Lecture 1
Machine Learning Algorithms

Math Essentials for Machine Learning

Why worry about the math?

- There are lots of easy-to-use machine learning packages out there
- After this course, you will know how to apply several of the most general-purpose algorithms

HOWEVER

- To get really useful results, you need good mathematical intuitions about certain general machine learning principles, as well as the inner workings of the individual algorithms.

Why worry about the math?

- These intuitions will allow you to:
  - Choose the right algorithm(s) for the problem
  - Make good choices on parameter settings, validation strategies
  - Recognize over- or under fitting
  - Troubleshoot poor / ambiguous results
  - Put appropriate bounds of confidence / uncertainty on results
  - Do a better job of coding algorithms or incorporating them into more complex analysis pipelines

Notation

- $a \in A$ — *set membership*: *a* is *member* of set *A*
- $|B|$ — *cardinality*: number of ˙tems ˙n set *B*
- $\|\mathbf{v}\|$ — *norm*: length of vector *v*
- $\sum$ — *summation*
- $\int$ — *integral*
- $\Re$ — the set of *real* numbers
- $\Re^n$ — *real number space* of dimension *n*

  n = 2 : plane or 2-space
  n = 3 : 3- (dimensional) space
  n > 3 : *n*-space or *hyperspace*

Notation

- **x**, **y**, **z**, **u**, **v** — *vector* (bold, lower case)
- **A, B, X** — *matrix* (bold, upper case)
- $y = f(x)$ — *function* (*map*): assigns unique value in range of *y* to each value in domain of *x*
- dy / dx — *derivative* of *y* with respect to single variable *x*
- $y = f(\mathbf{x})$ — *function* on multiple variables, i.e. a vector of variables; *function* in *n*-space
- $\partial y / \partial x_i$ — *partial derivative* of y with respect to element *i* of vector **x**

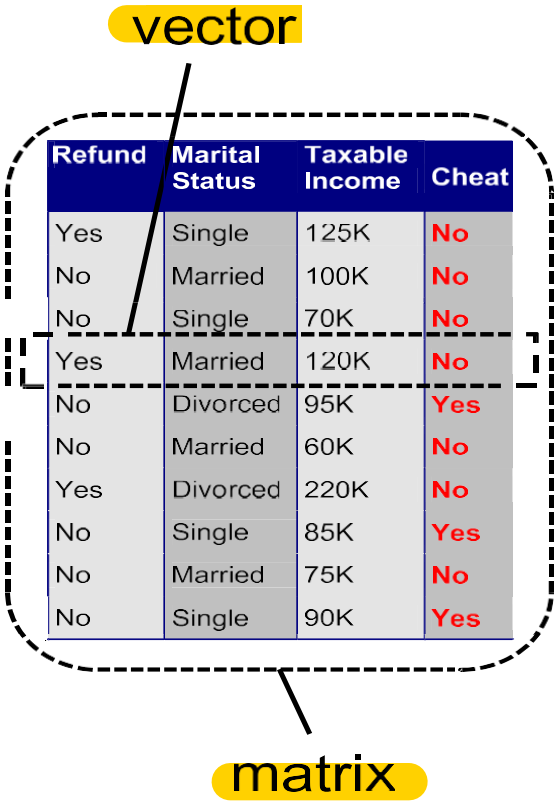
Linear Algebra

- Linear algebra applications:
  1) Operations on or between vectors and matrices
  2) Coordinate transformations
  3) Dimensionality reduction
  4) Linear regression
  5) Solution of linear systems of equations
  6) others

- Applications 1) – 4) are directly relevant to this course.

Why vectors and matrices?

- Most common form of data organization for machine learning is a 2D array, where
- – *rows* represent samples (records, items, datapoints)
- – *columns* represent attributes (features, variables)
- Natural to think of each sample as a *vector* of attributes, and whole array as a *matrix*



Refund	Marital Status	Taxable Income	Cheat
Yes	Single	125K	No
No	Married	100K	No
No	Single	70K	No
Yes	Married	120K	No
No	Divorced	95K	Yes
No	Married	60K	No
Yes	Divorced	220K	No
No	Single	85K	Yes
No	Married	75K	No
No	Single	90K	Yes

Vectors

- Definition: an *n*-tuple of values (usually real numbers)
  - *n* referred to as the dimension of the vector
  - *n* can be any positive integer, from *1* to infinity
- Can be written in column form or row form
  - Column form is conventional
  - Vector elements referenced by subscript

$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \quad , \quad \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix}$$

- Can think of a vector as a point in space *or*
- A directed line segment with a magnitude and direction

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad \vec{x}^T = (x_1, \ldots, x_n)$$

T. transpose

Vector Arithmetic

- Addition of two vectors

$$\vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad \vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \Rightarrow \begin{pmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{pmatrix}$$

- Scalar multiplication of a vector

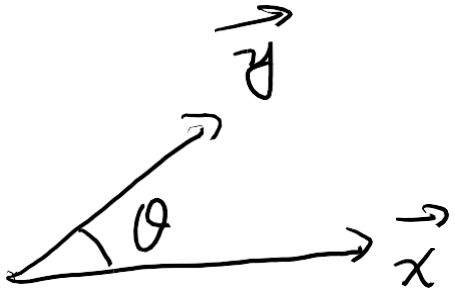
$$\vec{y} = a\vec{x} = \begin{pmatrix} ax_1 \\ \vdots \\ ax_n \end{pmatrix}$$

Vector Arithmetic

- Dot product of two vectors

$$a = \vec{x} \cdot \vec{y} = \sum_{i=1}^{n} x_i y_i$$

↳ scalar



- Dot product alternative form

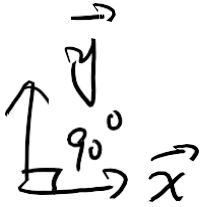
$$a = \vec{x} \cdot \vec{y} = \|\vec{x}\| \|\vec{y}\| \cos\theta$$

if $\theta = 0$

$a = \|\vec{x}\| \|\vec{y}\|$

$\theta = 90^\circ$

$a \quad 0$



Matrices

- Definition: an $m \times n$ two-dimensional array of values (usually real numbers)
  - m rows
  - n columns
- Matrix referenced by two-element subscript

$$\vec{A} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

Matrices

- A vector can be regarded as a special case of a matrix, where one of matrix dimensions = 1
- Matrix transpose (denoted ^{T})
  - Swap columns and rows
    - row 1 becomes column 1, etc.
  - m × n matrix becomes n × m matrix

$$\vec{A} = \begin{pmatrix} 2 & 7 & -1 & 0 & 3 \\ 4 & 6 & -3 & 1 & 8 \end{pmatrix} \rightarrow A^T = \begin{pmatrix} 2 & 4 \\ 7 & 6 \\ -1 & -3 \\ 0 & 1 \\ 3 & 8 \end{pmatrix}$$

Matrix Arithmetic

- Addition of two matrices

$$\vec{C} = \vec{A} + \vec{B} = \begin{pmatrix} a_{11}+b_{11} & \cdots & a_{1n}+b_{1n} \\ \vdots & & \vdots \\ a_{m1}+b_{m1} & \cdots & a_{mn}+b_{mn} \end{pmatrix}$$

- Scalar multiplication of a matrix

$$\vec{B} = d \cdot \vec{A} = \begin{pmatrix} d \cdot a_{11} & \cdots & d\,a_{1n} \\ \vdots & & \vdots \\ d\,a_{m1} & \cdots & d\,a_{mn} \end{pmatrix}$$

