
 - You cannot possibly learn with an algorithm a number that does not even have a finite representation
- We handle this by restricting the set of candidate hypotheses
 - Suppose both τ and $h \in \mathcal{H}$ are integers or even 32-bit floats

Exercise. Design a correct and efficient learning algorithm for the case when τ and $h \in \mathcal{H}$ are integers, and $\mathcal{X} = \mathbb{R}$
- We will later prove that learning is always possible when \mathcal{H} is finite or infinite with certain desirable properties

52

52

Exercise

- Let's revisit the online learning task, but this time assume that samples are drawn independently from an arbitrary distribution \mathcal{D}
- Suppose we keep track of the largest negative sample τ_L and smallest positive sample τ_U . Clearly $\tau_L \leq \tau \leq \tau_U$
- The algorithm behaves as follows: given a sample $x \sim \mathcal{D}$, $g(x)$ reports -1 if $x \leq \tau_L$; $+1$ if $x \geq \tau_U$; and 0 if $\tau_L < x < \tau_U$.
- An "error" occurs when $g(x) = 0$ as the algorithm cannot guarantee a correct decision. Thus, we would like $\Pr[g(x) = 0]$ to be small
- Suppose that you take N independent random samples from \mathcal{D} . Let $\epsilon > 0$ and $\delta \leq 1/2$ be user-given
- Prove that independent of \mathcal{D} , if N is large enough, then:

$$\Pr[g(x) = 0 \leq \epsilon] \geq 1 - \delta$$

53

53

Warmup: predicting the sentiment on a course

- The CS Chair would like to predict the likelihood that a class will be successful based on data, rather than simple intuition
 - Is a new parallel programming class offered in the morning likely to be popular?
 - Plenty of filled out questionnaires evaluating past offerings are available (a small subset S of these is shown below)

Rating	Morning?	AI?	Systems?	Theory?	Required?	Rating	Morning?	AI?	Systems?	Theory?	Required?
+2	y	y	n	y	n	0	n	y	n	y	n
+2	y	y	n	y	n	0	y	y	y	y	y
+2	n	y	n	n	n	-1	y	y	y	n	y
+2	n	n	n	y	n	-1	n	n	y	y	n
+2	n	y	y	n	y	-1	n	n	y	n	y
+1	y	y	n	n	n	-1	y	n	y	n	y
+1	y	y	n	y	n	-2	n	n	y	y	n
+1	n	y	n	y	n	-2	n	y	y	n	y
0	n	n	n	n	y	-2	y	n	y	n	n
0	y	n	n	y	y	-2	y	n	y	n	y

- How can we construct a reasonable prediction model based on the available data?
 - For starters, we assume that non-negative ratings are favorable and negative ones are not

54

54

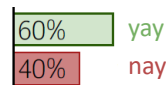
Predicting sentiment on a course...

- Each entry consists of five binary attributes (*yes* or *no* values) and an integer rating that ranges from -2 to $+2$. We decide to consider non-negative ratings as positive sentiment ($+1$) and negative ones as negative sentiment (-1)

- The data suggests that students like 60% of the classes

- Without considering any of the attributes, a reasonable initial guess is that a new class will be met with 60% positive and 40% negative approval \Rightarrow predict $+1$

- Prediction is wrong on 40% of the data



- If you could consider just *one* attribute to help you predict, what should it be?

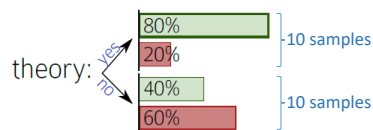
Rating	Morning?	AI?	Systems?	Theory?	Required?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

55

55

Prediction using one attribute

- If you could consider *one* attribute, which one should it be?
 - Choose the one *most useful* for making an accurate prediction on the available data
- Suppose, for example, that you ask about the “theory” content on the new class



- If the new class has a strong theory component (value = *yes*), based on S , you should probably predict $+1$; if it lacks it, you should probably predict -1
- How does this perform on the available data S ?
 - 20% of *theory* = *yes* and 40% of *theory* = *no* are misclassified
 - # of errors = 6 and the error rate is $6/20 = 30\%$, an improvement over the previous 40%

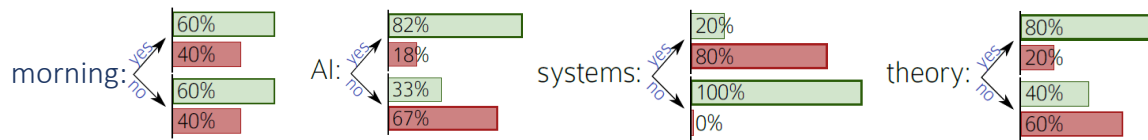
Rate	Mrng?	AI?	Sys?	Thry?	Req?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

56

56

The contribution of individual attributes

- We can figure out which attribute has higher discriminatory power by building similar histograms for each of the available attributes



- What are the histograms when partitioning whether the class is required?
- Choose the feature that would help you guess most effectively the correct label, i.e., the feature that produces a partition with the most *homogeneous* subsets
Exercise. Compute the error rate (# of incorrect guesses) that result from using each of the features. Which feature results in the smallest error rate?
- What do we next if we want to reduce our error rate when making predictions?

57

57

A Divide-and-Conquer Learner

- Once a feature is selected, its values (*yes* or *no*) are used to partition S into two sets
- Each of these two sets can then be processed recursively using one of the remaining features
 - We have converted one classification problem into two smaller classification problems
 - When should the recursive process stop?
 - How do we represent the model we have created?
 - How do we evaluate the model? In other words, what is the P in the learning problem $\langle T, E, P \rangle$ we are considering.
 - Is our approach guaranteed to yield an optimal solution?
 - What do we mean by an optimal solution?

Exercise. Compute the model resulting from empirical risk minimization (ERM)

Exercise. How big is the hypothesis space \mathcal{H} ?

58

58

Feature Extraction

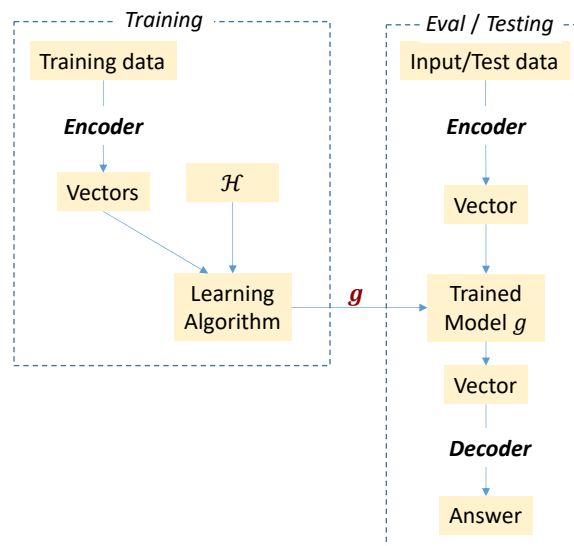
- Raw data is rarely in a form suitable for machine learning. It may be unstructured, have missing values, or contain features irrelevant for the learning task
 - Before using, data needs to be cleaned and transformed into *numeric vector form*
 - These vectors are used to compute distances, angles, and various functions, including predictions and measures of the quality of a learned model
- **Feature extraction** refers to the process of mapping each sample of “raw data” into a fixed-length list of numeric values (**features**) intended to capture the information in the original data relevant to the learning task
 - Includes numeric transformations, value aggregation, TF-IDF, one-hot encodings, etc.
- Each data sample must be converted into a **feature vector** $\mathbf{x} = (x_1, x_2, \dots, x_d)$
 - For efficient processing, we want $\mathbf{x} \in \mathbb{R}^d$, i.e., each **feature** x_i is a real number that describes or encodes some aspect of the data
- The related task of identifying a subset of the relevant features is called **feature selection**, while **feature engineering** includes both extraction and selection

59

59

Workflow

- An *encoder* transforms raw data into vectors (also called *vectorization* or *embedding*)
- A *decoder* transforms output vectors into human interpretable form



60

60

Example

- DU offers a professional MS program in Data Science
 - Three proficiency exams are required, each with scores in $[0,100]$
 1. Programming Basics
 2. Calculus for Data Science
 3. Discrete Mathematics & Linear Algebra for Data Science
 - Can we use this information to predict the likelihood of graduation?
 - Based on historical data that includes exam scores and graduation status (a Boolean), you want to construct a model, specified by a vector $\mathbf{w} = (w_0, w_1, w_2, w_3)$ that issues the following prediction:
 - Student is likely to graduate if $w_1x_1 + w_2x_2 + w_3x_3 > w_0$
 - Student is unlikely to graduate if $w_1x_1 + w_2x_2 + w_3x_3 < w_0$
- where $\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3$ is the feature vector of exam scores
- Note:* the representation allows us to use vector algebra on data


61

61

Example

- Converting images into a set of vectors in \mathbb{R}^{59}

```
In[72]:= dogFeatures = FeatureExtraction[dogImages]
```

```
Out[72]= FeatureExtractorFunction[ Input type: Image  
Output type: NumericalVector (length: 59)]
```

 \mathbb{R}^{59}

```
In[73]:= dogFeatures[
```

A 132 × 190 true color image

```
Out[73]= {11.0441, -3.34324, -0.699841, -5.93598, 2.50764, -3.91537, -7.36306, -0.823358, -2.69656,  
-0.576547, -3.92669, 2.01144, 3.94393, -2.23952, -3.32647, 4.16257, -0.953543,  
-2.76584, 3.19871, 0.637723, 0.999546, -0.757539, -2.50948, 2.02879, -1.3957,  
-0.893909, -0.589072, 1.96187, -1.66788, -0.828078, -1.15881, -2.58314, 2.35955,  
3.01757, 4.32476, 1.23646, -3.17519, -0.557602, 0.992646, 2.69478, -1.52464, 1.2962,  
0.512279, 0.532948, -1.88122, -0.448584, -0.674886, -1.39356, 0.840563, -0.811924,  
-0.465089, 2.13805, 3.48403, 0.0899465, 2.08834, -0.274228, 1.06812, -2.8968, 0.422043}
```

62

62

Example...



