Lecture 1 - Jan 8

Data Matrices
Linear Models
Ordinary Least Squares

Recommended Reading

Week 1 Notes in GitHub

Data Mining and Machine Learning
1.1-1.3.1: Data Matrix, Attributes, Geometric View of Data

Elements of Statistical Learning
Ch 1: Introduction
Ch 2.1-2: Variable Types and Terminology

Upcoming Deadlines

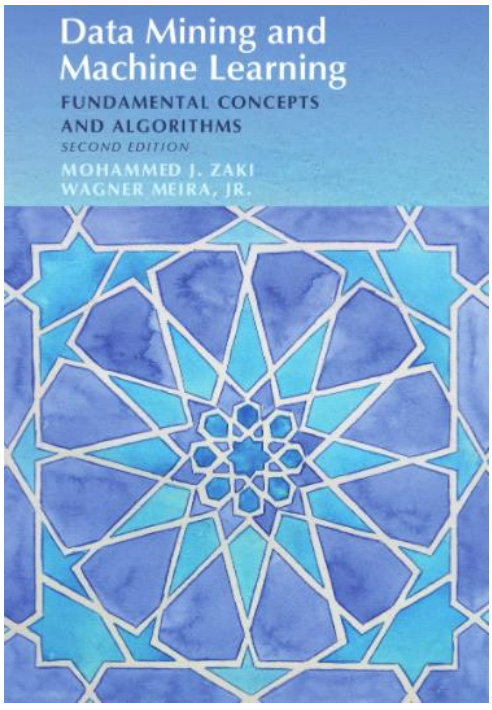
Python Assignment (Jan 20)
Homework 1 (Jan 27)

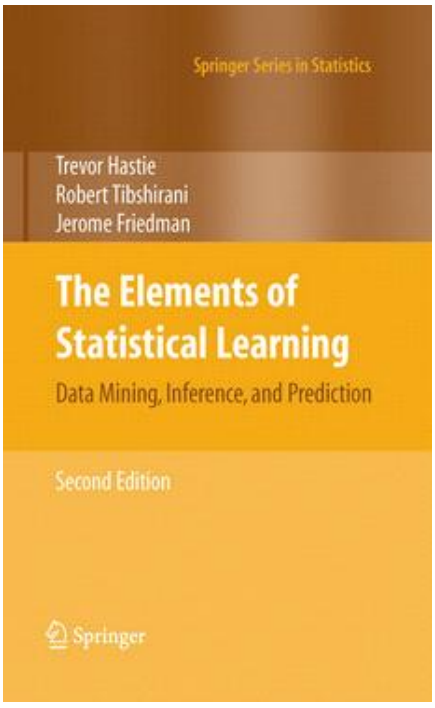
Syllabus

Thursday, January 4, 2024 2:22 PM

Books and Online Resources

Everything is freely available online!





www.dataminingbook.info/
Easy to read, pseudocode

web.stanford.edu/~hastie/ElemStatLearn/
Classic, more advanced, comprehensive

GitHub: https://github.com/rtwhite1546/Spring-2023-Intro-to-Machine-Learning
Code from class, minimal notes. Feel free to download it, use parts of my code, etc.

We will frequently refer to videos, academic papers, or websites too.

Assignments and Grading

- Homework (165 points)
  - Mathematical and practical problems
- Python Exam (35 points) -- **WEEK 2**
  - Python basics: variables, lists, if statements, loops, functions, classes, importing libraries
  - Libraries: NumPy arrays, plotting with matplotlib, pandas dataframes, scikit-learn classes
- 2 Projects (150 points each)
  - Design and work on machine learning problems
  - Choose your own topic (subject to approval)
  - Group work is permitted for ambitious projects
- 2 Exams (100 points each)
  - Ideas, theory, math, practical questions (no code)

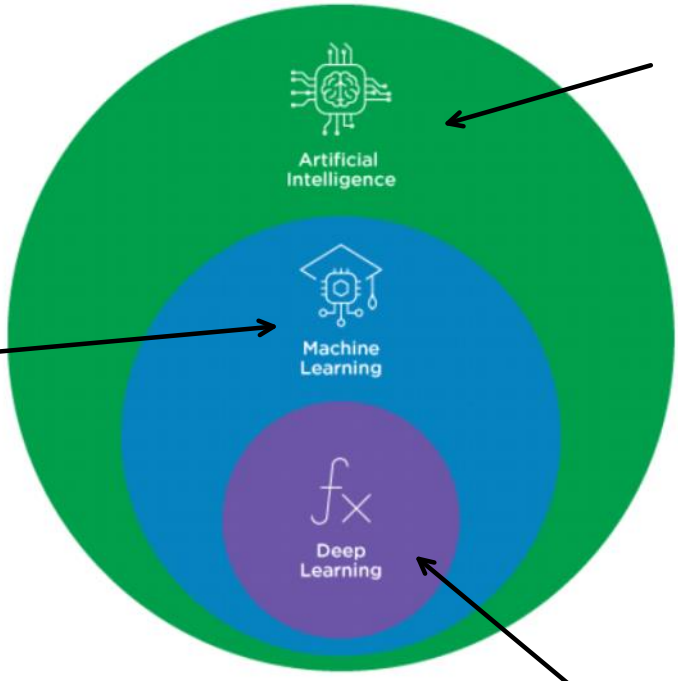
Background Assumptions for Students

- Calculus (derivatives, optimization)
- Basic linear algebra (vectors, matrices, eigenvalues/vectors)
- Proficiency in some programming language

Not assumed but helpful:

- Algorithms
- Data structures
- Deep learning
- Linear algebra
- Multivariate calculus
- Probability & Statistics
- Optimization

Machine Learning



Machines **thinking** and **learning**

Algorithms that **learn** from looking at data

A class of many-layered (**deep**) neural networks that **learn** from data

Given:

- A task T
- A performance measure P
- An experience E

A computer program **learns** from experience E with respect to some task T and performance measure P **if it improves at task T, measured by P, with experience E.**

Machine learning **tasks** are generally too difficult for pre-designed programs to do. Instead, we write programs for computers to **learn** to solve them.

- For example, we might want a machine learning algorithm to determine if a picture has a cat in it.

has a cat in it.

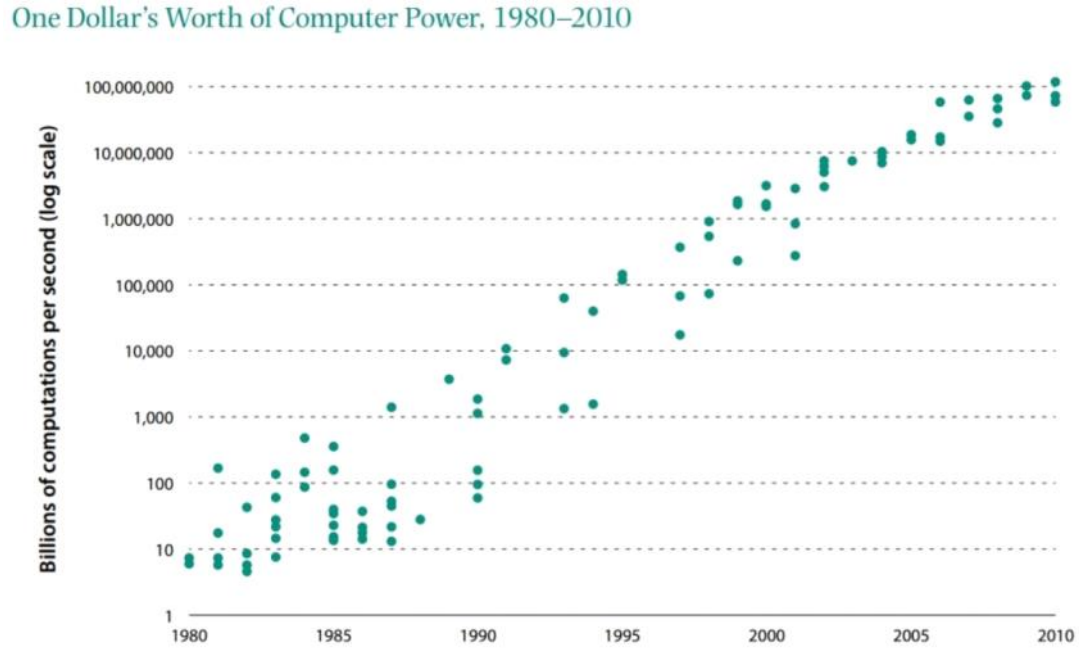
- We **WOULD NOT** give a computer set of instructions for how to decide what cats look like. (What would the instructions be?!)
- We **WOULD** feed many cat pictures to the computer, feed many non-cat pictures to the computer, and let it adjust and adjust until it learns to correctly label them