Matrix Arithmetic

- Matrix-matrix multiplication

$$AB = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \Rightarrow \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21}, & \cdots \\ \cdots & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

- Multiplication is associative

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

- Multiplication is not commutative

$$A \cdot B \neq B \cdot A \quad \text{(generally)}$$

- Transposition rule

$$(A \cdot B)^T = B^T \cdot A^T$$

Matrix Arithmetic

- RULE: in any chain of matrix multiplications, the column dimension of one matrix in the chain must match the row dimension of the following matrix in the chain

$A_{3\times 5}$ $B_{5\times 5}$ $C_{3\times 1}$

$A_{3\times 5} B_{5\times 5} A^T_{5\times 3}$ ✓

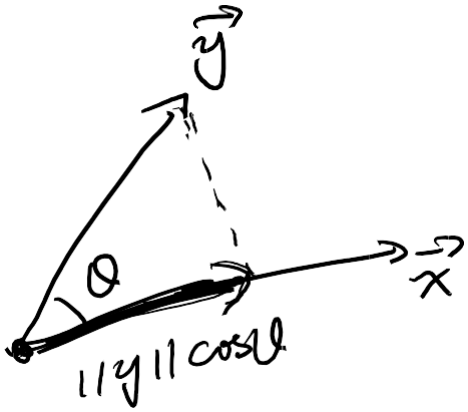
$C^T_{1\times 3} C_{3\times 1} A_{3\times 5}$ ✗

$C^T A B$ ✓

$A A^T B$ ✗

Vector Projection

- Orthogonal projector of **y** onto **x**
- Can take place in any space of dimensionality ≥ 2
- Unit vector in the direct of **x** is $\frac{\vec{x}}{\|\vec{x}\|}$



- The length of the projection of **y** in the direction of x is $\|y\| \cos\theta$
- Orthogonal projection of **y** onto **x** is the vector

$$\text{proj}_{\vec{x}}(\vec{y}) = \frac{\vec{x}}{\|\vec{x}\|} \cdot \|y\| \cdot \cos\theta = \frac{\vec{x}}{\|\vec{x}\|} \cdot \|\vec{y}\| \cdot \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \|\vec{y}\|} = \frac{(\vec{x} \cdot \vec{y})}{\|\vec{x}\|^2} \cdot \vec{x}$$

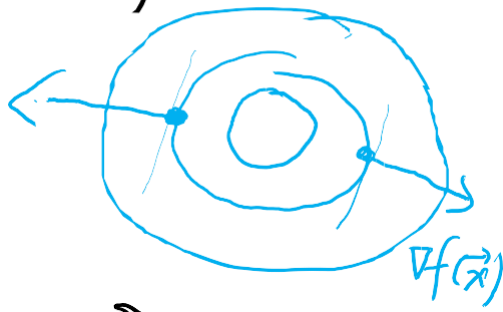
$$\vec{x} \cdot \vec{y} = \|\vec{x}\| \|\vec{y}\| \cos\theta$$

Multivariable Calculus

- Gradients – the single most important concept from calculus in the context of machine learning

$f(\vec{x})$ eg: $f = x_1^2 + x_2^2$ $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

$\nabla f(\vec{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$, i.e. $|\nabla f|_i = \frac{\partial f}{\partial x_i}$



↳ points in the direction of steepest ascent from $\vec{x}$

$-\nabla f(\vec{x})$ . . . . . descent from $\vec{x}$

Matrix Calculus

- A lot of optimization reduces to finding points where the gradients vanishes
- Two most important differentiation rules for matrix and vector expressions for machine learning

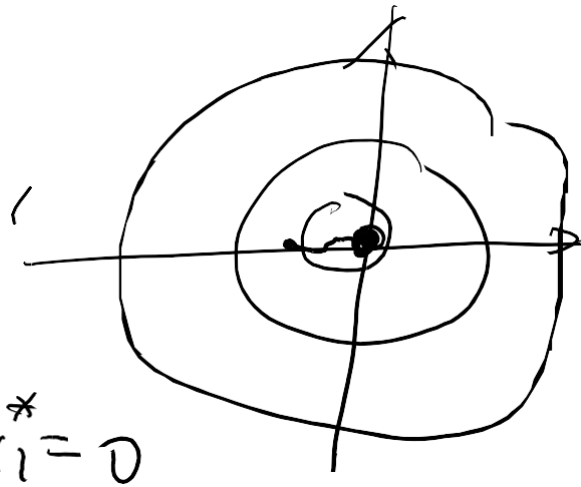
① $\nabla_{\vec{x}}(\vec{a}^T\vec{x}) = \vec{a}$ $\quad \vec{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}$ $\quad \vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$

② $\nabla_{\vec{x}}(\vec{x}^T A \vec{x}) = (A + A^T)\vec{x}$

Conditions for local Minima

if $\vec{x}^*$ is a local minimum of f, f is continuously differentiable in neighborhood of x^*.

$$\rightarrow \quad \nabla f(\vec{x}^*) = 0$$



$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad f = x_1^2 + x_2^2$$

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{matrix} x_1^* = 0 \\ x_2^* = 0 \end{matrix}$$

The Concept of Probability

- **The concept of probability**

- In some process, several outcomes are possible. When the process is repeated a large number of times, each outcome occurs with a characteristic *relative frequency*, or *probability*. If a particular outcome happens more often than another outcome, we say it is more probable.

Probability Spaces

- A *probability space* is a *random process* or *experiment* with three components:
- Ω, the set of possible *outcomes O*
  - number of possible outcomes = $|\Omega| = N$
- *F*, the set of possible *events E*
  - an event comprises 0 to *N* outcomes
  - number of possible events = $|F| = 2^N$
- *P*, the *probability distribution*
  - function mapping each outcome and event to real number between 0 and 1 (the *probability* of *O* or *E*)
  - probability of an event is *sum* of probabilities of possible outcomes in event

→ 1, 2, 3, 4, 5, 6

→

events : outcomes are odd #

$E = \{1, 3, 5\}$

Axioms of Probability

- Non-negativity:

  for any event, $E \in F$, $P(E) \geq 0$

- All possible outcomes:

  $P(\Omega) = 1$

- Additivity of disjoint events:

  for all events $E, E' \in F$ where $E \cap E' = \emptyset$

  $P(E \cup E') = P(E) + P(E')$

Types of Probability Spaces

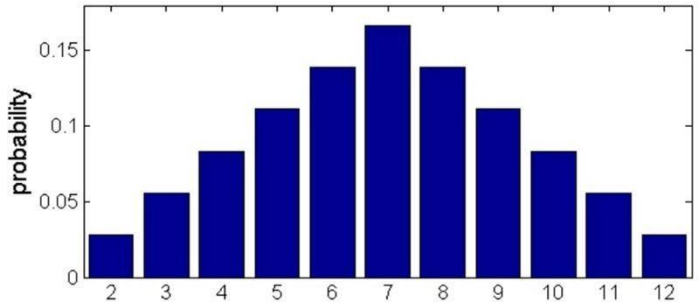
- Define $|\Omega|$ = number of possible outcomes
- Discrete space: $|\Omega|$ is finite, Analysis involves *summations* ($\sum$)

- Continuous space $|\Omega|$ is infinite, Analysis involves *integrals* ($\int$)