$x_0 =$ [11.0441, -3.34324, -0.699841, -5.93598, 2.50764, -3.91537, -7.36306, -0.823358, -2.69656, -0.576547, -3.92669, 2.01144, 3.94393, -2.23952, -3.32647, 4.16257, -0.953543, -2.76584, 3.19871, 0.637723, 0.999546, -0.757539, -2.50948, 2.02879, -1.3957, -0.893909, -0.589072, 1.96187, -1.66788, -0.828078, -1.15881, -2.58314, 2.35955, 3.01757, 4.32476, 1.23646, -3.17519, -0.557602, 0.992646, 2.69478, -1.52464, 1.2962, 0.512279, 0.532948, -1.88122, -0.448584, -0.674886, -1.39356, 0.840563, -0.811924, -0.465089, 2.13805, 3.48403, 0.0899465, 2.08834, -0.274228, 1.06812, -2.8968, 0.422043]



$x_1 =$ [36.1331, 2.82876, -24.4825, -15.2095, 26.4428, -19.2674, -13.4012, -14.4384, -14.6078, 12.8759, 9.03336, 25.8269, -12.107, 1.97599, 2.99312, 28.0356, 12.8927, 5.40877, -11.8331, 10.6467, -3.41838, -37.66, 18.9037, -6.74665, -3.01998, -2.31616, 26.4984, 21.3005, -4.26163, 28.5145, 3.24226, -11.8897, 12.278, 9.07582, 13.8868, -5.72016, -13.8429, 7.37154, -19.3305, -21.1336, -7.21226, -1.18156, -23.4879, 4.14814, -11.841, -10.1581, -10.5729, 25.0829, 0.24281, -6.4124, -17.4222, 8.40178, -8.21909, -3.1472, 19.9038, -0.605468, -2.80223, -23.1513, 3.9007]



$x_2 =$ [-1.53514, 8.2207, 9.50114, 10.5743, -2.73273, 2.12039, -4.09526, 0.61094, -3.26219, -6.48154, -9.03465, -1.90776, -0.975246, -4.16115, -3.74694, 3.66253, 2.9164, 0.493594, 1.37877, 0.40543, 1.32522, -0.629348, -4.89851, -0.0289976, -3.44033, -0.219109, -4.66922, 2.66169, 0.322745, 0.799798, -0.472088, -0.0746354, 3.62066, 1.26609, -2.57828, -3.30924, -2.40182, -3.51722, 4.48906, 0.230277, 3.96838, -3.09817, 0.633618, 0.298719, 0.468354, 1.55659, -1.72312, 2.55912, -2.83487, 0.348612, -1.56584, 0.0828985, 0.179958, -1.45781, -0.6554, -3.08233, -0.903872, 0.657923, 1.85222]

```
print(len(x0), len(x1), len(x2))      59 59 59
print(distance(x0, x1))               0.6251314473618036
print(distance(x0, x2))               1.0010412477871031
```

63

63

Types of measurements

- Increasing functionality ↓
- Measurements can be one of four different scales: *nominal*, *ordinal*, *interval*, or *ratio*. The type determines what operations makes sense with that type of data
 - Nominal*: data consists of *unordered labeled* categories
 - Country of birth, gender, car brand, marital status
 - Ordinal*: data consists of *ordered* and *labeled* categories
 - Top four teams in the world cup, Amazon reviews (😞 😐 😊 😄 😁)
 - Interval*: a numerical scale that allows you to interpret intervals between any two values in a consistent manner, with no natural notion of true zero
 - Intelligence quotient, temperature (°C or °F)
 - Ratio*: Measurement scale has a true zero (absence of the quantity being measured)
 - Age, height, weight, length, speed, number of alcoholic drinks per week
 - Each category adds functionality to the earlier categories

Exercise. What scale do US zip codes, time of an event, GPA belong to?

64

64

Measurement scale properties

Categories with no ordering	Nominal Data	Qualitative data				
Order among categories	Ordinal Data					
Subtraction is meaningful	Interval Data	Quantitative data				
Notion of zero is well-defined	Ratio Data					
Provides	Nominal	Ordinal	Interval	Ratio		
Frequency	✓	✓	✓	✓		
Mode	✓	✓	✓	✓		
Median		✓	✓	✓		
Mean			✓	✓		
Relational	=, ≠	=, ≠, <, >	=, ≠, <, >	=, ≠, <, >		
Arithmetic			+, −	+, −, ×, ÷		
Natural 0				✓		

- Measurements from a total order can be encoded numerically (with some restriction on meaningful operations)
 - In the “learning likelihood to graduate” example, the contribution of exam 2, measured as w_2x_2 is intuitive and interpretable
- Nominal data should *not* be encoded using a numeric scale (why?)
 - Zip code 80208 (DU) is *not* twice as large as zip code 40104 (Kentucky)

65

65

Encoding nominal data

- Techniques include *one-hot* and *bag-of-terms* encodings, with one separate weight for each possible nominal value (vs. one weight per feature)
- One-hot encoding** is used to encode categorical data, i.e., values from a collection of k mutually exclusive categories, e.g., *county-of-birth*, *home-zip-code*, etc.
 - A feature is encoded as a bit string of length k with exactly one bit set
 - Example.* If $\langle \text{Argentina, Belgium, Canada, ..., Uruguay, Vietnam, Zimbabwe} \rangle$ is our list of categories for the *country-of-birth* feature, Canada is encoded as $\langle 0, 0, 1, \dots, 0, 0, 0 \rangle$ and Zimbabwe as $\langle 0, 0, 0, \dots, 0, 0, 1 \rangle$
- Bag-of-terms** encoding is used to encode a set or multiset from a collection of k categories that are not necessarily mutually exclusive
 - An instance is encoded as a bit string of length k with any number of bits set
 - If a category can appear multiple times, integer counts can be used instead of bits
 - When the feature encodes a multiset of words, the representation is a *bag-of-words*
 - Example.* If the same list of countries is used to encode the feature *citizen-of* then a citizen of both Argentina and Canada is encoded as $\langle 1, 0, 1, \dots, 0, 0, 0 \rangle$

66

66

Example 1: Predicting graduation rates (revisited)

- DU offers a professional MS program in Data Science
 - Three proficiency exams are required, each with scores in $[0,100]$
 1. Programming Basics
 2. Calculus for Data Science
 3. Discrete Mathematics & Linear Algebra for Data Science
 - Based on historical data, that included exam scores, undergraduate GPA and majors, and more, 7 features were selected to compute a model, specified by a vector $\mathbf{w} = (w_0, w_1, w_2, w_3, w_4, w_5, w_6, w_7)$ that issues the following prediction:
 - Student likely to graduate if $w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7 > w_0$
 - Student unlikely to graduate if $w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7 < w_0$
- where $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ is the feature vector with:
- x_1, x_2, x_3 : proficiency exam scores; x_4 : undergraduate GPA
 - $x_5 = 1$ if student holds major in CS, 0 otherwise
 - $x_6 = 1$ if student holds major in Math or Statistics, 0 otherwise
 - $x_7 = 1$ if student holds any major *other* than CS, Math, or Statistics, 0 otherwise

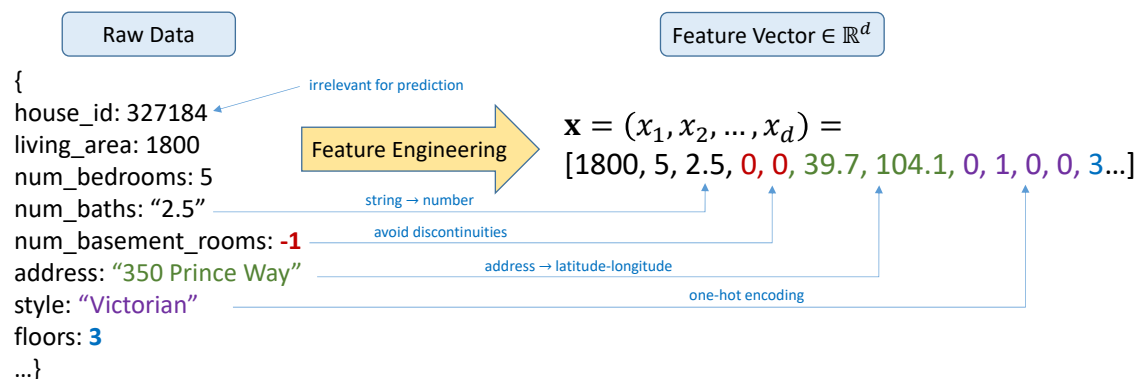
67

67

Example 2: How much is my house worth?

- Learning task: recommend the listing price for a house you want to sell
- Model specified by vector of weights: $\mathbf{w} = (w_0, w_1, \dots, w_d)$. For feature vector \mathbf{x} , the predicted price is:

$$\hat{f}(\mathbf{x}; \mathbf{w}) = w_0 + w_1x_1 + \dots + w_dx_d, \text{ } \mathbf{x} \text{ is the feature vector}$$



68

68

Transformations

- Often, to achieve certain effects, one applies selected functions to the features of each sample
Examples. $x_i \rightarrow x_i^2$, $x_i \rightarrow \sqrt{x_i}$, 0-1 normalization, z-score normalization, etc.
- What effect do the following transformations have?
 - Squaring the number of bedrooms
 - Taking the square root of the lot size
 - Scaling every feature to values in $[0, 1]$
- Normalization transforms all features of S to a standard scale
 - Equalizing feature scales prevents certain features from dominating others
 - Some learning algorithms (e.g., gradient descend) converge faster on normalized data
 - Mitigate the influence of outliers
 - Normalized data is easier to interpret by directly comparing weight contributions

69

69

Example 3: Recommending movies

- In a movie recommendation system, users and movies are represented using similarly structured vectors
 - User feature vectors are computed from past user ratings (user, movie, rating)
 - Movie feature vectors score categories (comedy, action, etc.) for each movie
 - How do use these vectors to predict if a user will like a movie?
 - Hint. Interpret the vectors as *directions* in \mathbb{R}^d

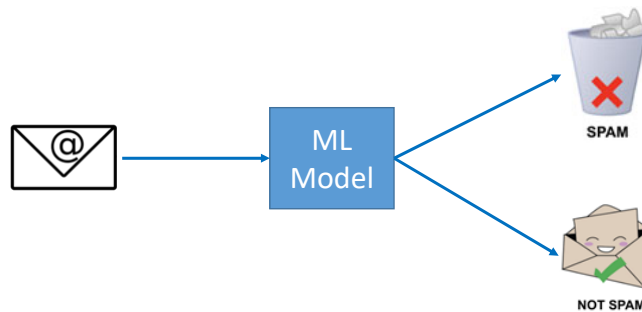
		likes action?	likes comedy	likes blockbusters	likes Brad Pitt
user	0.9	0.3	0.7	0.5
movie	0.4	0.9	0.5	0.0
		action content	comedy content	blockbuster?	Brad Pitt in it?