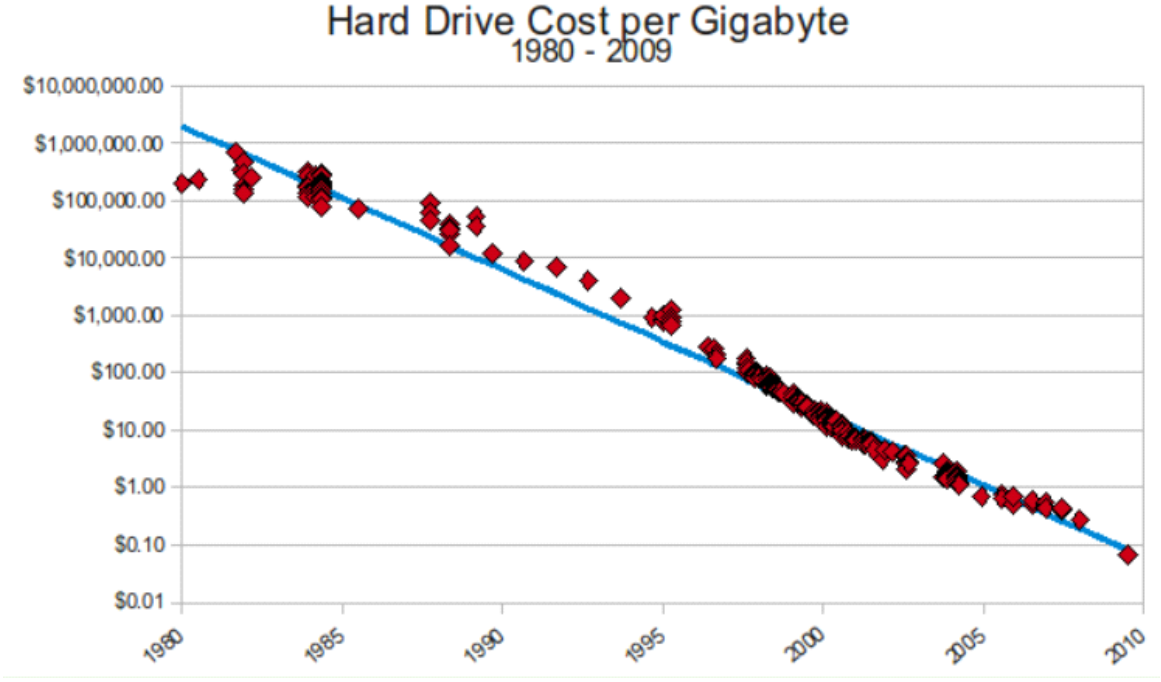
There are many types of tasks. Two large classes of tasks are supervised and unsupervised learning tasks.

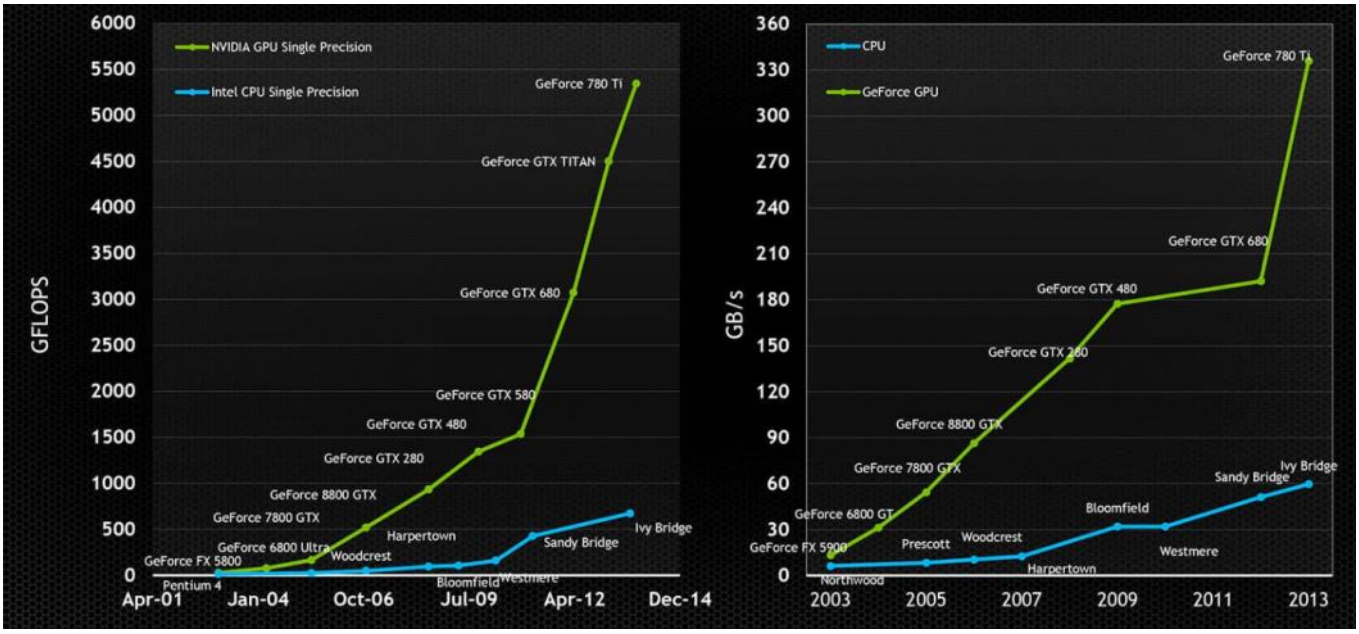
- **Supervised learning algorithms** have an experience of observing a dataset of examples *with* labels a correct algorithm would output.
- **Unsupervised learning algorithms** have an experience of observing a dataset *without* labels and seek to learn useful patterns in the dataset.

It should be noted that machine learning practitioners commonly use multiple methods, build methods customized to their problems, and create pipelines using multiple methods in a specified sequence.

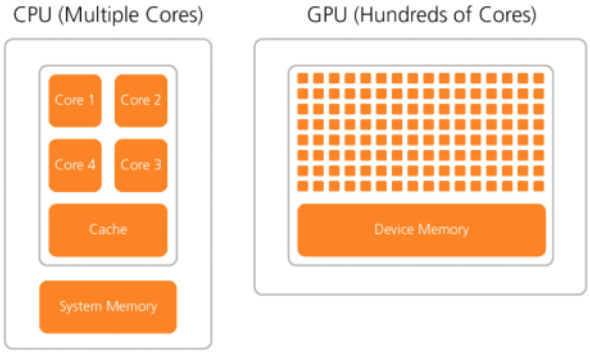
Why has ML grown in popularity?