Probability Distributions

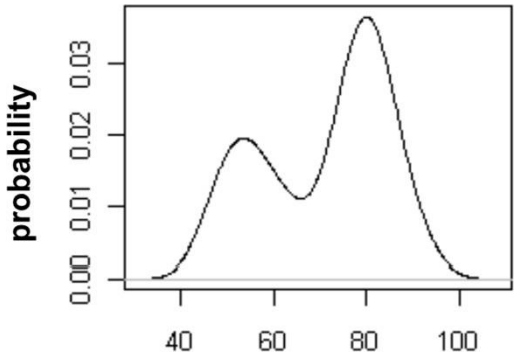
- Discrete: probability mass function (pmf)

Example: sum of two fair dice



- Continuous: probability density function (pdf)

Example: waiting time between eruptions of Old Faithful



Multivariate Probability Distributions

- Scenario
  –Several random processes occur (doesn't matter whether in parallel or in sequence)
  –Want to know probabilities for each possible combination of outcomes
- Can describe as *joint probability* of several random variables
- – Example: two processes whose outcomes are represented by random variables *X* and *Y*. Probability that process *X* has outcome *x* and process *Y* has outcome *y* is denoted as:

$$P(X = x, Y = y)$$

Multivariate Probability Distributions

- *Marginal* probability
  - Probability distribution of a single variable in a joint distribution

$$P(X=x) = \sum_{b=\text{all values of } Y} P(X=x, Y=b)$$

- *Conditional* probability

  Probability distribution of one variable *given* that another variable takes a certain value

$$p(X=x \mid Y=y) = \frac{P(X=x, Y=y)}{P(Y=y)}$$

Expected Value

- Given:
  - A discrete random variable X, with possible values $x = x_1, x_2, \ldots x_n$
  - Probabilities $p(X = x_i)$ that X takes on the various values of x_i
  - A function $y_i = f(x_i)$ defined on X

- The *expected value* of f is the probability-weighted "average" value of $f(xi)$:

$$E(f) = \sum_i P(x_i) f(x_i)$$

Common Forms of Expected Value

- Mean

$$\mu = E(f) = \sum_i p(x_i)\, x_i$$

measures the center of a distribution

- Variance

$$\sigma^2 = \sum_i p(x_i) \cdot (x_i - \mu)^2$$

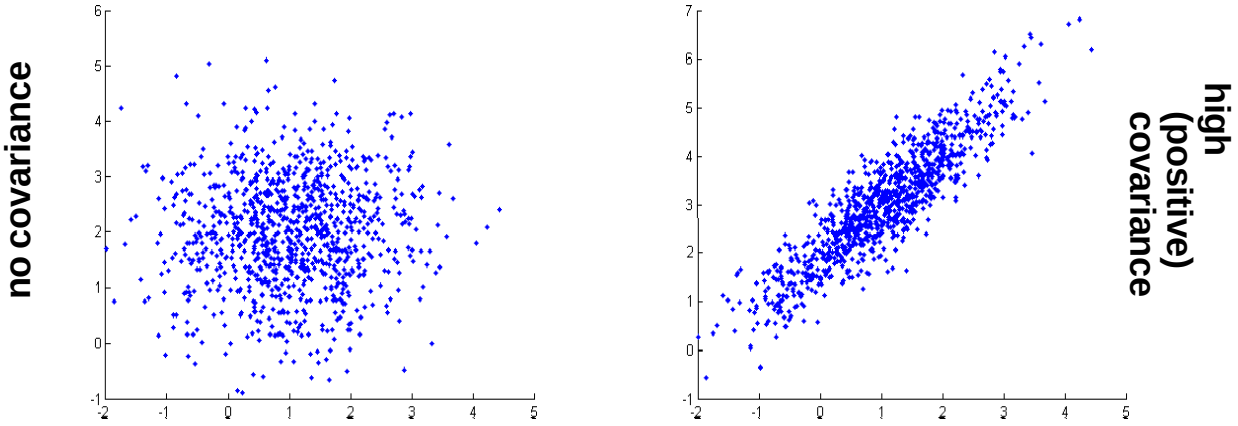
spread

σ = standard

Common Forms of Expected Value

- Covariance

$$Cov(x, y) = \sum_i p(x_i, y_i)(x_i - \mu_x)(y_i - \mu_y)$$



Example

$(x, y) \in S = \{(5, 8), (6, 8), (7, 8), (5, 9), (6, 9), (7, 9)\}$:

$f(x, y)$		x			$f_Y(y)$
		5	**6**	**7**	
y	**8**	0	0.4	0.1	0.5
	9	0.3	0	0.2	0.5
$f_X(x)$		0.3	0.4	0.3	1

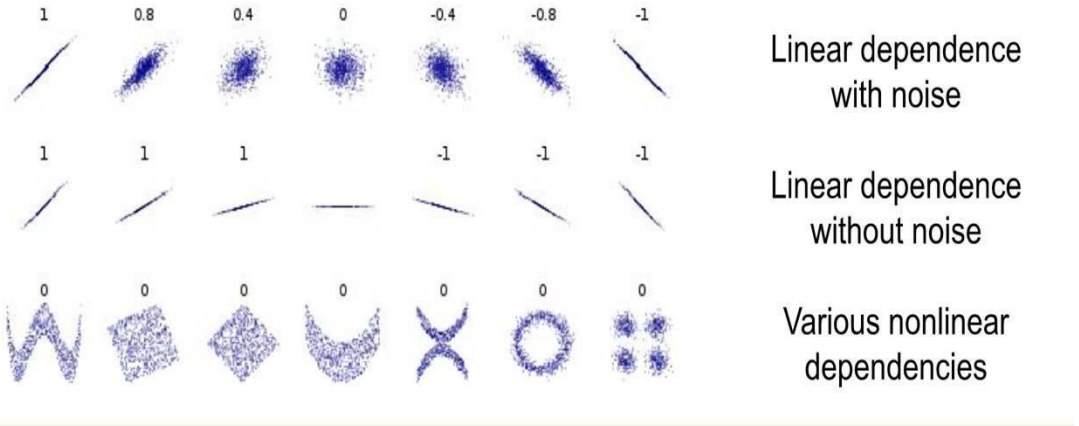
X can take on three values (5, 6 and 7) while Y can take on two (8 and 9). Their means are $\mu_X = 5(0.3) + 6(0.4) + 7(0.1 + 0.2) = 6$ and $\mu_Y = 8(0.4 + 0.1) + 9(0.3 + 0.2) = 8.5$. Then,

$$\begin{aligned}
\operatorname{cov}(X, Y) = \sigma_{XY} &= \sum_{(x,y)\in S} f(x, y)\,(x - \mu_X)\,(y - \mu_Y) \\
&= (0)(5 - 6)(8 - 8.5) + (0.4)(6 - 6)(8 - 8.5) + (0.1)(7 - 6)(8 - 8.5) + \\
&\quad (0.3)(5 - 6)(9 - 8.5) + (0)(6 - 6)(9 - 8.5) + (0.2)(7 - 6)(9 - 8.5) \\
&= -0.1\ .
\end{aligned}$$

Correlation

- Pearson's correlation coefficient is covariance normalized by the standard deviations of the two variables

$$corr(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$



Rules

- Complement rule

$$P(\text{not } A) = 1 - P(A)$$

- Product rule

$$P(A, B) = P(A|B)P(B)$$

- Rule of total Probability

$$P(A) = P(A, B) + P(A, \text{not } B)$$

$$P(A) = \sum_n P(A \cap B_n) = \sum_n P(A|B_n)P(B_n)$$

Bayes Rule

- A way to find conditional probabilities for one variable when conditional probabilities for another variable are known.

$$P(B|A) = \frac{P(A|B)\,P(B)}{P(A)}$$

A related experiment

Consider a 'bin' with red and green marbles.