70

70

Exercise

- What kind of feature engineering would you use when designing a system to detect spam emails?
- What kind of feature vectors would you derive from the raw data consisting of arbitrarily long strings of text?
- What would be a reasonable *linear model* for spam detection?



71

71

Feature Extraction on Text

- Feature encoding on text is especially challenging (why?)
 - Text is unstructured, noisy, and arbitrarily long
- How do you represent a sample as a fixed dimensional vector in \mathbb{R}^D that takes into account the importance of different words?
 - *One-hot encoding*: for each word, a Boolean indicates presence or absence
 - *Bag-of-words*: for each word, store a count of number of times it appears
 - *TF-IDF*: give higher (resp. lower) weight to rare (resp. common) words

Example. Encode the text *'The fat cat likes the big cat'* as a feature vector, we first identify a **vocabulary** that includes “relevant” words in a corpus of text, and removes *stop words*, words considered “irrelevant” (e.g., ‘the’, ‘a’, etc.)

['big' 'cat' 'destroyed' 'dog' 'fat' 'hat' 'house' 'likes' 'people' 'sat' 'price']

one-hot encoding: [1 1 0 0 1 0 0 1 0 0 0]

bag-of-words encoding: [1 2 0 0 1 0 0 1 0 0 0]

Vocabulary

72

72

Term Frequency-Inverse Document Frequency (TF-IDF)

- Each word of a corpus (collection of documents) gets a number (ratio scale) representative of the importance of that word in a particular document
 - Documents with similar *relevant* words are expected to get similar vectors
- TF-IDF** accounts both for *common words* occurring in many documents and for *important words* appearing in few documents

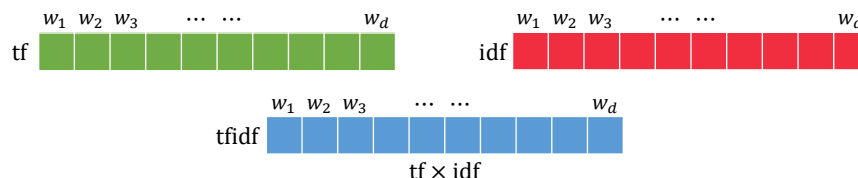
Example: in a corpus of sports article the word 'player' is very common, but the word 'touchdown' is more germane to football articles
- Cannot be computed in isolation for a single document, it requires a big corpus of documents (e.g., a large set of Wikipedia articles) to be meaningful
- Provides a tradeoff between local frequency (in 1 doc) and global rarity (in all docs)
 - The weight of a word w is proportional to its frequency in the document (TF term) and inversely proportional to the number of documents containing it (IDF term)
 - Word w is *important* in document s if it appears frequently in s (*common locally*), but appears infrequently in the corpus (*uncommon globally*)

73

73

Computing the TF-IDF Vectors

- For each document s and word $w \in s$ compute:
 - The term frequency $\text{tf}(w, s)$ is the raw count (or relative frequency) of w in s . Thus, the tf vector of s is the bag-of-words B of s (or, alternatively, $B/|s|$, where $|s|$ is the number of words in s)
 - The inverse document frequency of w is $\text{idf}(w) = \log\left(\frac{n}{n_w}\right)$, where n is the total number of docs and n_w is the number of docs in the corpus that contain w
 - The TF-IDF of w in s is $\text{tfidf}(w, s) = \text{tf}(w, s) \cdot \text{idf}(w)$
- $\text{idf}(w)$ acts as a weight on the term frequency of w : if n_w is large, this weight is small; as n_w gets smaller, the weight gets bigger
- Similar but slightly different formulas are used by different libraries [scikit-learn TF-IDF](#)



74

74

Example

- How important is a word such as “Romeo”, “salad”, or “forest” when trying to identify one of Shakespeare’s $n = 37$ plays
- Document frequency (df) and inverse document frequency (idf) of a few words

Word	df	idf
Romeo	1	1.568
salad	2	1.267
Falstaff	4	0.966
forest	12	0.489
battle	21	0.246

Word	df	idf
wit	34	0.037
fool	36	0.012
good	37	0.000
sweet	37	0.000
the	37	0.000

$$\text{idf}(w) = \log \left(\frac{n}{\text{df}(w)} \right)$$

of docs ↓
of docs containing w ↑

- ‘Romeo’, ‘salad’, and ‘Falstaff’ appear in few plays and, consequently, are more informative than words like ‘good’, ‘sweet’, and ‘the’ which are very common
 - More informative words get a higher idf weight

75

75

Measuring Similarity

- How do you measure similarity between two pieces of text?

<i>The fat cat sat in the hat</i>	[[0 1 0 0 1 1 1 0 0 1 0]]	[[0.0 0.37 0. 0.0 0.37 0.37 0.55 0.0 0.0 0.55 0.0]]
<i>The cat destroyed the hat</i>	[[0 1 1 0 0 1 0 0 0 0 0]]	[[0.0 0.49 0.73 0.0 0.0 0.49 0.0 0.0 0.0 0.0 0.0]]
<i>The hat is too big</i>	[[1 0 0 0 0 1 0 0 0 0 1]]	[[0.49 0.0 0.0 0.0 0.0 0.49 0.0 0.0 0.0 0.0 0.73]]
<i>The fat cat likes the big cat</i>	[[1 2 0 0 1 0 0 1 0 0 0]]	[[0.37 0.73 0.0 0.0 0.37 0.0 0.0 0.44 0.0 0.0 0.0]]
<i>The big dog likes fat people</i>	[[1 0 0 1 1 0 0 1 1 0 0]]	[[0.36 0.0 0.0 0.53 0.36 0.0 0.0 0.43 0.53 0.0 0.0]]

Vocabulary = ['big' 'cat' 'destroyed' 'dog' 'fat' 'hat' 'in' 'likes' 'people' 'sat' 'too']

- After feature extraction, each text fragment is encoded as a vector $u \in \mathbb{R}^d$
- Some choices to measure similarity include:

– Euclidean distance. $d(u, v) = \sqrt{\text{dot}(u - v, u - v)}$

– Cosine distance. $d(u, v) = 1 - \underbrace{\text{dot}(u, v)}_{\text{cosine similarity}}$ (requires $\|u\| = \|v\| = 1$)

[scikit-learn TF-IDF](#)

76

76

The need for feature selection

- The term **curse of dimensionality** refers to the fact that the running time of a learning algorithm grows very fast, often exponentially, with the dimensionality (number of features) of the input data
- It may not be feasible (or wise) to use all features available for each sample
 - A greyscale image may easily require over a million bytes (3x more for color)
 - Many features may be irrelevant or redundant
 - The price of a house hardly depends on whether a microwave is included or not
 - To identify the breed of a dog from a picture, we used 59 values, instead of 25,080
 - The more features you include in your model, the more likely it is to overfit
- To keep computation times within budget and the learned model useful, we must first reduce the input data to short feature vectors
- It is the job of the ML practitioner to find a sensible subset of features, in the appropriate format, to use as input to the learning algorithm
 - A good design always starts with a thorough exploration of the available data

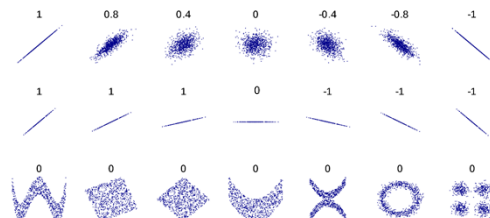
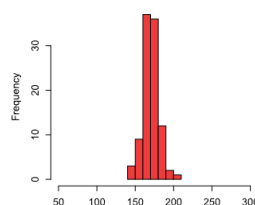
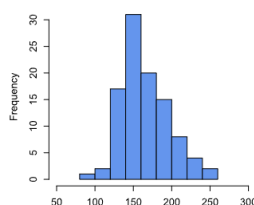


77

77

Basic Statistics

- Before you attempt any ML, you should become familiar with the data through visualization of feature distribution(s) and collection of basic statistics: mean, standard deviation (std), correlation coefficients
 - The *standard deviation* is a measure of the amount of dispersion of a set of values around their mean
 - The *correlation coefficient* is a measure of the strength (and direction) of the linear relation between two features. It has a value between in $[-1,1]$



- How does this information help when engineering the features to use?

78

78

Basic Statistics

- Since we don't have access to the source distribution \mathcal{D} , we instead rely on the *sample* means, standard deviations, and correlation coefficients
- For a set of N feature vectors, with $\mathbf{x}_i = (x_{i1}, \dots, x_{id})$, we have:

Mean of feature j : $\bar{x}_j = \frac{1}{N} \sum_{i=1}^N x_{ij}$

Covariance of features j and k : $s_{jk} = \frac{1}{N} \sum_{i=1}^N (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)$

Standard deviation of feature j : $s_j = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_{ij} - \bar{x}_j)^2}$

Correlation coefficient of features j and k : $r_{jk} = \frac{s_{jk}}{s_j \cdot s_k}$

Exercise. Create a set S with N feature vectors in \mathbb{R}^4 , store them in a $N \times 4$ matrix and use NumPy to compute the correlation matrix of the features

79

79

Example: Diagnosing appendicitis

- Appendicitis is the most common cause of abdominal pain
- Accurate diagnosis is difficult, with up to 20% rate of false positives
- Data: historical records of 473 patients. Each record includes 15 symptom features and outcome (1=appendicitis, 0=no).
- Goal is to design an ML algorithm to yield a correct diagnosis for new unseen data. This is done by computing w_0, w_1, \dots, w_{15} :

$$x_{16} = \begin{cases} 1 & \text{if } w_1 x_1 + \dots + w_{15} x_{15} > w_0 \\ 0 & \text{otherwise} \end{cases}$$

Example. patient 1 = (26, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 37.9, 38.8, 23100, 0, 1)

Exercise. Can you find at least one design flaw in the above encoding?