NVIDIA CUDA



TensorFlow: Expressing High-Level ML Computations

- Core in C++
  - Very low overhead
- Different front ends for specifying/driving the computation
  - Python and C++ today, easy to add more

Automatically Runs on Variety of Platforms

phones

single machines (CPU and/or GPUs) …

distributed systems of 100s of machines and/or GPU cards

custom ML hardware

Fathom

TensorFlow

Developer(s)	Google Brain Team[1]
Initial release	November 9, 2015; 4 years ago
Stable release	2.2.0[2] / May 6, 2020; 2 months ago
Repository	github.com/tensorflow/tensorflow
Written in	Python, C++, CUDA

NumPy

Original author(s)	Travis Oliphant
Developer(s)	Community project
Initial release	As Numeric, 1995; as NumPy, 2006
Stable release	1.19.1 / 21 July 2020; 5 days ago[1]
Repository	github.com/numpy/numpy
Written in	Python, C

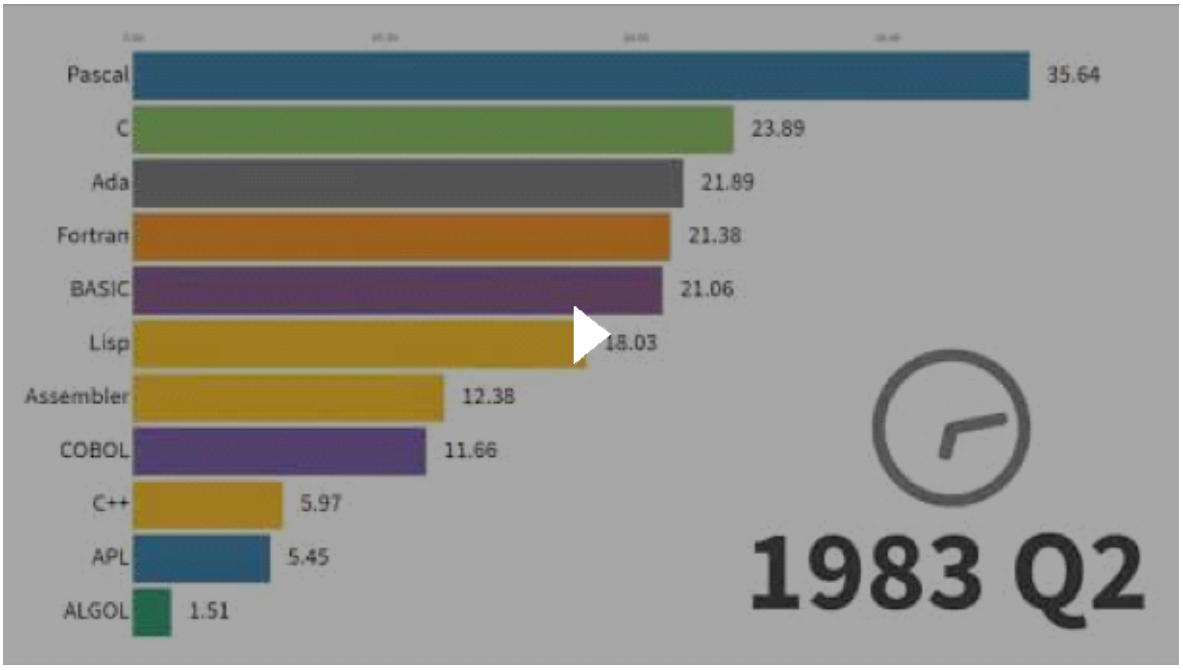
python™

Paradigm	Multi-paradigm: functional, imperative, object-oriented, structured, reflective
Designed by	Guido van Rossum
Developer	Python Software Foundation
First appeared	1990; 30 years ago[1]
Stable release	3.8.5 / 20 July 2020; 7 days ago[2]

scikit learn

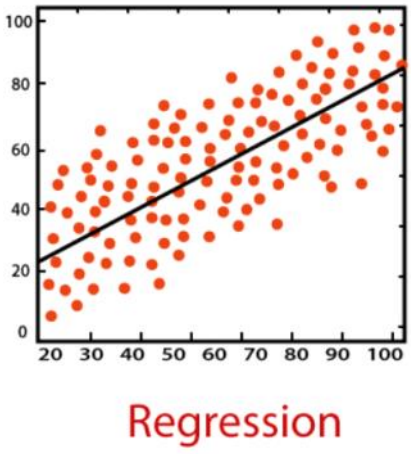
Original author(s)	David Cournapeau
Initial release	June 2007; 13 years ago
Stable release	0.23 / 12 May 2020; 2 months ago[1][2]
Repository	github.com/scikit-learn/scikit-learn
Written in	Python, Cython, C and C++

[Most Popular Programming Languages 1965 - 2020]


Pascal
35.64
C
23.89
Ada
21.89
Fortran
21.38
BASIC
21.06
Lisp
18.03
Assembler
12.38
COBOL
11.66
C++
5.97
APL
5.45
ALGOL
1.51
1983 Q2

Plan for the Course

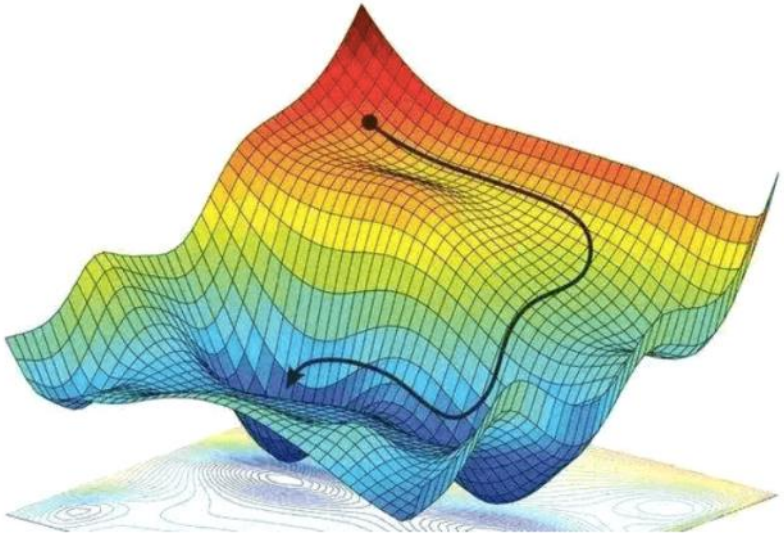
Weeks 1-4: Regression



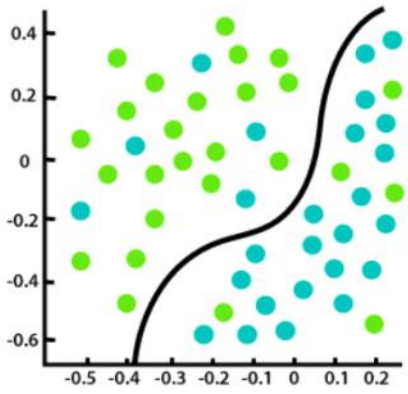
*Given a (high-dimensional) input point, predict **numerical** outputs*

Math Required	**Regression Methods**	**Practical ML Ideas**
Linear Algebra Numerical Optimization	Linear Regression (LBF, Ridge, and LASSO) Local Regression (kernel methods, RBF) Structured Regression (additive models)	Supervised learning Data Representation Loss Functions Train/Test Splits