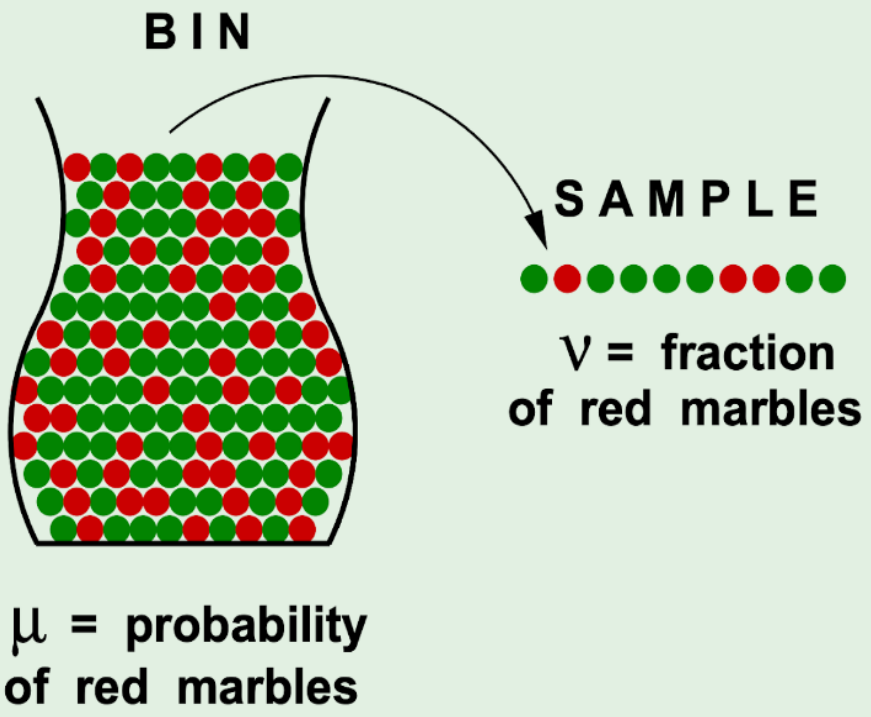
$$\mathbb{P}[\text{ picking a red marble }] = \mu$$

$$\mathbb{P}[\text{ picking a green marble }] = 1 - \mu$$

The value of μ is unknown to us.

We pick N marbles independently.

The fraction of red marbles in sample $= \nu$



What does ν say about μ?

In a big sample (large N), ν is probably close to μ (within ϵ).

Formally,

$$\mathbb{P}\left[\,|\nu - \mu| > \epsilon\right] \leq 2e^{-2\epsilon^2 N}$$

This is called **Hoeffding's Inequality**.

In other words, the statement "$\mu = \nu$" is P.A.C.

Machine Learning Tasks

Given a data set of instances of size *N*, create a model that is fit from the data (built) by extracting features and dimensions. Then use that model to predict outcomes …

1. Data Wrangling (normalization, standardization, imputing)
2. Feature Analysis/Extraction
3. Model Selection/Building
4. Model Evaluation
5. Operationalize Model

Components of Learning

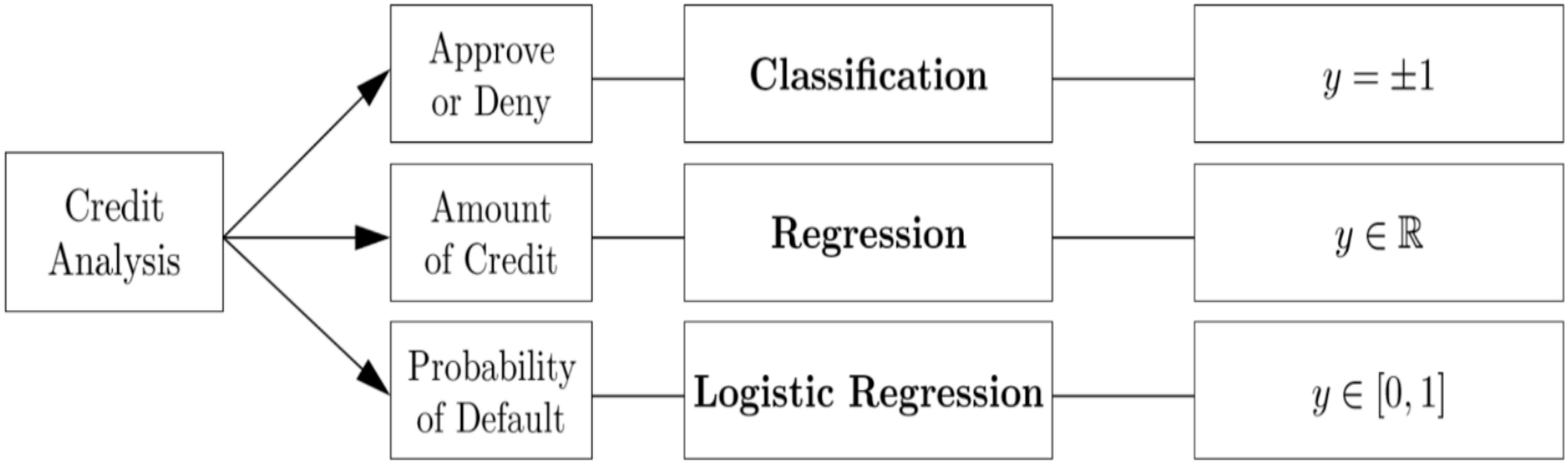
Formalization:

- Input: $\mathbf{x}$ (*customer application*)
- Output: y (*good/bad customer?*)
- Target function: $f : \mathcal{X} \to \mathcal{Y}$ (*ideal credit approval formula*)
- Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$ (*historical records*)

$\downarrow \quad \downarrow \quad \downarrow$

- Hypothesis: $g : \mathcal{X} \to \mathcal{Y}$ (*formula to be used*)

Three Learning Problems



- Linear models are perhaps ***the*** fundamental model.
- The linear model is the first model to try.