Var. num.	Description	Values
1	Age	Continuous
2	Sex (1 = male, 2 = female)	1, 2
3	Pain quadrant 1	0, 1
4	Pain quadrant 2	0, 1
5	Pain quadrant 3	0, 1
6	Pain quadrant 4	0, 1
7	Local muscular guarding	0, 1
8	Generalized muscular guarding	0, 1
9	Rebound tenderness	0, 1
10	Pain on tapping	0, 1
11	Pain during rectal examination	0, 1
12	Axial temperature	Continuous
13	Rectal temperature	Continuous
14	Leukocytes	Continuous
15	Diabetes mellitus	0, 1
16	Appendicitis	0, 1

80

80

Example: Correlation Matrix K

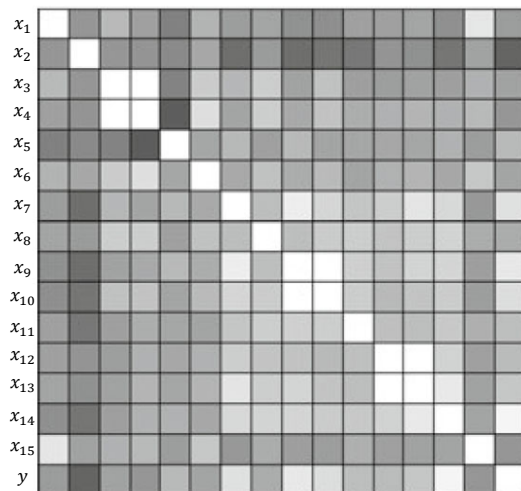
1.	-0.009	0.14	0.037	-0.096	0.12	0.018	0.051	-0.034	-0.041	0.034	0.037	0.05	-0.037	0.37	0.012
-0.009	1.	-0.0074	-0.019	-0.06	0.063	-0.17	0.0084	-0.17	-0.14	-0.13	-0.017	-0.034	-0.14	0.045	-0.2
0.14	-0.0074	1.	0.55	-0.091	0.24	0.13	0.24	0.045	0.18	0.028	0.02	0.045	0.03	0.11	0.045
0.037	-0.019	0.55	1.	-0.24	0.33	0.051	0.25	0.074	0.19	0.087	0.11	0.12	0.11	0.14	-0.0091
-0.096	-0.06	-0.091	-0.24	1.	0.059	0.14	0.034	0.14	0.049	0.057	0.064	0.058	0.11	0.017	0.14
0.12	0.063	0.24	0.33	0.059	1.	0.071	0.19	0.086	0.15	0.048	0.11	0.12	0.063	0.21	0.053
0.018	-0.17	0.13	0.051	0.14	0.071	1.	0.16	0.4	0.28	0.2	0.24	0.36	0.29	-0.0001	0.33
0.051	0.0084	0.24	0.25	0.034	0.19	0.16	1.	0.17	0.23	0.24	0.19	0.24	0.27	0.083	0.084
-0.034	-0.17	0.045	0.074	0.14	0.086	0.4	0.17	1.	0.53	0.25	0.19	0.27	0.27	0.026	0.38
-0.041	-0.14	0.18	0.19	0.049	0.15	0.28	0.23	0.53	1.	0.24	0.15	0.19	0.23	0.02	0.32
0.034	-0.13	0.028	0.087	0.057	0.048	0.2	0.24	0.25	0.24	1.	0.17	0.17	0.22	0.098	0.17
0.037	-0.017	0.02	0.11	0.064	0.11	0.24	0.19	0.19	0.15	0.17	1.	0.72	0.26	0.035	0.15
0.05	-0.034	0.045	0.12	0.058	0.12	0.36	0.24	0.27	0.19	0.17	0.72	1.	0.38	0.044	0.21
-0.037	-0.14	0.03	0.11	0.11	0.063	0.29	0.27	0.27	0.23	0.22	0.26	0.38	1.	0.051	0.44
0.37	0.045	0.11	0.14	0.017	0.21	-0.0001	0.083	0.026	0.02	0.098	0.035	0.044	0.051	1.	-0.0055
0.012	-0.2	0.045	-0.0091	0.14	0.053	0.33	0.084	0.38	0.32	0.17	0.15	0.21	0.44	-0.0055	1.

- Values of r_{ij} close to -1 or $+1$ suggest strong correlation between features i and j

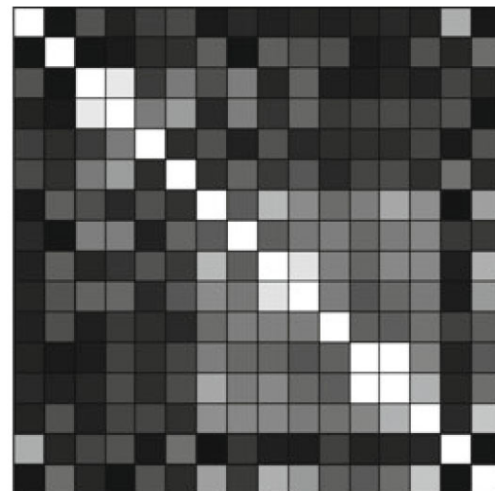
81

81

Visualizing the Correlation Matrix



$K_{ij} = -1$: black, $K_{ij} = 1$: white



$|K_{ij}| = 0$: black, $|K_{ij}| = 1$: white

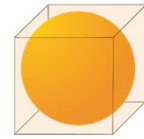
- What does this suggest in terms of relevant or redundant features?

82

82

The curse of dimensionality

- High dimensional data poses serious challenges for machine learning
- The most important assumption underlying many ML techniques is that samples with similar feature vectors are similar
 - We assume similar people behave similarly (like the same movies, repay loans, etc.)
 - What if no one resembles you very much or everyone resembles you equally
- High dimensional spaces are rather empty
 - The volume of a d -dimensional cube $C_d(r)$ of width r is r^d
 - For $r = 1$, $\text{vol}(C_d(1)) = 1$, but for $0 < \lambda < 1$, $\text{vol}(C_d(\lambda)) = \lambda^d \cdot \text{vol}(C_d(1)) = \lambda^d$
 - $\lim_{d \rightarrow \infty} \text{vol}(C_d(\lambda r)) / \text{vol}(C_d(r)) = 0 \Rightarrow$ most points lie close to the outer shell!
 - The volume of d -dimensional sphere $S_d(r)$ of radius r is $(\pi^{d/2} / \Gamma(d/2 + 1)) r^d$
 - $\lim_{d \rightarrow \infty} \text{vol}(S_d(r)) / \text{vol}(C_d(2r)) = 0 \Rightarrow$ most points lie outside the inscribed sphere
 - If we want points within distance δ of a point, we find fewer and fewer as d increases
 - All points are at similar distance from each other, and all are good candidates for NN!
- Fortunately, latent structure in data is a great dimensionality reducer



83

83

Types of ML Methods and Models

- Supervised learning
 - Regression: logistic and linear
 - Decision trees
 - Artificial neural networks (ANNs)
 - Support vector machines (SVMs)
- Unsupervised learning
 - k -means clustering
 - Anomaly detection
 - Dimensionality reduction
 - Self-organizing maps
- Reinforcement learning
 - Q-learning
 - State-action-reward-state-action

84

84

Learning Methods: Supervised vs Unsupervised

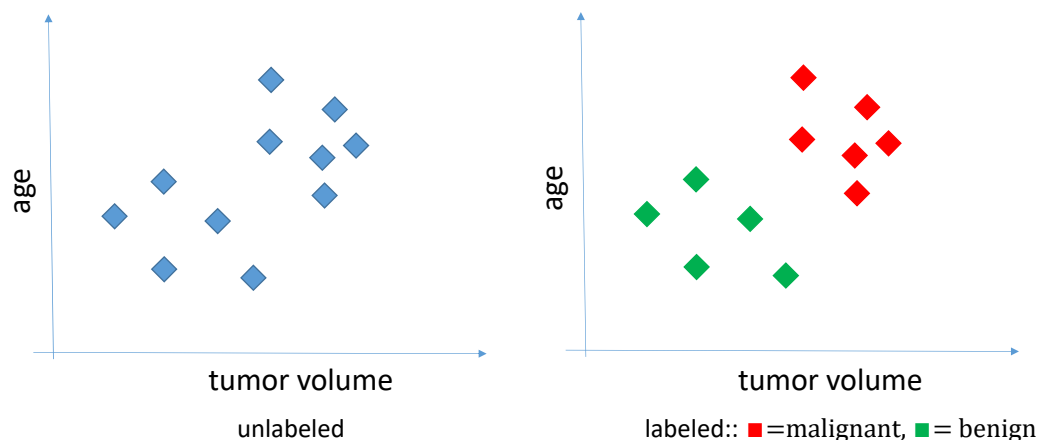
- In **supervised learning**, training data is labeled with correct target values
 - Goal is to provide a function g that predicts the target value f on new inputs
 - A feedback loop is used to assess prediction results and improve performance
 - Example.* Given Denver house attributes and prices over the last year, predict selling price for your house
- In **unsupervised learning**, data is unlabeled
 - Goal is to find latent structure in the data, i.e., suggest similarities or unusual observations, without knowing the effect of the features involved
 - Can infer structure by simplifying or by clustering the data based on relationships among the variables used to describe it
 - No feedback that allows you to compare computed results to the ground truth
 - Example.* Given a collection of one million different newspaper articles, group them into clusters of related articles