Numerical Optimization



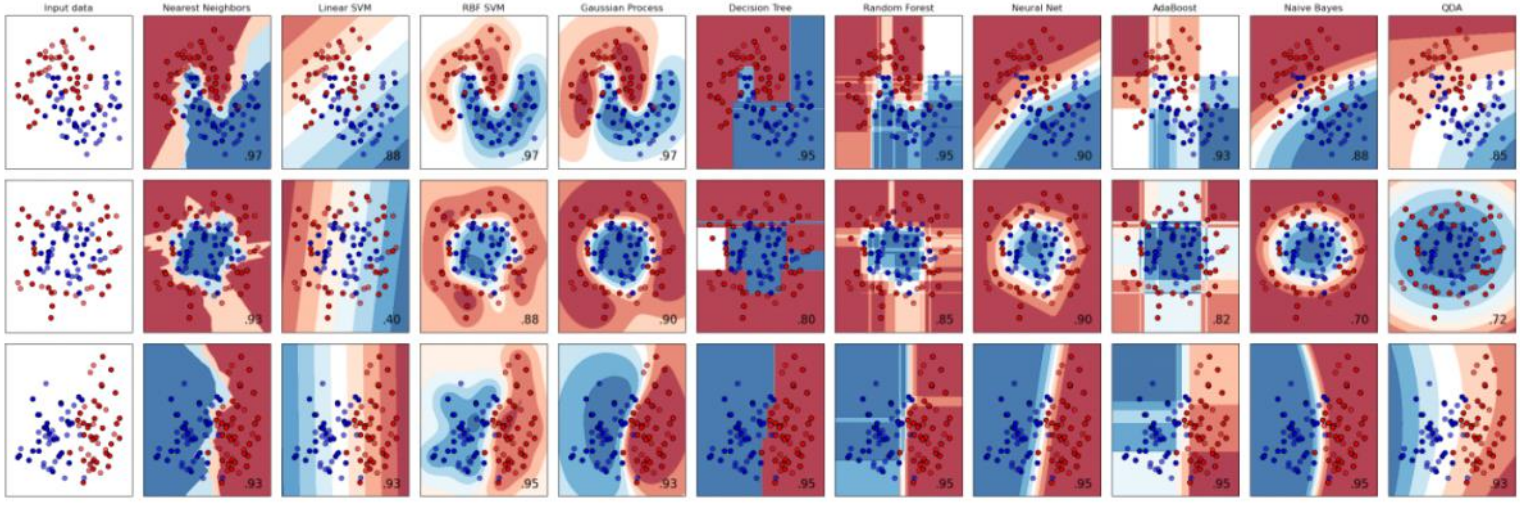
Weeks 5-10: Classification



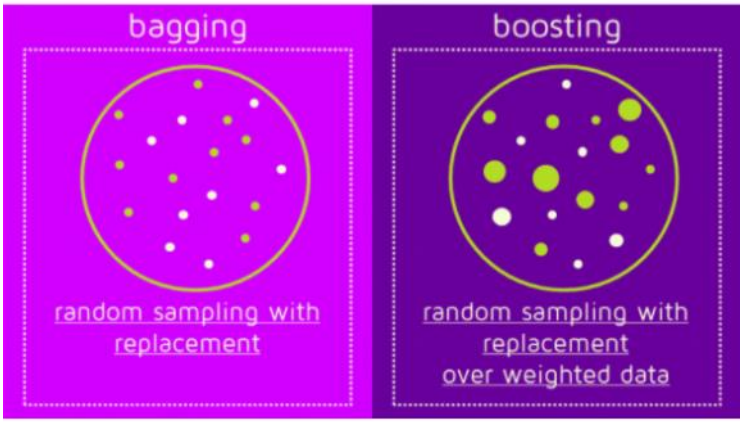
Classification

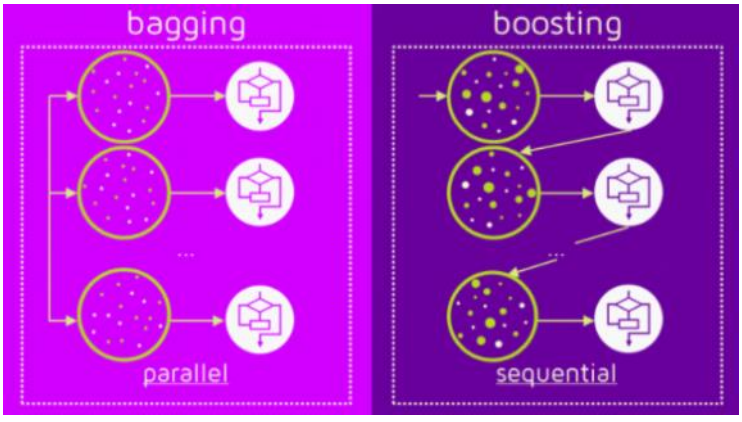
*Given a (high-dimensional) input point, predict **categorical** outputs*

Math Required	**Classification Methods**	**Practical ML Ideas**
Probability and statistics Maximum likelihood Gaussian distribution	Probabilistic classification (Bayes classifier, logistic regression) Linear/quadratic classifiers (LDA/QDA, SVMs) Kernel methods (k-nearest neighbors) Support vector machines Decision trees	Cross-validation Interpretable ML



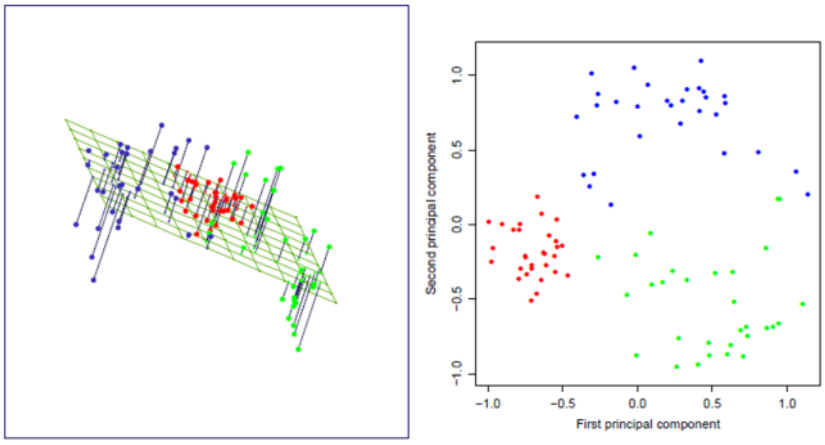
Weeks 11-12: Ensemble methods for classification and regression



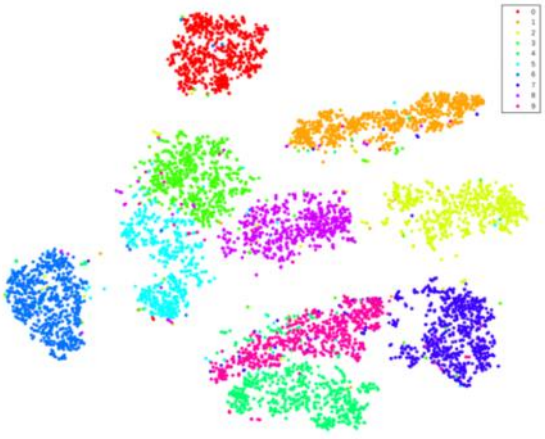


Math Required	**Methods**	**Practical ML Ideas**
Expected value and variance Random sampling	Bagging -- random forests Boosting -- AdaBoost, XGBoost	Bias-variance decomposition Ensemble learning Feature importance