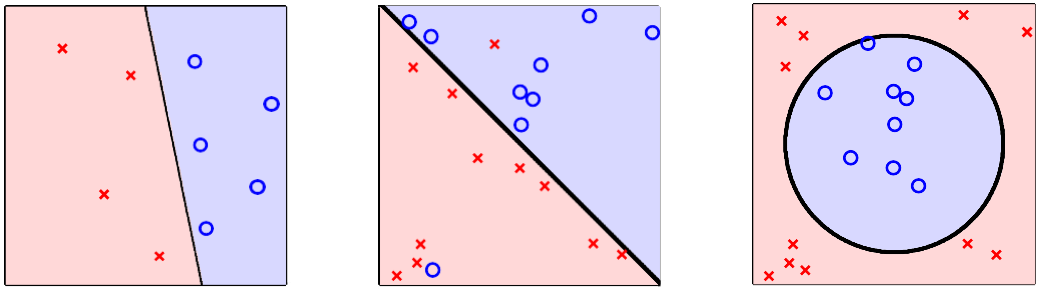
Types of Learning

- Supervised learning
  - the data is (**x**, f(**x**)) -- you are told the **answer**
- Unsupervised learning
  - only given **x**, learn to "organize" the data
- Reinforcement learning
  - you get feedback on potential answers you try
  - **x** → try something → get feedback

Supervised Learning -- Binary Classification Problems

- PLA A takes linear separable D and perceptrons H to get hypothesis g

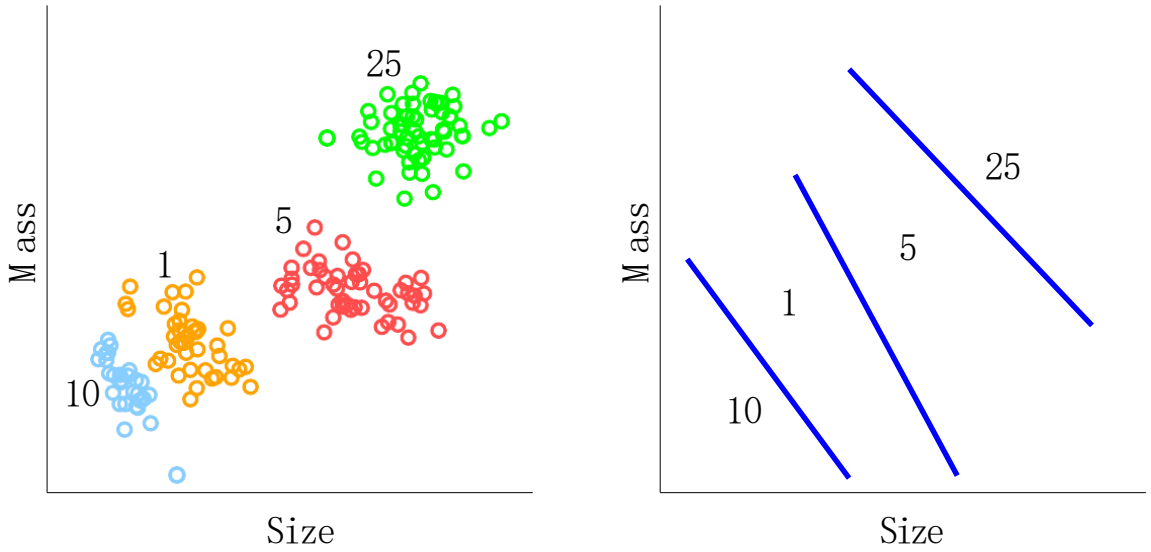
- Credit approve / disapprove
- Email spam / non-spam
- Patient sick / not sick
- Ad profitable / not profitable
- Answer correct / incorrect



- Core and important problem with many tools as building blocks of other tools, e.g., multiclass classification

Supervised Learning – Multiclass Classification

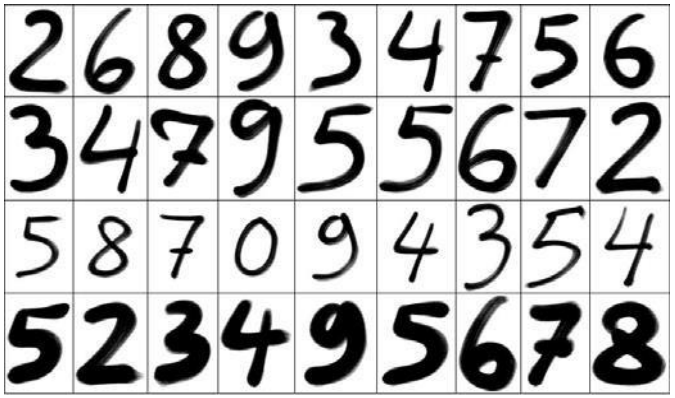
- Classify US coins (1c, 5c, 10c, 25c) by (size, mass)
- Y = {1c, 5c, 10c, 25c}



- Supervised learning: every x_i comes with corresponding y_i

Supervised Learning – Multiclass Classification

- Other classification problems
  - Written digits → 0, 1, …, 9
  - Pictures → apple, orange, strawberry
  - Emails → spam, primary, social, promotion



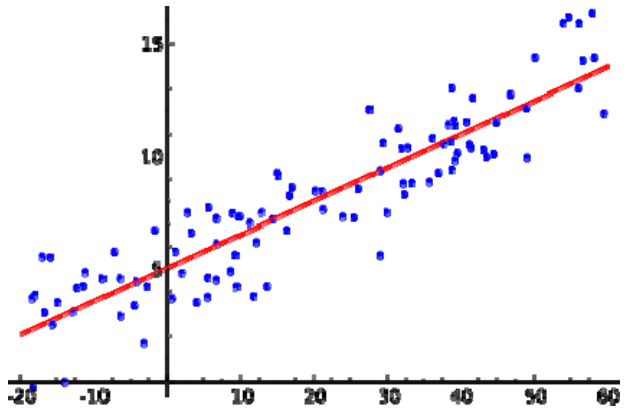
- Many applications in practice, especially for ‘recognition’

Supervised Learning – Regression

- binary classification: patient features ⇒ sick or not
- multiclass classification: patient features ⇒ which type of cancer
- regression: patient features ⇒ how many days before recovery
- $Y = R$ or $Y = [\text{lower}, \text{upper}] \subset R$ (bounded regression)

  —deeply studied in statistics

- Other regression problems
  - Company data → stock price
  - Climate data → temperature



Unsupervised Learning

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

We can derive this structure by clustering the data based on relationships among the variables in the data.

With unsupervised learning there is no feedback based on the prediction results.

Example

Clustering: Take a collection of 1,000,000 different genes, and find a way to automatically group these genes into groups that are somehow similar or related by different variables, such as lifespan, location, roles, and so on.

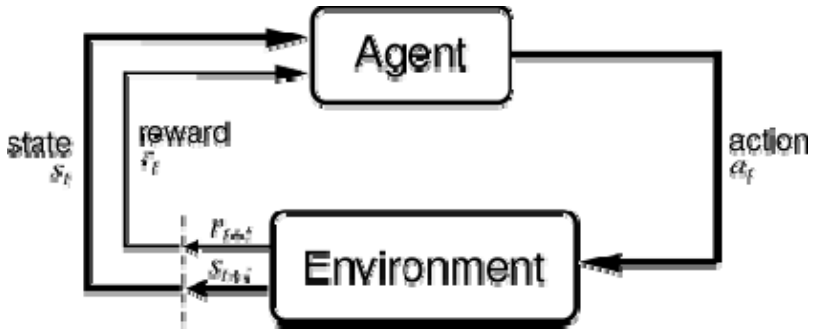
Reinforcement Learning

- Teach your dog: Say “Sit Down”
  - The dog runs away – BAD dog. No cookies.
  - The dog sits down – GOOD dog. Let me give you some cookies.