85

85

Example

- You are given some data on tumor characteristics: size and patient age



86

86

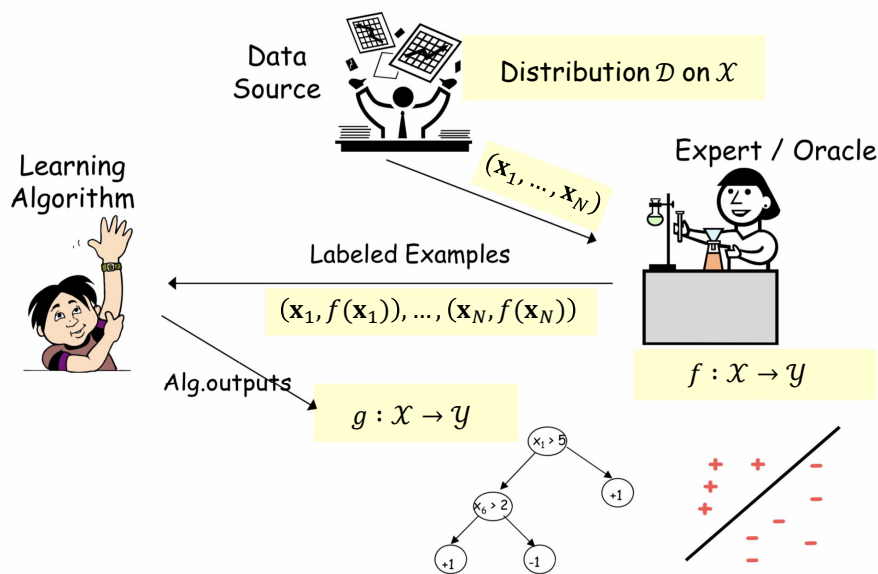
Supervised learning

- Let \mathcal{X} and \mathcal{Y} be sets and $f: \mathcal{X} \rightarrow \mathcal{Y}$ a target function (ground truth) to be learned
- Goal is to compute a function g (the model) that matches or approximates the mapping f , based on a correctly labeled set $S \subset \mathcal{X}$ of **training data**
 - Model g is selected from a **hypothesis space** \mathcal{H} of candidate models
 - We will explore different learning algorithms to select g
 - S consists of N iid samples from \mathcal{D} , a fixed but unknown probability distribution over \mathcal{X}
- After feature extraction, each input becomes a **feature vector** $\mathbf{x} \in \mathbb{R}^d$, a list of d real numbers that capture relevant characteristics of the input instance
 - Labels y may be *continuous* (e.g., the price of a house, # of retweets of a tweet) or *discrete* (e.g., spam or not spam, type of traffic sign, etc.), corresponding to the main types of supervised learning: **regression** or **classification**, respectively
- Function g is expected to predict the correct output $f: \mathcal{X} \rightarrow \mathcal{Y}$ for all $\mathbf{x} \in \mathcal{X}$, including unseen instances, i.e., it is expected to **generalize**
 - Assumes that, once deployed, new data comes from the same distribution \mathcal{D} as S

87

87

Workflow of supervised learning



M.F. Balcan

88

88

Sample applications

- An email client that removes incoming email that appears to be spam
- Houses that learn from past data to optimize energy costs based on usage patterns of their occupants
- Medical software that alerts a doctor that a tumor might be malignant
- An app that takes an audio clip and tells you the title of the song
- Software that detects if a product flowing through an assembly line is defective (e.g., has a scratch or dent) or not
- An app that predicts the age of a person
- A function that predicts whether an online shopper is likely to click on an ad
- A self-driving car that adjusts the wheel turn angle to keep the car on the road

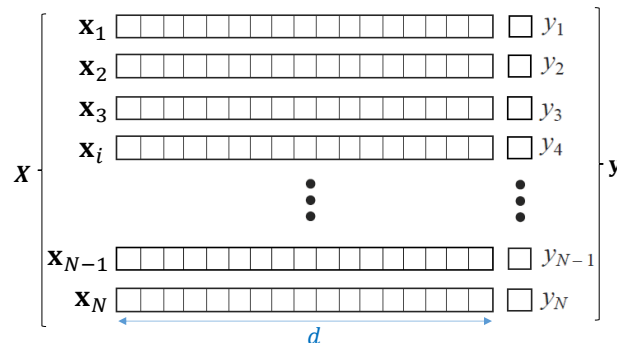
*Can you identify \mathcal{X} and \mathcal{Y} for each application above?
What kind of learning task (classification or regression) is it?*

89

89

Notation

- The training set S consists of N labeled feature vectors in \mathbb{R}^d , often organized in a $N \times d$ **design matrix** X and $N \times 1$ **target vector** y



- Use any of the following formats to refer to the i^{th} training sample:

$$\mathbf{x}_i = (x_i(1), \dots, x_i(d)) = (x_{i1}, \dots, x_{id}) = (\mathbf{x}_i^{(1)}, \dots, \mathbf{x}_i^{(d)})$$
- For convenience, will often use **tall vectors**: $\mathbf{x}_i = (1, x_{i1}, \dots, x_{id})$
- Either $g(\mathbf{x})$ or $\hat{f}(\mathbf{x})$ denote the predicted value on sample \mathbf{x}

90

90

Key assumption of *all* supervised learning systems

- Real-world data exhibits some unknown underlying pattern that is hard to discern or to program
 - The pattern to detect is a function $f: \mathcal{X} \rightarrow \mathcal{Y}$, the “ground truth” we wish to learn
 - The input is a **training set** of N data points/vectors $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^d$, drawn independently and identically (iid) from a distribution \mathcal{D} on the set \mathcal{X} of all possible samples, as well as the corresponding ground-truth labels $f(\mathbf{x}_1), \dots, f(\mathbf{x}_N) \in \mathcal{Y}$
- Good performance is sought by creating a statistical model of the pattern
 - The model is a function $g: \mathcal{X} \rightarrow \mathcal{Y}$, the output of the ML algorithm, its best guess on f
- For this to be feasible, there must be enough (training) data about the pattern
 - How much is “enough”? *Depends on the size and nature of the hypothesis space \mathcal{H}*
- For this approach to be successful, future data must be “similar” to the one used while learning the pattern
 - Success is measured by how close f and g are on arbitrary $\mathbf{x} \in \mathcal{X}$
 - New inputs \mathbf{x} are assumed to be drawn from the same distribution \mathcal{D}

91

91

Binary Classification

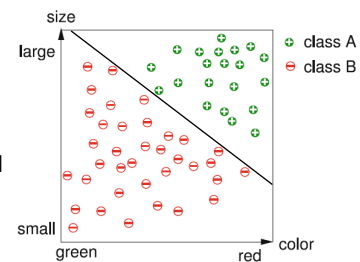
- The most basic classification task is one with two classes: $\{-1, +1\}$ (or $\{0,1\}$)
 1. After feature-engineering the input, data points correspond to vectors in \mathbb{R}^d
Example. The bag-of-words representation of emails for the 10,000 most common words
 2. A “ground truth” function $f: \mathbb{R}^d \rightarrow \{-1, +1\}$ defines the correct class of each feature vector. This unknown function is what we hope to learn
Example. f indicates whether an email is spam or not spam
 3. Our training set S is a sample from an unknown statistical distribution \mathcal{D} :
 $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^d$, with each \mathbf{x}_i an iid sample from \mathcal{D} , with the corresponding ground truth labels $f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)$
Example. \mathcal{D} is an unknown distribution over all possible emails (this induces a distribution on \mathbb{R}^d), and S is a set of emails correctly labeled by an expert as spam or not
 4. The output of our algorithm is a *predictor* $g: \mathbb{R}^d \rightarrow \{-1, +1\}$, a “best guess” about f
 5. The ML algorithm succeeds if for a random point $\mathbf{x} \in \mathcal{D}$, and user-specified ϵ , the probability that \mathbf{x} misclassified is at most ϵ (will address how to achieve this later)

92

92

Warmup: a first look at linear classifiers

- To simplify our discussion, we initially assume that $d = 2$
- A major fruit company uses sensor measurements to sort apples into one of two market types (classes): A (+1) or B (−1). For illustration, assume that the two relevant measures are *color* and *size*
- For each of N apples (the training set), correctly labeled with A or B by an expert, you create a feature vector $\mathbf{x} = (x^{(1)}, x^{(2)})$ where $0 < x^{(1)} < 1$ is the degree of “redness”, and $x^{(2)}$ measures the diameter of the apple
- Your job is to design a *binary classifier* for apples
- What is a reasonable hypotheses space and how do you parameterize it?
- How do you classify new, not seen before, apples?

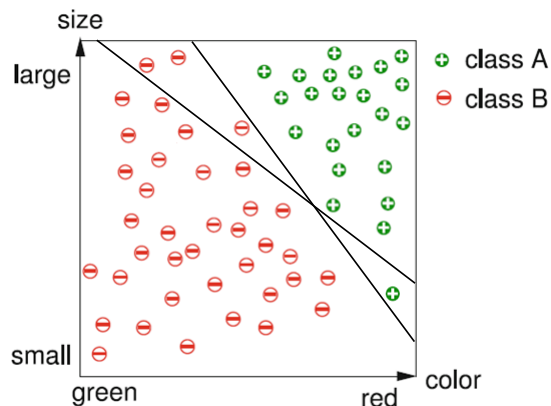
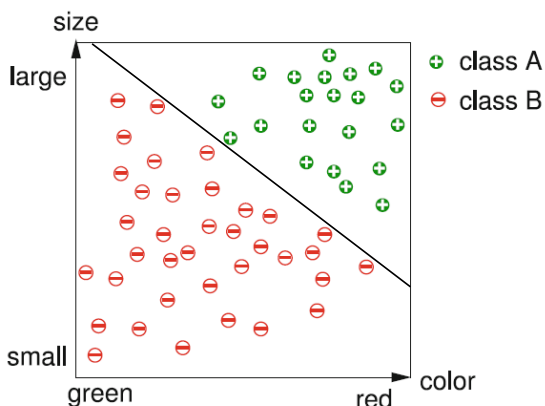


93

93

A Linear Hypothesis Space

- What happens to your hypothesis space with a different training set?