Weeks 13-14: Dimensionality Reduction



Principal Components Analysis

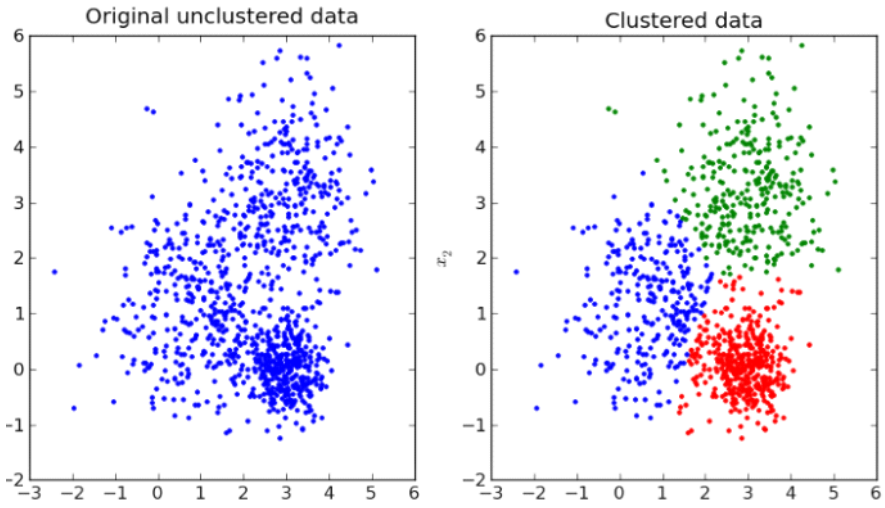


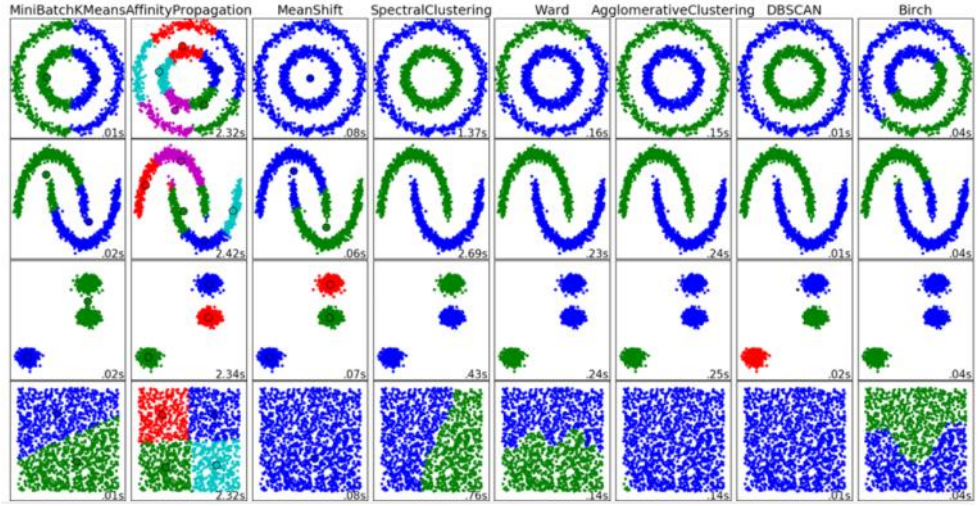
t-Stochastic Neighbors Embedding

Math Required	**Methods**	**Practical ML Ideas**
Eigenvectors and linear subspaces Orthogonal bases and projection *t*-distributions	PCA and variations MDS t-SNE	Unsupervised learning Dimensionality reduction Data viz

15-16

Weeks ~~12-14~~4: Clustering





Math Required	**Methods**	**Practical ML Ideas**
Local topology	K-means and Gaussian mixtures	Unsupervised learning
Maximum likelihood	Hierarchical clustering	Clustering
	Density-based clustering	Expectation maximization

Matrices

$A \in \mathbb{R}^{n \times d}$ is a **matrix**

row vectors $\in \mathbb{R}^d$; column vectors $\in \mathbb{R}^n$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1d} \\ a_{21} & a_{22} & \cdots & a_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nd} \end{bmatrix} \begin{matrix} \leftarrow \text{row } 1 \\ \leftarrow \text{row } 2 \\ \vdots \\ \leftarrow \text{row } n \end{matrix} = \begin{bmatrix} — & a_1 & — \\ — & a_2 & — \\ & \vdots & \\ — & a_n & — \end{bmatrix} = \begin{bmatrix} | & | & & | \\ A_1 & A_2 & \cdots & A_d \\ | & | & & | \end{bmatrix}$$

$\uparrow$ col 1 $\uparrow$ col 2 $\uparrow$ col d

$I_n = \begin{bmatrix} 1 & & 0 \\ & 1 & \\ & \ddots & \\ 0 & & 1 \end{bmatrix}_{n \times n}$ is the **identity matrix**

$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1d} \\ & u_{22} & \cdots & u_{2d} \\ & & \ddots & \\ 0 & & & u_{nd} \end{bmatrix}$ + $L = \begin{bmatrix} l_{11} & & & 0 \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nd} \end{bmatrix}$ are **upper triangular** and **lower triangular** matrices, respectively

$D = \begin{bmatrix} d_{11} & & & 0 \\ & d_{22} & & \\ & & \ddots & \\ 0 & & & d_{nn} \end{bmatrix}$ is a **diagonal matrix**

Data Matrix

In this section, we will cover how data is frequently stored such that it can be used by machine learning methods. It should not be thought that this is the only way or always the best way to store data, but it will be a series of conventions used in many fields.

For many problems, we will have a **dataset** consisting of some number n points in $\mathbb{R}^d$ that we will store in a matrix

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} \end{pmatrix}$$

The ith *row* $x_i = (x_{i1}, x_{i2}, ..., x_{id}) \in \mathbb{R}^d$ is the ith point in the dataset. These points have many names in different fields: **points, datapoints, examples, vectors, records, feature-vectors**. In some sources, these points x_i are denoted $\mathbf{x}_i$, but it should be clear that x with a single subscripts indicates a point while x with two subscripts is the component of a point.

The jth *column* $X_j = (x_{1j}, x_{2j}, ..., x_{nj}) \in \mathbb{R}^n$ is the jth component of each point in the dataset. These are likewise called by many names dependent on field: **features, attributes, variables, dimensions, properties, fields**. In some cases, each column can be considered a random sample of a random variable, or the rows of the matrix X can be considered a random sample of vector-valued random variables.

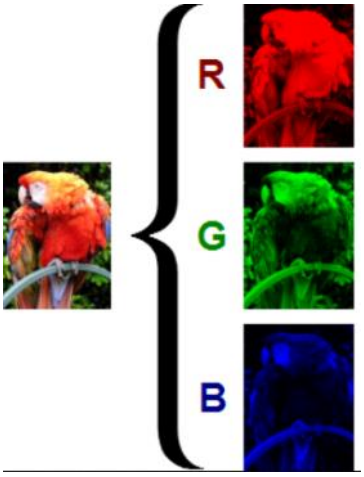
The number of points n is the **size** of the dataset and the length of the points d is called the **dimensionality** of the dataset.