- Cannot easily show the dog that y_n = sit when x_n = “sitdown”
- But can “punish” or “reward”

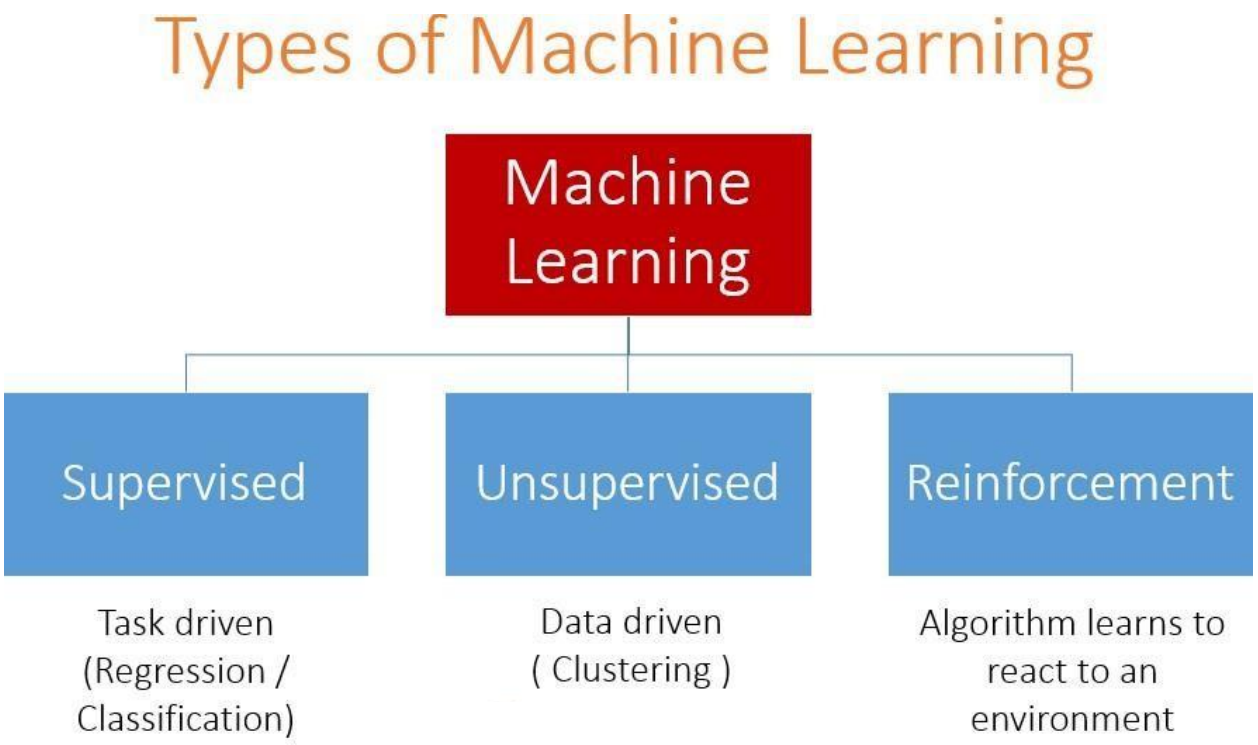
Reinforcement Learning

- Reinforcement learning
- learn with "partial/implicit information" (often sequentially)
- a very 'different' but natural way of learning



- Other reinforcement learning problems
  - (customer, ad choice, ad click earning) → ad system
  - (cards, strategy, winning amount) → black jack agent

Summary


Types of Machine Learning
Machine Learning
Supervised
Unsupervised
Reinforcement
Task driven (Regression / Classification)
Data driven (Clustering)
Algorithm learns to react to an environment

Exercise

- What is this learning problem?
- To build a tree recognition system, a company decides to gather one million of pictures on the Internet. Then, it asks each of the 10 company members to view 100 pictures and record whether each picture contains a tree. The pictures and records are then fed to a learning algorithm to build the system. What type of learning problem does the algorithm need to solve?

A. Supervised
B. Unsupervised
C. Semi-supervised
D. Reinforcement

A Simple Learning Model – Credit Approval Problem

- Input vector $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$
- Give importance weights to the different inputs and compute a "Credit Score"

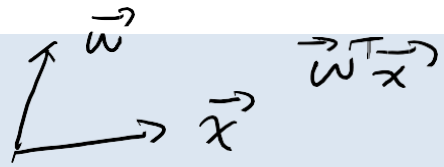
$$\text{"credit score"} = \sum_{i=1}^{d} w_i x_i \quad \begin{cases} > \text{threshold} & \checkmark \\ < \text{threshold} & \times \end{cases}$$

Customer 1	
age	24 years
gender	female
annual salary	$60,000
years in residence	1 year
years in job	0.5 year
current debt	$10,000
…	…

How to choose w_i ?

- input x_i is important, $|w_i|$ large weight
- x_i is beneficial for credit $\Rightarrow$ positive weight $w_i > 0$
- x_i is detrimental for credit $\Rightarrow$ negative weight $w_i < 0$

Perceptron Hypothesis



- A Hypothesis set $H = \{h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})\}$ → $\{\text{sign}(\vec{w}_1^T\vec{x}), \text{sign}(\vec{w}_2^T\vec{x}) \cdots \text{sign}(\vec{w}_M^T\vec{x})\}$
- This hypothesis set is call the perceptron or linear separator

$$h(\vec{x}) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i - \text{threshold}\right)$$

$$= \text{sign}\left(\sum_{i=1}^{d} w_i x_i + \underbrace{(-\text{threshold})}_{w_0}\cdot\underbrace{(+1)}_{x_0}\right)$$

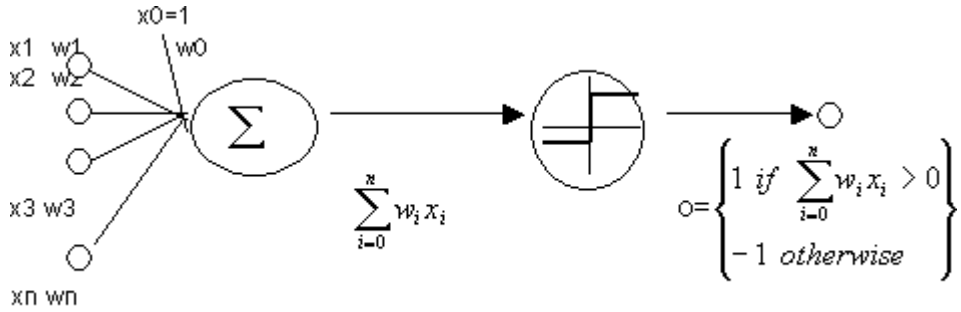
$$= \text{sign}\left(\sum_{i=0}^{d} w_i x_i\right)$$

$$= \text{sign}(\vec{w}^T\vec{x})$$

$$\vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{pmatrix} \quad \vec{x} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{pmatrix}$$

find g

$g \approx f$



[A perceptron]