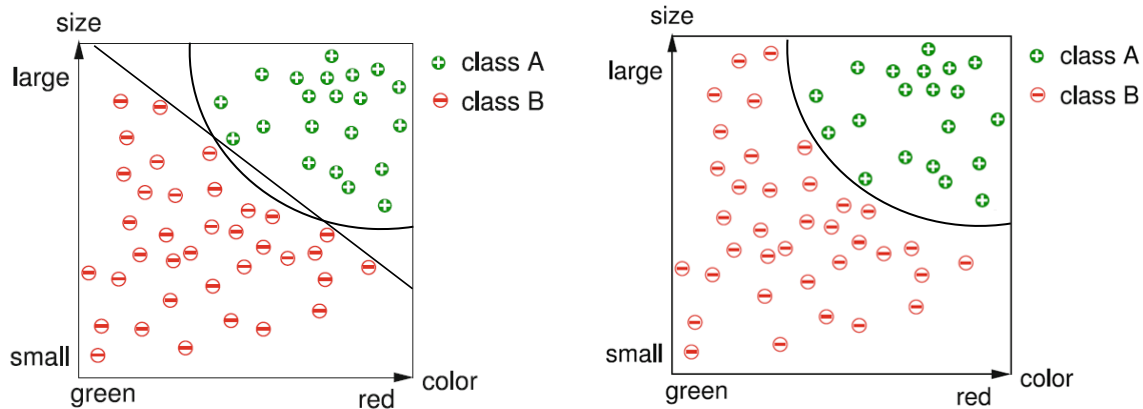


94

94

Quadratic Hypothesis Space

- If the ground truth f and \mathcal{H} are both quadratic, then there is a model g that classifies *all* training sets correctly

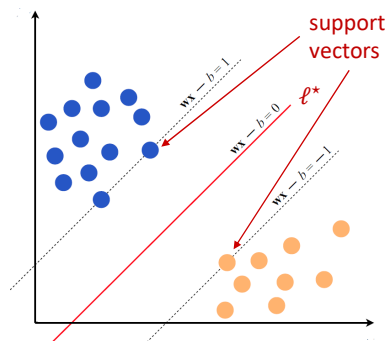


95

95

Finding a linear classifier

- The hypothesis space \mathcal{H} is the set of lines in the plane
 - Some lines classify the training set S correctly, and some don't
 - Among the lines that classify S correctly, we prefer the line ℓ^* that maximizes the distance to the nearest points of S (the **support vectors**, at least one on each side of ℓ^*)
 - Line ℓ^* is equidistant from *all* support vectors (why?)
 - Any fixed line $\mathbf{w} \cdot \mathbf{x} = b$, has multiple parameterizations (why?)
 - Let's choose one with $|\mathbf{w} \cdot \mathbf{s} - b| = 1$, for all support vectors \mathbf{s}

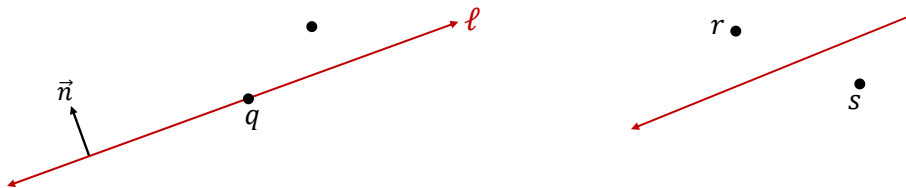


96

96

Review

- What is the general equation of a line ℓ ?
- How do you find a vector \vec{n} normal to a line ℓ ?
Hint. What information does the dot product of two vectors give you?
- What is the dot product of \vec{n} with a point q on the line?
- Interpret the dot product of \vec{n} with a point q *not* on the line
- How do you find a line ℓ^* that separates points r and s and is farthest from both
- Can you generalize to points and planes in \mathbb{R}^3 ?

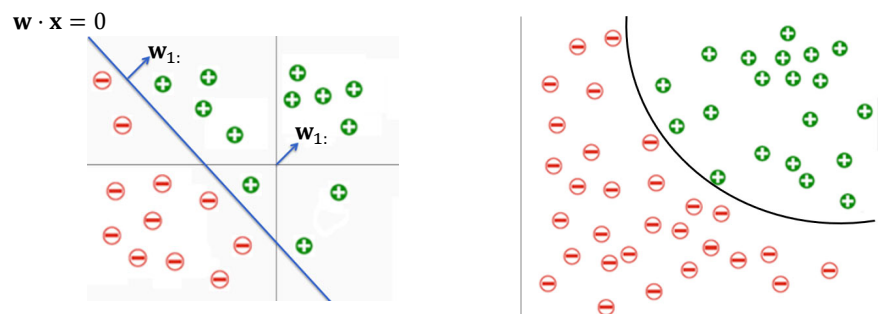


97

97

How does a linear classifier classify?

- Consider a linear classifier $\mathbf{w} = (w_0, w_1, \dots, w_d)$ in \mathbb{R}^{d+1} with $\mathcal{X} \subset \mathbb{R}^d$
 - For convenience, we use $\mathbf{w}_{1:} = (w_1, \dots, w_d)$
 - Tall vector $\mathbf{x} \in \mathbb{R}^{d+1}$ is classified as $+1$ if $\mathbf{w} \cdot \mathbf{x} > 0$ and -1 if $\mathbf{w} \cdot \mathbf{x} < 0$
 - Since we only care about the sign, not the magnitude, predict $\hat{y} = \text{sgn}(\mathbf{w} \cdot \mathbf{x})$
 - The classifier \mathbf{w} represents a hyperplane that separates \mathbb{R}^d into two half-spaces

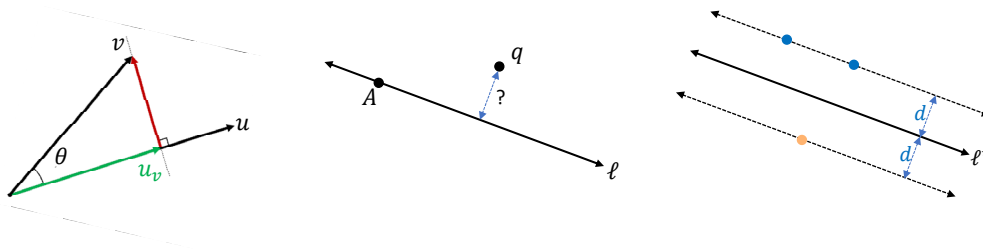


98

98

Exercise

- What is the distance from a point q to a line $\ell : \mathbf{w} \cdot \mathbf{x} - b = 0$?
Hint. Use $u \cdot v$ to find the projection u_v of vector v onto another vector u
- Show that if ℓ^* maximizes the distance to its nearest sample point, then ℓ^* admits at least 3 support vectors, with at least one on each side of ℓ^*
- Show that the optimal ℓ^* can always be chosen so that $|\mathbf{w} \cdot \mathbf{s} - b| = 1$

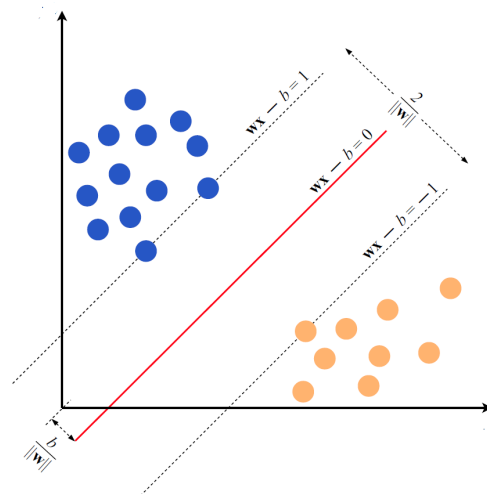


99

99

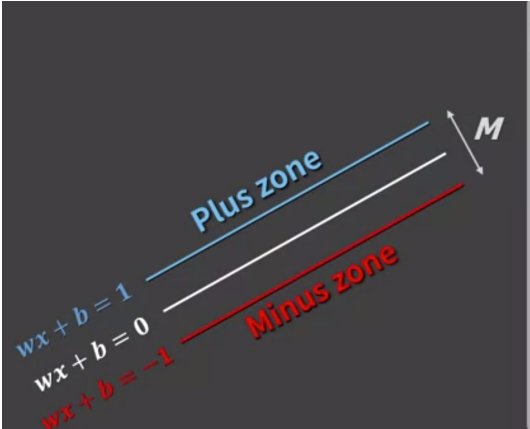
Binary Classification...

- The optimal line $\ell: \mathbf{w} \cdot \mathbf{x} - b = 0$ can be parameterized so that:
 - Any support vector s satisfies $\mathbf{w} \cdot \mathbf{s} - b = \pm 1$
 - The distance from ℓ to s is $1/\|\mathbf{w}\|$
- To maximize the distance to the support vectors, we need to minimize w
- Given a training set $S = S_1 \cup S_2$ with two classes (blue and yellow), how do you find the optimal splitting line ℓ ?
 - We consider an algorithm based on a transformation known as dual transform
- What we have is an optimization problem:
 $\min w_1^2 + w_2^2$, subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \forall i$



100

100



$$\begin{aligned} \text{③ } |x^+ - x^-| &= M \rightarrow |x^+ - x^-| = M \\ &= \lambda |w| \\ \text{④ } x^+ &= x^- + \lambda w \\ \text{⑤ } w(x^- + \lambda w) + b &= 1 \\ &\rightarrow wx^- + \lambda w \cdot w + b = 1 \\ \text{⑥ } -1 + \lambda w \cdot w &= 1 \rightarrow \lambda = \frac{2}{w \cdot w} \end{aligned}$$

Given $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ where $y_i \in \{-1, +1\}$

$$\max_{w, b} \frac{2}{|w|} \quad \text{subject to } y_i(w \cdot x_i + b) \geq 1 \quad (i = 1, \dots, n)$$

or

$$\min_{w, b} \frac{1}{2} w \cdot w \quad \text{subject to } y_i(w \cdot x_i + b) \geq 1 \quad (i = 1, \dots, n)$$

101

101

An iterative improvement algorithm

- Relax the requirements: find any separating line such that $w \cdot x_i > 0$ for positive samples and $w \cdot x_i < 0$ for negative ones

Precondition: instance x_i is given as a *tall vector* $x_i = (1, x_{i1}, \dots, x_{id})$

1. Initialize w with random or arbitrary values, e.g., $w := 0$
2. Cycle through the training data until every item is correctly classified or a maximum number of cycles has been reached
3. Only update w on misclassified instances
4. **if** x_i is misclassified:
 5. **if** x_i is a positive instance ($y_i = +1$ but misclassified as -1):
 6. $w = w + x_i$
 7. **if** x_i is a negative instance ($y_i = -1$ but misclassified as $+1$):
 8. $w = w - x_i$
9. **return** w