Examples

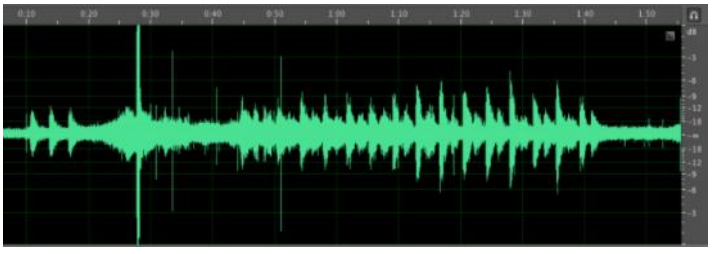
- If we have a dataset of medical records where each datapoint has a single patient's age, weight, blood pressure, status as a smoker or not, and other information and each patient is known to either have or not have kidney disease. We might want take data from a new patient and predict if he or she is likely to develop kidney disease.
- If we have a dataset of labeled images of cat and dogs, we might want to take a new image and classify whether it has a dog or a cat. (How does Google image search know how to find pictures of what you search?)
- If we have a dataset of audio files, each of which is a jazz, classical, rock, pop, or hip-hop song, we might want to predict the genre of a new audio file. (Spotify does this kind of analysis to recommend songs based on your listening history.)
- If we have a dataset of traffic logs on a network, some known to be infected by a specific virus and some are not, we might want to use this information to classify a new traffic log as likely to be infected or not.
- If we have a dataset of sounds of people speaking along with transcripts of the words, we might want to classify the words spoken into a microphone. (Think Siri!)
- The medical records would have numbers for age, weight, and blood pressure and a binary digit for non-smoker or smoker.

	Gender (M/F)	Age	Weight (lbs.)	Height (in.)	Smoking (0=No, 1=Yes)	Race
Patient #1	M	59	175	69	0	White
Patient #2	F	67	140	62	1	Black
Patient #3	F	73	155	59	0	Asian
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
Patient #75	M	48	190	72	0	White

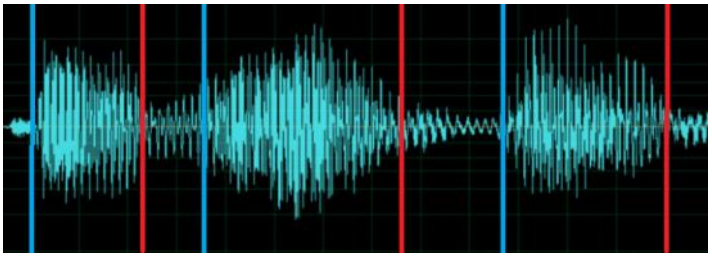
- The dog/cat images might have three channels for a picture, meaning three numbers for each pixel (the red, green, and blue levels) like the bird picture below.



- The audio files might have numbers specifying the type of sound for the song many times per second.



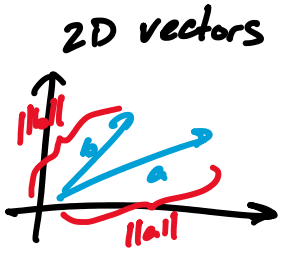
- The network traffic logs might have numbers of packets transferred, file size, ports, addresses, the content of the packets, etc.
- The audio files might have numbers specifying the type of sound for a word many times per second.



In all mature applications, there are likely preprocessing steps done before the classification is done.

I chose these applications to demonstrate two things: (1) classification problems are interesting and useful in almost every area of study and (2) a huge class of classification problems have much in common mathematically. All apply to datapoints, although some types of data may have far more dimensions than others--a medical record may only have 10 to 12 numbers, but a 12-megapixel photo from the latest iPhone would have $4,000 \times 3,000 \times 3 = 36,000,000$ numbers, 3 for each pixel).

Let $a, b \in \mathbb{R}^d$, $a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_d \end{bmatrix}$, $b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_d \end{bmatrix}$



$a \cdot b = a^T b = a_1 b_1 + a_2 b_2 + \cdots + a_d b_d = \sum_{i=1}^{d} a_i b_i$

is the dot product of a and b

$\|a\| = \sqrt{a^T a} = \sqrt{a_1^2 + a_2^2 + \cdots + a_d^2} = \left(\sum_{i=1}^{d} |a_i|^2 \right)^{1/2}$ is the Euclidean length of a (or the L^2 norm of a)

Note: $\|a\|^2 = a^T a$ ← "sum of squares" $a_1^2 + a_2^2 + \cdots + a_d^2$

$\|a\|_p = \left(\sum_{i=1}^{d} |a_i|^p \right)^{1/p}$ is the L^p norm of a (for any $p \neq 0$)

The Euclidean distance between a and b is $\|a - b\|$

Matrix Multiplication

it is not elementwise multiplication

must be the same to multiply matrices

Let $A \in \mathbb{R}^{n \times d}$, $B \in \mathbb{R}^{d \times m}$

$$AB = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1d} \\ a_{21} & a_{22} & \cdots & a_{2d} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nd} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & & \vdots \\ b_{d1} & b_{d2} & \cdots & b_{dm} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{d} a_{1i}b_{11} & \sum_{i=1}^{d} a_{1i}b_{12} & \cdots & \sum_{i=1}^{d} a_{1i}b_{1m} \\ \sum_{i} a_{2i}b_{11} & \sum_{i} a_{2i}b_{12} & \cdots & \sum_{i} a_{2i}b_{1m} \\ \vdots & \vdots & & \vdots \\ \sum_{i} a_{ni}b_{11} & \sum_{i} a_{ni}b_{12} & \cdots & \sum_{i} a_{ni}b_{1m} \end{bmatrix}_{n \times m}$$

(a_2 picks out row 2 of A; B_1 picks out column 1 of B)

$$= \begin{bmatrix} a_1 \cdot B_1 & a_1 \cdot B_2 & \cdots & a_1 \cdot B_m \\ a_2 \cdot B_1 & a_2 \cdot B_2 & \cdots & a_2 \cdot B_m \\ \vdots & \vdots & & \vdots \\ a_n \cdot B_1 & a_n \cdot B_2 & \cdots & a_n \cdot B_m \end{bmatrix}$$

properties: $AB \neq BA$ in general ← matrix multiplication is not commutative

distributive properties: $A(B+C) = AB + AC$ & $(A+B)C = AC + BC$

$c(AB) = (cA)B = A(cB) = (AB)c$ ← constant

associative property: $(AB)C = A(BC)$

identity matrix multiplication: $AI = A$, $IA = A$

do all of these properties assume the matrices are shaped such that these operations are defined

Computational Notes:

As of 2020, the fastest algorithm is $O(n^{2.3728596})$ for $n \times n$ matrices

Special types of matrices can be faster (e.g. symmetric, banded, triangular, sparse, diagonal matrices) — mostly 0's

We can safely rely on tools like numPy, tensorflow, CUDA, MATLAB, etc. to take care of this efficiently

Suppose we have a dataset with data points (inputs)

$$x_1, x_2, \ldots, x_n \in \mathbb{R}^{d+1}, \quad x_i = \begin{bmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{id} \end{bmatrix}$$

with numerical targets

$$y_1, y_2, \ldots, y_n \in \mathbb{R}$$

Linear regression tries map the inputs x_i to their targets y_i for $i=1,\ldots,n$ with a function linear in some learnable parameters $\theta_0, \theta_1, \ldots, \theta_d \in \mathbb{R}$ as

$$\hat{f}(x_i) = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \cdots + \theta_d x_{id}$$