Perceptrons in $\boldsymbol{R}^2$

- $h(\mathbf{x}) = \text{sign}\,(w_0 + w_1 x_1 + w_2 x_2)$

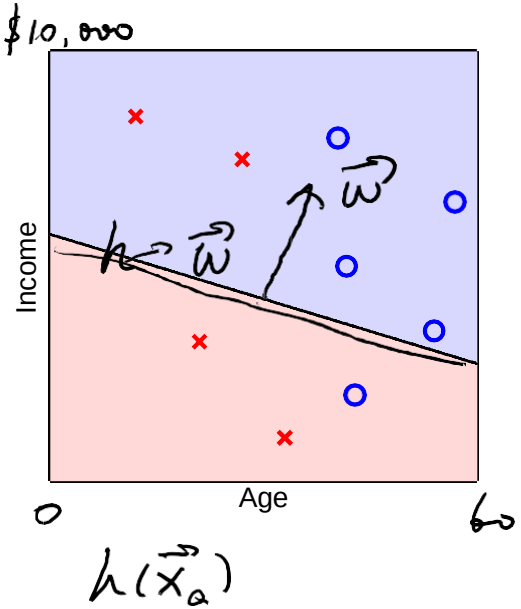
customer features

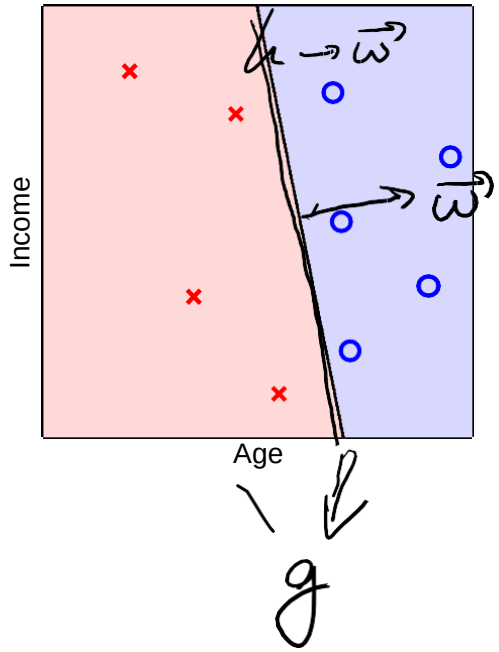
$\vec{x}$: points on the plane

labels y : (+1) ○ , (−1) ×

hypothesize h : lines

→ positive on one side of a line, negative on the other





pecoptrons ⟺ linear (binary) classifiers

Select *g* From *H*

- *H* = all possible perceptrons, *g* = ?

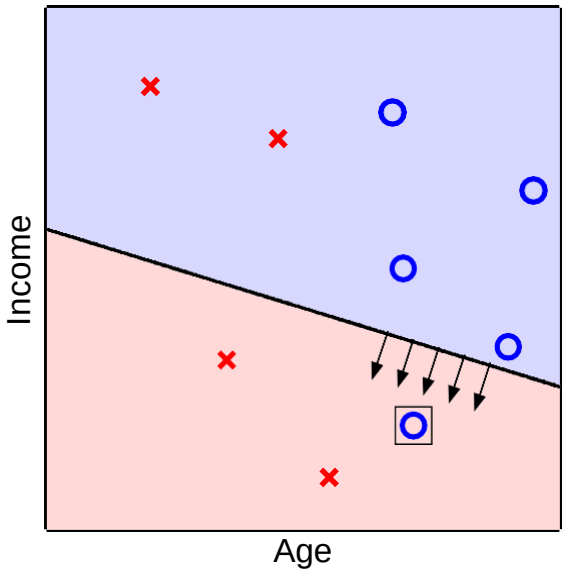
We want to select $g \in H$ so that $g \approx f$

We certainly want $g \simeq f$ on the set D,

$g(\vec{x}_i) = y_i$

Idea: Start with some g_0, and "correct" its mistakes on D

will represent g_0 by $\vec{w}_0$ → $\text{sign}(\vec{w}^T \vec{x})$



Perceptron Learning Algorithm (PLA)

- A simple iterative algorithm

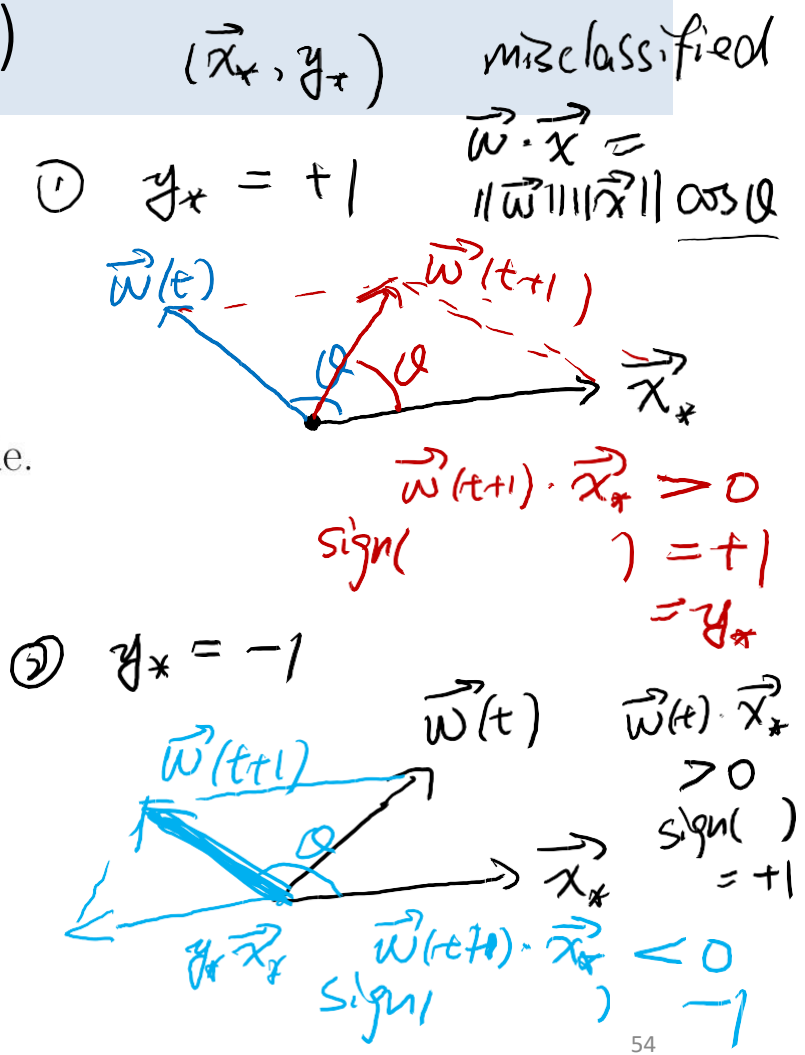
1: $\mathbf{w}(1) = \mathbf{0}$
2: **for** iteration $t = 1, 2, 3, \ldots$
3: the weight vector is $\mathbf{w}(t)$.
4: From $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ pick any misclassified example.
5: Call the misclassified example $(\mathbf{x}_*, y_*)$,