102

102

Exercise

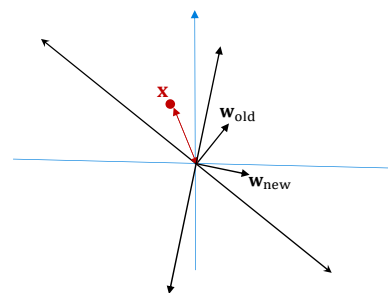
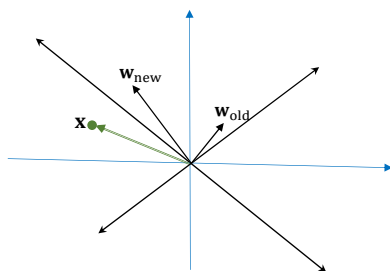
1. Explain why the updates to the weight vector (lines 6 and 8) increase the likelihood that a misclassified vector is correctly classified next time around
2. Explain how to condense lines 4-8 into a single conditional and update to \mathbf{w}
3. Argue that the algorithm converges iff the two classes are separable by a line (one direction is hard, the other easy) and loops forever otherwise
4. As described, the algorithm may take a long time to converge. This is partly because some misclassified points result in corrections that are too big. Propose a simple heuristic to improve the running time
5. A better initial guess for \mathbf{w} results in faster convergence. Instead of using a predetermined or random weight vector, suggest better initial guesses for \mathbf{w}
6. Explain how to combine an optimal line ℓ^* once a separating line ℓ is known
7. Analyze experimentally the impact on performance from using the heuristics in (3) and (4) concurrently and separately

103

103

Geometric interpretation

- Suppose x is $+1$ but classified -1 : we update $\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} + \mathbf{x}$
- If x is -1 but classified $+1$: we update $\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} - \mathbf{x}$



- In practice, may want to attenuate the update vector with a carefully chosen **hyperparameter** α : $\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} \pm \alpha \cdot \mathbf{x}$
 \uparrow learning rate

104

104

Convergence

- We show that if there are lines separating the positive from the negative samples, the algorithm will always find one such separating line

Theorem. Suppose there exist $\gamma > 0$ and parameter vector \mathbf{w}^* with $\|\mathbf{w}^*\| = 1$ such that $y_i(\mathbf{w}^* \cdot \mathbf{x}_i) \geq \gamma$ for $i = 1, \dots, N$ (i.e., \mathbf{w}^* has 0 training error). Then our algorithm makes at most R^2/γ^2 errors, where $R = \max_{1 \leq i \leq N} \|\mathbf{x}_i\|$.

Note: an error occurs if in some iteration, $y_i \neq \text{sign}(\mathbf{w} \cdot \mathbf{x}_i)$ for some $1 \leq i \leq N$.

Proof. Let \mathbf{w}^i denote the value of \mathbf{w} when the i^{th} error occurs, with $\mathbf{w}^1 = \mathbf{0}$.

Then, assuming the k^{th} mistake happens on sample t , $\mathbf{w}^{k+1} = \mathbf{w}^k + y_t \mathbf{x}_t$, and
 $\mathbf{w}^{k+1} \cdot \mathbf{w}^* = (\mathbf{w}^k + y_t \mathbf{x}_t) \cdot \mathbf{w}^* = \mathbf{w}^k \cdot \mathbf{w}^* + y_t \mathbf{x}_t \cdot \mathbf{w}^* \geq \mathbf{w}^k \cdot \mathbf{w}^* + \gamma$

Since $\mathbf{w}^1 = \mathbf{0}$, $\mathbf{w}^2 \cdot \mathbf{w}^* \geq \mathbf{w}^1 \cdot \mathbf{w}^* + \gamma = \gamma$, $\mathbf{w}^3 \cdot \mathbf{w}^* \geq \mathbf{w}^2 \cdot \mathbf{w}^* + \gamma \geq 2\gamma$, ...

By induction, $\mathbf{w}^{k+1} \cdot \mathbf{w}^* \geq k\gamma$ (Eq.1)

105

105

Convergence...

Recall the *Cauchy-Schwarz inequality*: arbitrary vectors \mathbf{u} and \mathbf{v} satisfy

$$(\mathbf{u} \cdot \mathbf{u})(\mathbf{v} \cdot \mathbf{v}) \geq (\mathbf{u} \cdot \mathbf{v})^2$$

Replacing $\mathbf{u} = \mathbf{w}^{k+1}$ and $\mathbf{v} = \mathbf{w}^*$ yields $(\|\mathbf{w}^{k+1}\|)(\|\mathbf{w}^*\|) \geq \mathbf{w}^{k+1} \cdot \mathbf{w}^*$

(Eq.1) and $\|\mathbf{w}^*\| = 1$ imply $\|\mathbf{w}^{k+1}\| \geq \mathbf{w}^{k+1} \cdot \mathbf{w}^* \geq k\gamma$ (Eq.2)

$\|\mathbf{w}^{k+1}\|^2 = \|\mathbf{w}^k + y_t \mathbf{x}_t\|^2 = \|\mathbf{w}^k\|^2 + y_t^2 \|\mathbf{x}_t\|^2 + 2y_t \mathbf{x}_t \cdot \mathbf{w}^k \leq$
 $\|\mathbf{w}^k\|^2 + \|\mathbf{x}_t\|^2 \leq \|\mathbf{w}^k\|^2 + R^2$ (using (a) $y_t^2 = 1$, (b) $\|\mathbf{x}_t\| \leq R$, and
 (c) $y_t \mathbf{x}_t \cdot \mathbf{w}^k \leq 0$ because \mathbf{w}^k yielded an error on the t^{th} sample)

Inducting on k yields an upper bound $\|\mathbf{w}^{k+1}\|^2 \leq kR^2$ (Eq.3)

From (Eq.2) and (Eq.3) we get $k^2\gamma^2 \leq \|\mathbf{w}^{k+1}\|^2 \leq kR^2$ which forces $k \leq \frac{R^2}{\gamma^2}$,
 a finite upper bound on the maximum number of mistakes

106

106

Learning binary classes online

- The iterative improvement algorithm can be adapted for *online learning*
 - Samples come one at a time, algorithm is asked to predict label and only then is given the correct answer
 - Past mistakes are “water under the bridge”
 - Does not require assumption that data comes from a fixed distribution \mathcal{D}
 - In fact, distribution \mathcal{D} can change over time
 - Winnow’s learning algorithm is a variant that uses a *multiplicative* correction (α)
 - Effective on high dimensional data from $\{0,1\}^d$ with many irrelevant features
 - Ground truth f is the logical-or of $k \ll d$ features, e.g., 1-hot encoding of docs
1. Initialize weights: $w_1 = w_2 = \dots = w_d = 1$, $b = d$ (or $d/2$), and $\alpha > 1$ (e.g., 2)
 2. Predict **+1** if $\mathbf{w} \cdot \mathbf{x} = w_1 x_1 + \dots + w_d x_d > b$; else predict **-1**
 - errors { 3. If $\mathbf{w} \cdot \mathbf{x} \leq b$ but $y = +1$: for every feature i with $x_i = 1$, multiply w_i by α
 4. If $\mathbf{w} \cdot \mathbf{x} > b$ but $y = -1$: for every feature with $x_i = 1$, divide w_i by α

107

107

Convergence of Winnow’s algorithm

Theorem. If the target concept f is the logical-or of k variables, Winnow’s algorithm learns a linear model g after making $\mathcal{O}(k(1 + \log d))$ mistakes

Proof. We bound the number of false negatives and false positives separately.

False negatives ($\hat{y} = -1, y = +1$).

Each mistake on a positive sample must double at least one of the weights of f (one of the “relevant” weights), and a mistake on a negative one will *not* halve any of these weights. Furthermore, each relevant weight can be doubled at most $1 + \lg d$ times, since only weights that are less than d can ever be doubled.

Therefore, the algorithm makes at most $k(1 + \lg d)$ mistakes on positive examples.

False positives ($\hat{y} = +1, y = -1$).

Initially, the total weight summed over all features is d and is always non-negative.

Each mistake on a positive sample increases the total weight by at most d (since before doubling we had $w \cdot x < d$). On the other hand, each mistake on a negative sample decreases the total weight by at least $d/2$ (since before halving, we had $w \cdot x \geq d$).

Therefore, the number of mistakes made on negative samples is at most twice the number of mistakes made on positive samples.

108

108

Geometric Duality

- We can find the optimal line ℓ^* using a geometric transformation based on the fact that lines and points may each be specified by two numbers: a and b , or x and y , respectively
 - Allows a simple bijection in \mathbb{R}^2 between points and non-vertical lines
 - We transform one to the other while preserving some useful spatial relations
- Consider the mapping $\ell : y = ax - b \leftrightarrow p : (a, b)$

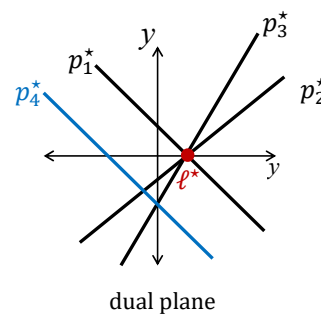
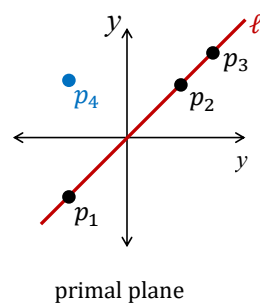
Example. $y = 3x + 2$ is mapped to $(3, -2)$ and vice versa
- Denote by $D(\ell) = \ell^* = p$ and $D(p) = p^* = \ell$
- The problem of finding a certain line for a set of points becomes a problem of finding a certain point for a set of lines

109

109

Duality Properties

- *Involution:* $p^{**} = (p^*)^* = p$ and $\ell^{**} = (\ell^*)^* = \ell$
- *Incidence preserving:* $p \in \ell$ iff $\ell^* \in p^*$
- *Order preserving:* p is above ℓ iff ℓ^* is above p^*



Exercise. Prove that duality is incidence and order preserving.