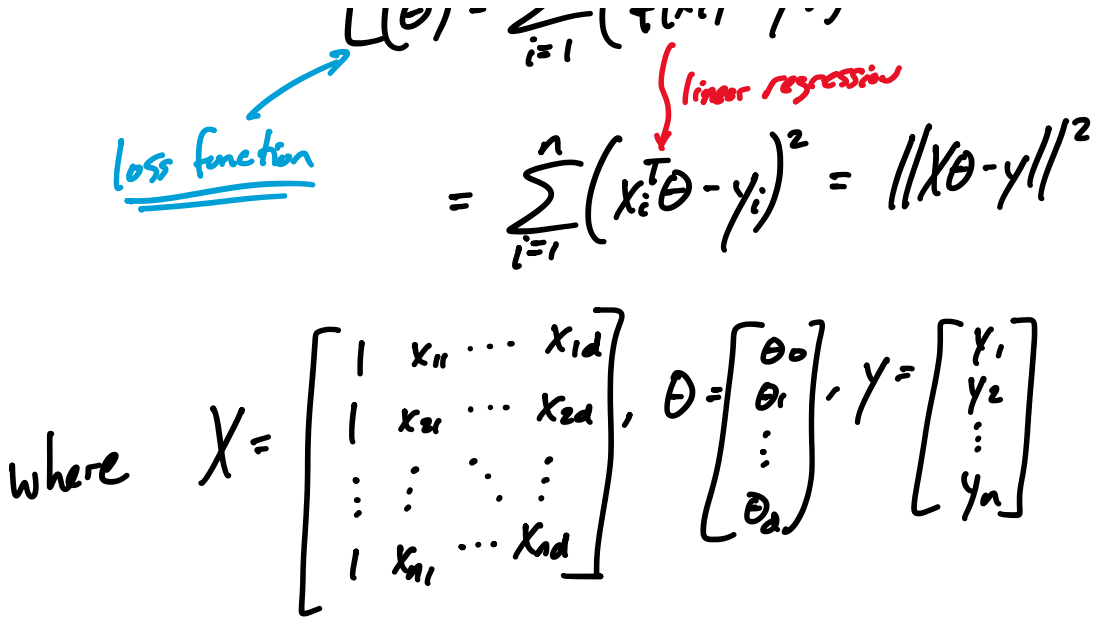
$$\hat{f}(x_i) = \begin{bmatrix} 1 & x_{i1} & \cdots & x_{id} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} = x_i^T \theta$$

But we need some criteria for choosing numerical values for θ that will predict the targets effectively

Ordinary least squares aims to minimize the sum of squared errors in prediction of the targets in the training dataset.

$$L(\theta) = \sum_{i=1}^{n} \left(\hat{f}(x_i) - y_i \right)^2 \quad \text{- OLS}$$

(linear regression)



Goal: choose or "learn" a vector θ that minimizes the loss $L(\theta)$

Recall from Calc 1: Optimization

Let $f:[a,b]\to\mathbb{R}$ be a differentiable function

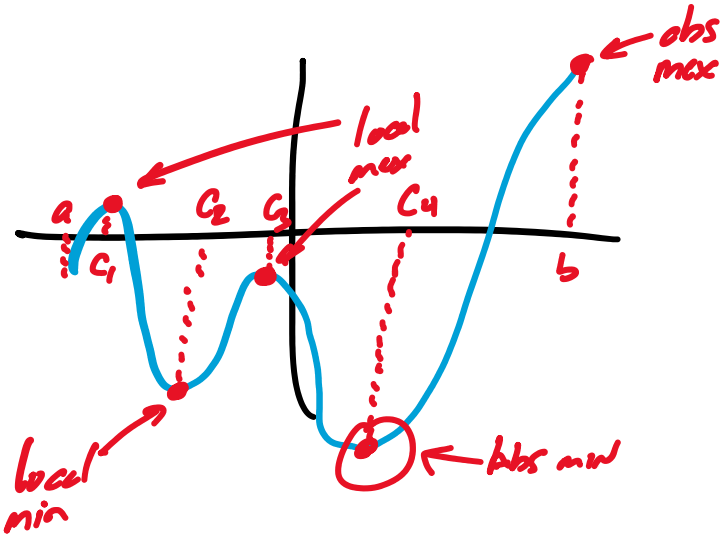
From differential calculus, we know there exists an absolute minimum located at $x=a$, $x=b$, or at some $x=c\in(a,b)$ s.t. $f'(c)=0$.

at a critical value

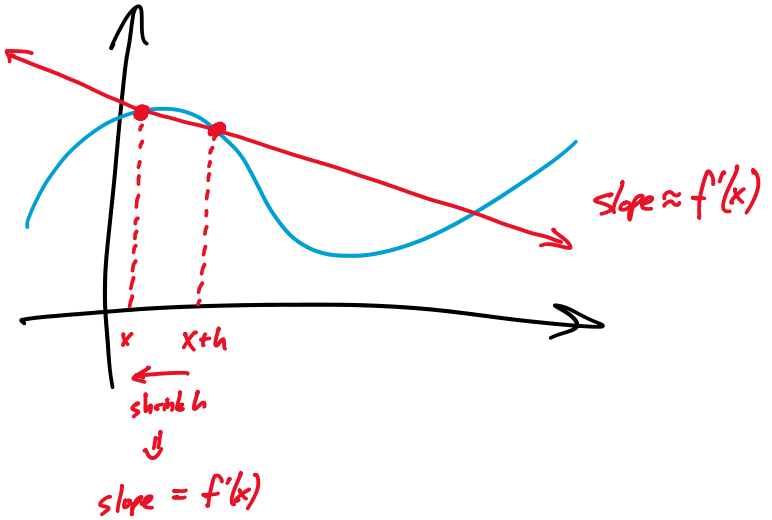
to find it... ① Find all critical values in (a,b)

② Compare f at $x=a$, $x=b$, and at each critical value

↳ lowest output = abs min



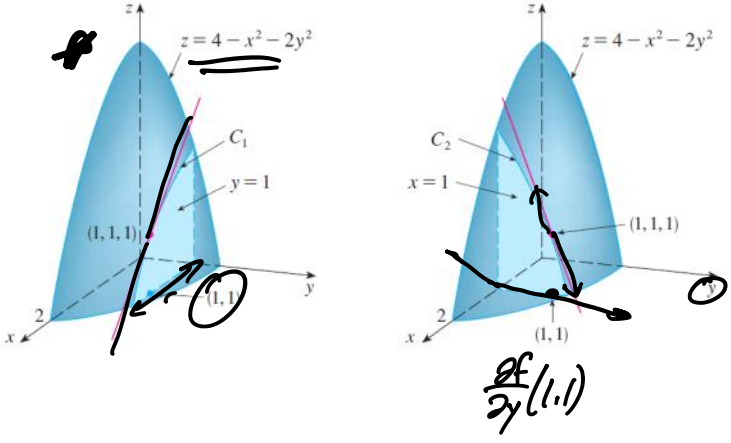
If $f: \mathbb{R} \to \mathbb{R}$, $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$



If $L: \mathbb{R}^{d+1} \to \mathbb{R}$, the partial derivative of L with respect to θ_i is

$$L_{\theta_i}(\theta) = \frac{\partial L}{\partial \theta_i} = \lim_{h \to 0} \frac{L(\theta_0, \ldots, \theta_{i-1}, \theta_i + h, \theta_{i+1}, \ldots, \theta_d) - L(\theta)}{h}$$