110

110

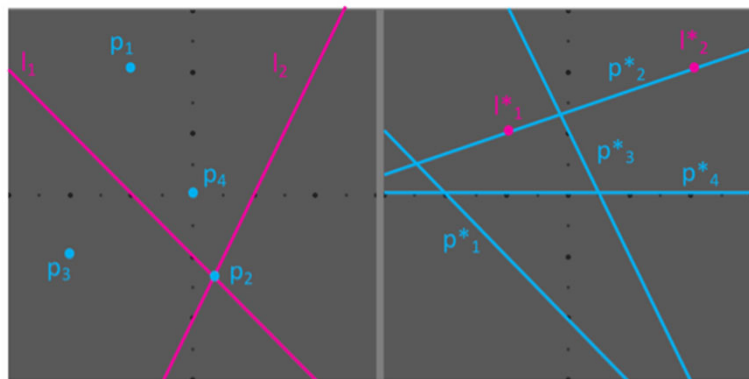
Exercise

- Create diagrams for both the primal and the dual planes that illustrate the following scenario :
 - Point p_1 lies above the line ℓ_1
 - Point p_2 lies on the line ℓ_1
 - Point p_2 lies on the line ℓ_2
 - Point p_3 lies below the line ℓ_1
 - Point p_4 lies above the line ℓ_2
- Are there any two such primal (or dual) diagrams topologically equivalent?

111

111

Duality Exercise



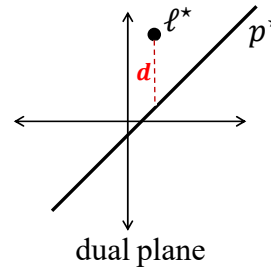
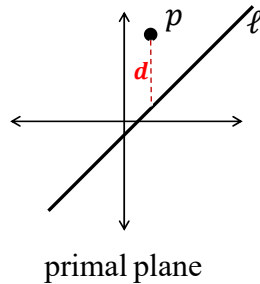
p_1 lies above ℓ_1	\Leftrightarrow	ℓ^*_1 lies above p^*_1
p_2 lies on ℓ_1	\Leftrightarrow	ℓ^*_1 lies on p^*_2
p_2 lies on ℓ_2	\Leftrightarrow	ℓ^*_2 lies on p^*_2
p_3 lies below ℓ_1	\Leftrightarrow	ℓ^*_1 lies below p^*_3
p_4 lies above ℓ_2	\Leftrightarrow	ℓ^*_2 lies above p^*_4

112

112

Duality Properties...

- *Distance preserving*: the vertical distance between p and ℓ is the same as that between ℓ^* and p^*



Exercise. What is the relation between the vertical distance between lines $\ell: y = ax + b$ and $g: y = ax + c$, and the length of segment $\overline{\ell^* g^*}$? In terms of the vertical distance d , what is the actual distance between ℓ and g ?

Exercise. Show that the bijection $(a, b) \leftrightarrow y = ax + b$ does not preserve all the duality properties.

113

113

Why Use Duality?

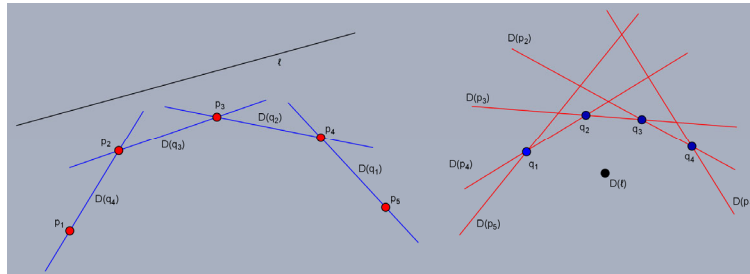
- Even though the dual transformation does not add any information, expressing a problem in the dual plane provides an alternative view of the problem, hopefully one for which a solution is easier to visualize
- Looking at the problem from a different angle may provide the insight needed to solve it
- Solution in dual space may be even simpler or amenable to solution with a known algorithm

114

114

Envelopes

- The **lower envelope** of a set of lines is the locus of points below all lines
 - The **upper envelope** is defined similarly, it is the locus of points above all lines
- Consider a solution where the line ℓ lies above all the samples x_{i_1}, \dots, x_{i_m} from one of the two classes (all + or all -)
- In the dual plane, each dual line $D(x_{i_k}), k = 1, \dots, m$ must lie above the dual of ℓ , namely the point $D(\ell)$. In other words, $D(\ell)$ must be in the lower envelope of the dual lines

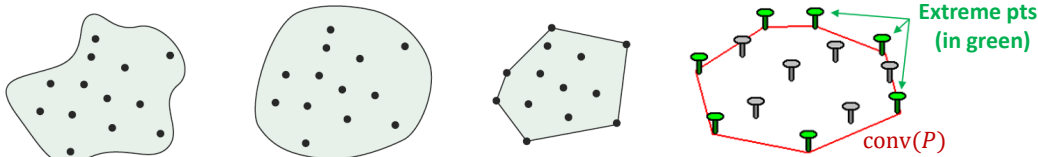
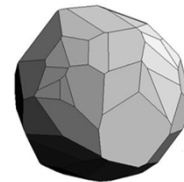


115

115

Convex Hull

- The **convex hull** of a set P of points, denoted $\text{conv}(P)$, is the smallest convex set that contains P
 - When $P \subset \mathbb{R}^d$, $\text{conv}(P)$ is a d -dimensional convex polyhedron
 - In 2D, $\text{conv}(P)$ is always a convex polygon
- The vertices of $\text{conv}(P)$ are the **extreme points** of P
 - Can you easily identify some extreme points of P ?

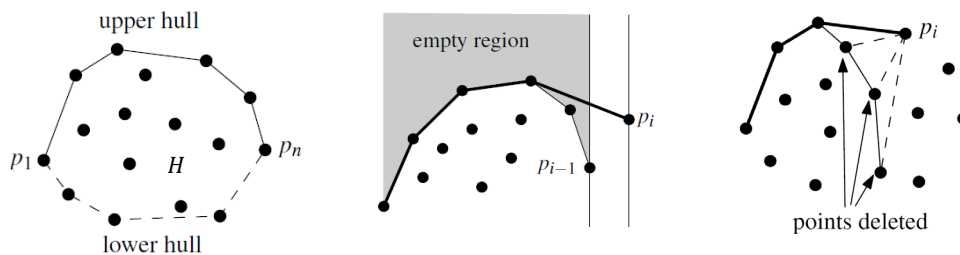


116

116

Computing envelopes

- The lower (resp. upper) envelope of a set of lines can be computed in $\mathcal{O}(n \log n)$ time, by computing the upper (resp. lower) hull of the duals of the lines (a set of points!)
- The upper hull of a set $P = \{p_1, \dots, p_n\}$ of points can be computed *incrementally*, by scanning P in ascending lexicographic order
- Basic step consists of updating the upper hull when processing point p_i
- Point p_i is always a vertex of $\text{UpperHull}(p_1, \dots, p_i)$
- *Invariant*: a CW traversal of the upper hull consists of a sequence of strict right turns
- May need to restore the invariant after inserting p_i (why and how?)



117

117

Upper Hull Algorithm

UPPERHULL(P)

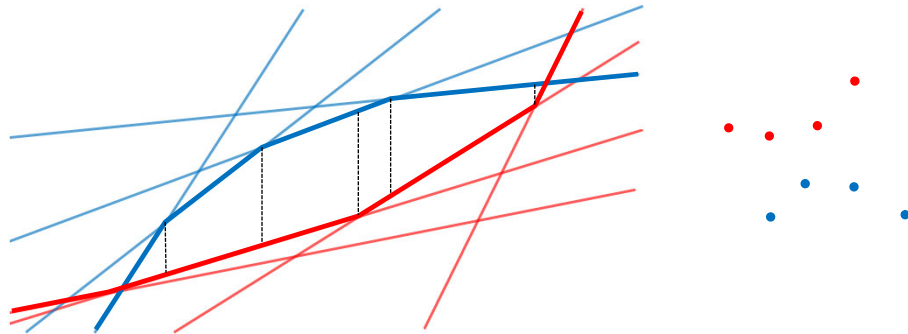
1. Sort P lexicographically by x -coordinate, yielding: p_1, \dots, p_n
2. Push p_1 and p_2 onto a stack S // stack always contains ≥ 2 points
3. **for** $i \leftarrow 3$ to n **do**
4. **while** $|S| \geq 2$ **and** $\langle S[\text{top}], S[\text{top} - 1], p_i \rangle$ do not turn right **do**
5. Pop(S)
6. Push(p_i)
7. Empty S onto H from bottom to top
8. **return** H

118

118

Using envelopes

- The optimal line ℓ^* has at least two support vectors of the same class with all points of that class on the same side of ℓ^* and the points of the other class on the other side $\Rightarrow D(\ell^*)$ is in the lower envelope of one class and the upper envelope of the other, aligned vertically with some vertex of the envelopes
- $D(\ell^*)$ can be found in the intersection of the upper envelope of one class and the lower envelope of the other



119

119

Conclusions

- We have discussed two algorithms to compute a line separating two classes: an *iterative improvement algorithm* and an *incremental algorithm* based on the duality between points and lines
- The iterative improvement algorithm requires up to R^2/γ^2 iterations to converge and fails to stop when the two classes are not separable. While in this case it produce a separating line, this line may not be optimal
- The incremental algorithm runs in $\mathcal{O}(N \log N)$ time, yields an optimal line or easily determines that no separating line exists
- Both algorithms work in any number of dimensions, but the incremental algorithm is hard to implement for higher dimensions while the iterative improvement one is the same for any d

120

120

Components of Supervised Learning: Summary

- A **domain set** \mathcal{X} , the set of objects we may wish to label, each represented by a vector of d features
- A **label set** \mathcal{Y} , the set of possible labels for objects in \mathcal{X} , e.g., \mathbb{Q}^d , $\{-1, +1\}$, etc.
- A **ground truth function** $f : \mathcal{X} \rightarrow \mathcal{Y}$ defining the correct labels for all $\mathbf{x} \in \mathcal{X}$
- A **training set** $S = \langle (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \rangle$, a finite sequence from $\mathcal{X} \times \mathcal{Y}$ which is the input to the learning algorithm
- The learner's output is a **predictor** $g : \mathcal{X} \rightarrow \mathcal{Y}$ (may use \hat{f} instead of g)
- A **hypotheses set** \mathcal{H} , the set of candidate predictors available to the learner. Thus, $g \in \mathcal{H}$
- A **data generation model**. All instances, both training and new data are generated by some (unknown) probability distribution \mathcal{D}
- A **success measure** that estimates the ability of model g to generalize

121