Example: Let $f(x,y) = 4 - x^2 - 2y^2$

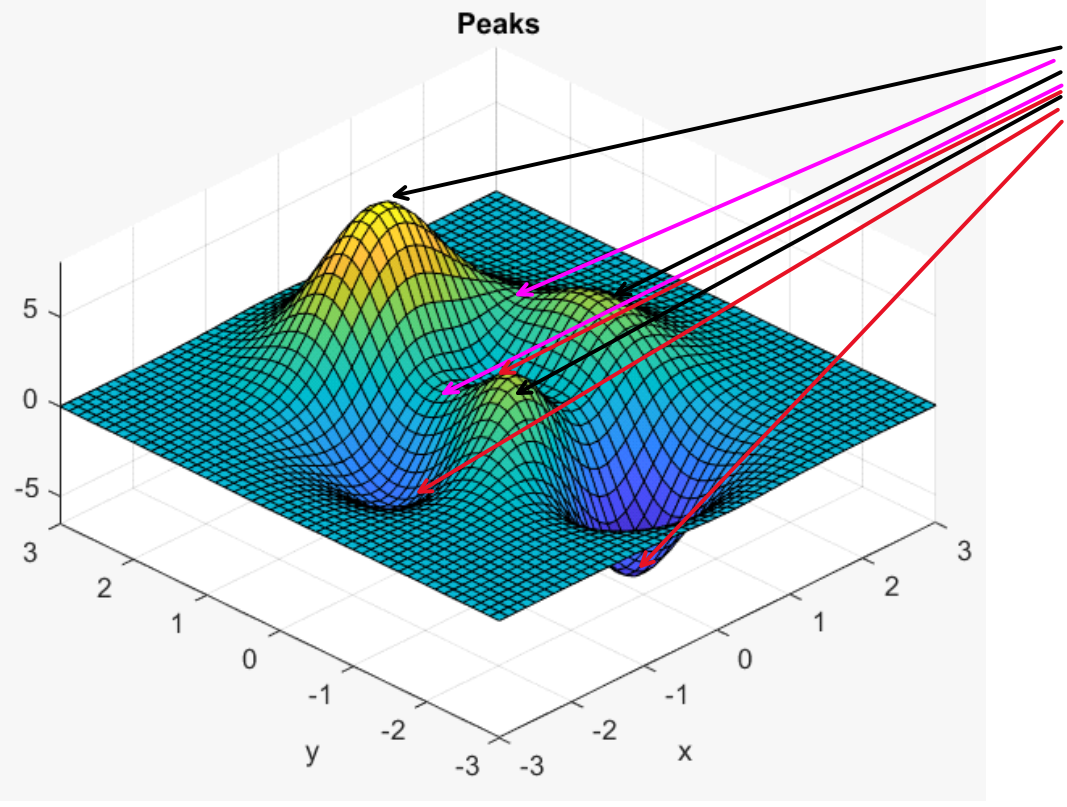


$\frac{\partial f}{\partial y}(1,1)$

The gradient of L is $\nabla L(\theta) = \begin{bmatrix} \frac{\partial L}{\partial \theta_0}(\theta) \\ \vdots \\ \frac{\partial L}{\partial \theta_d}(\theta) \end{bmatrix}$

In ML, we would like to minimize L.

Multivariate Calculus: Find θ such that $\nabla L(\theta) = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$ (crit. points)



Critical values (where $\nabla L(\theta) = 0$) are "flat spots" on the graph.

Red = local **minima**

Black = local **maxima**

Magenta = **saddle points**

Matrix Transpose

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1d} \\ a_{21} & a_{22} & \cdots & a_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nd} \end{bmatrix} = \begin{bmatrix} - & a_1 & - \\ - & a_2 & - \\ & \vdots & \\ - & a_n & - \end{bmatrix}$$

The transpose of A is

Swap rows and columns

$$A^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1d} & a_{2d} & \cdots & a_{nd} \end{bmatrix} = \begin{bmatrix} | & | & & | \\ a_1^T & a_2^T & \cdots & a_n^T \\ | & | & & | \end{bmatrix}$$

properties $(AB)^T = B^T A^T$

If $A^T = A$, then A is a symmetric matrix

Thursday, January 4, 2024 4:30 PM

OLS problem $\min_{\theta \in \mathbb{R}^{d+1}} L(\theta)$

To solve, find critical value where $\nabla L(\theta) = 0$, if possible.

$$L(\theta) = \|X\theta - y\|^2 = (X\theta - y)^T (X\theta - y)$$
$$= ((X\theta)^T - y^T)(X\theta - y)$$
$$= (X\theta)^T X\theta - (X\theta)^T y - y^T X\theta + y^T y$$
$$= \theta^T X^T X\theta - \theta^T X^T y - y^T X\theta + y^T y$$

Gradient of $\theta^T X^T X\theta$

$$\theta^T X^T X\theta = \theta^T \underbrace{\begin{bmatrix} a_{00} & \cdots & a_{0d} \\ \vdots & \ddots & \vdots \\ a_{d0} & \cdots & a_{dd} \end{bmatrix}}_{(d+1)\times(d+1)} \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_d \end{bmatrix} = [\theta_0 \cdots \theta_d] \begin{bmatrix} a_0 \cdot \theta \\ \vdots \\ a_d \cdot \theta \end{bmatrix} = \theta_0 a_0 \cdot \theta + \cdots + \theta_d a_d \cdot \theta$$

$$= \theta_0 \sum_{j=0}^{d} a_{0j}\theta_j + \cdots + \theta_d \sum_{j=0}^{d} a_{dj}\theta_j$$

$$= \sum_{i=0}^{d} \sum_{j=0}^{d} a_{ij}\theta_i\theta_j$$

$$\begin{bmatrix} 2a_{00}\theta_0 + \sum_{j\neq 0} a_{0j}\theta_j + \sum_{i\neq 0} a_{i0}\theta_i \\ \vdots \end{bmatrix} = \begin{bmatrix} a_0 \cdot \theta + A_0 \cdot \theta \\ \vdots \end{bmatrix}$$

$$\nabla(\theta^T X^T X\theta) = \begin{bmatrix} 2a_{00}\theta_0 + \sum_{j\neq 0} a_{0j}\theta_j + \sum_{i\neq 0} a_{i0}\theta_i \\ \vdots \\ 2a_{dd}\theta_d + \sum_{j\neq d} a_{dj}\theta_j + \sum_{i\neq d} a_{id}\theta_i \end{bmatrix} = \begin{bmatrix} a_0\theta + A_0\cdot\theta \\ \vdots \\ a_d\theta + A_d\cdot\theta \end{bmatrix}$$

$$= \begin{bmatrix} a_0\cdot\theta \\ \vdots \\ a_d\cdot\theta \end{bmatrix} + \begin{bmatrix} A_0\cdot\theta \\ \vdots \\ A_d\cdot\theta \end{bmatrix}$$

$$= X^T X\theta + (X^T X)^T\theta$$

Note $(X^T X)^T = X^T(X^T)^T = X^T X \Rightarrow X^T X$ is symmetric

$$\nabla(\theta^T X^T X\theta) = 2X^T X\theta$$

Middle terms

$$y^T X\theta = y\cdot(X\theta) = (X\theta)\cdot y = (X\theta)^T y = \theta^T X^T y$$

$$\Rightarrow \quad \theta^T X^T y + y^T X\theta = \theta^T X^T y + \theta^T X^T y = 2\theta^T X^T y = 2\theta\cdot(X^T y)$$

$$\Rightarrow \nabla(\theta^T X^T y + y^T X\theta) = \nabla(2X^T y\cdot\theta) = 2X^T y$$

Last term

$\nabla(\ldots) = 0$

Last term

$y^T y$ is independent of θ, so $\nabla(y^T y) = 0$

Altogether,

$$\nabla L(\theta) = 2X^T X\theta - 2X^T y = 0$$