$$\text{sign}\left(\mathbf{w}(t) \bullet \mathbf{x}_*\right) \neq y_*.$$

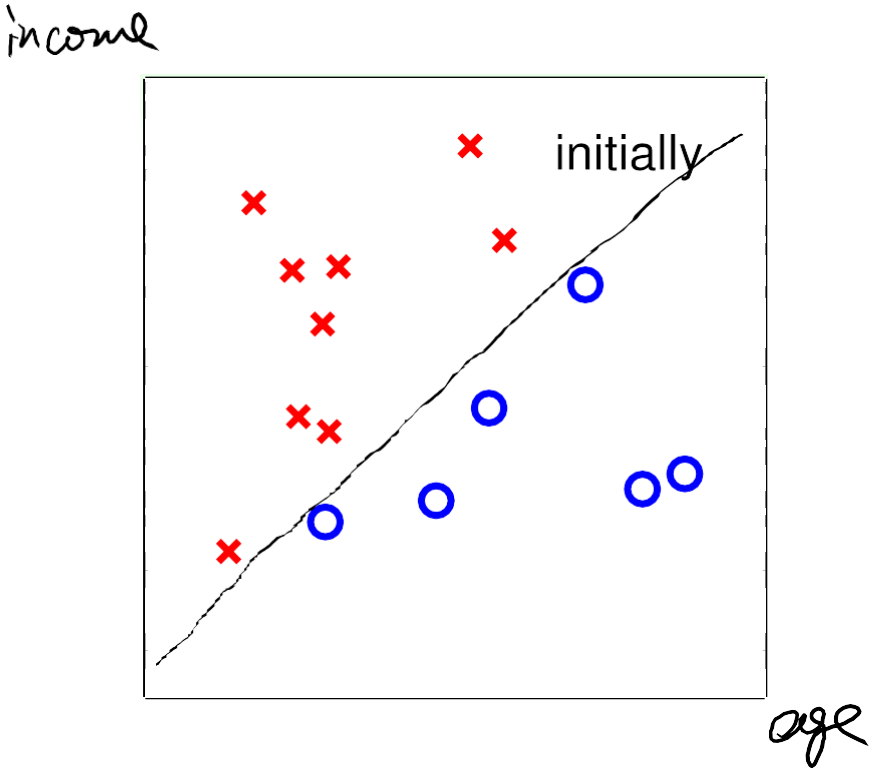
6: Update the weight:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + y_* \mathbf{x}_*.$$

7: $t \leftarrow t + 1$

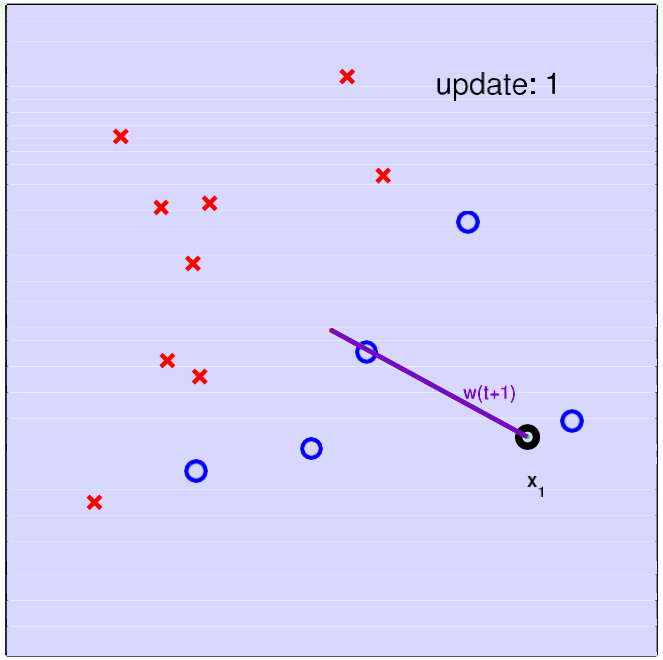


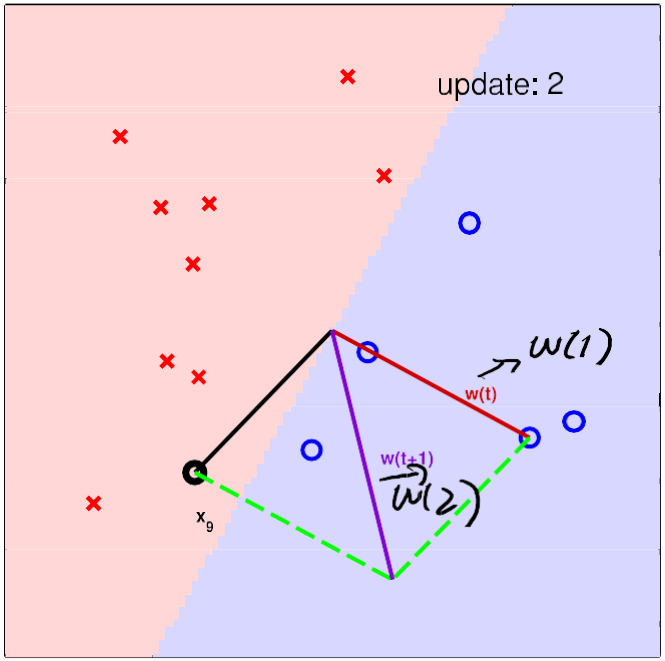
How PLA works?



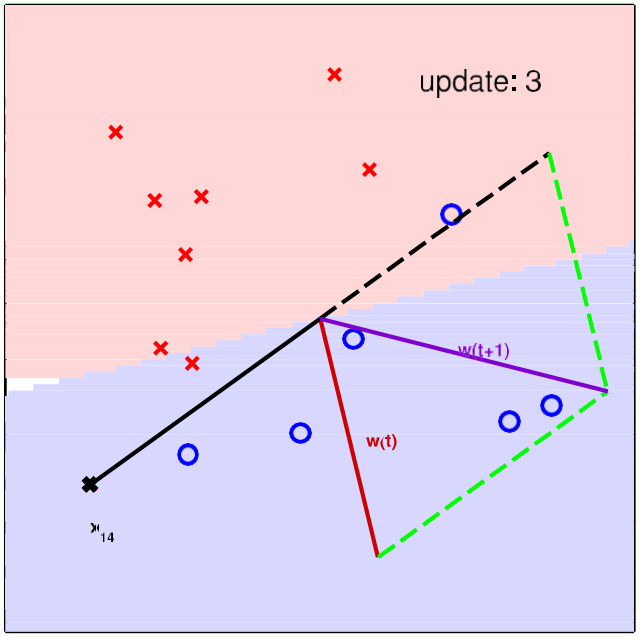
learn $\vec{w}$ → g

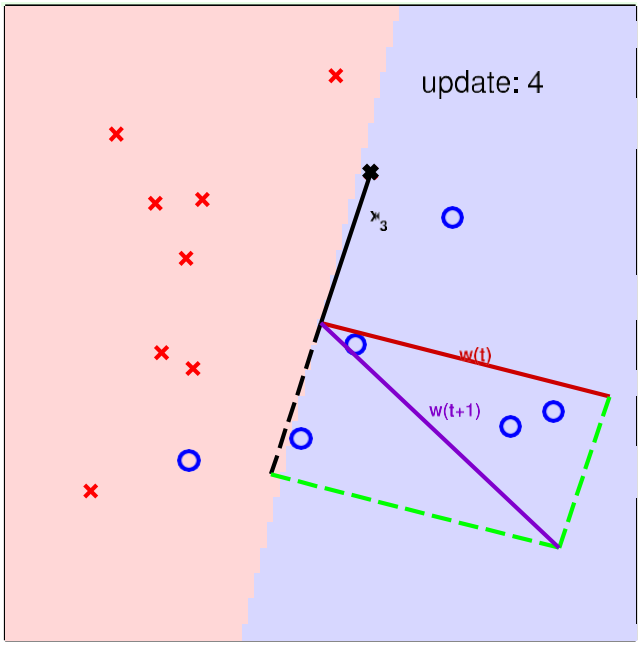
How PLA Works





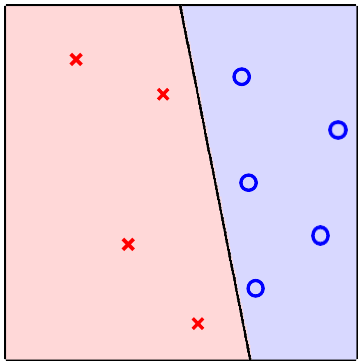
How PLA Works



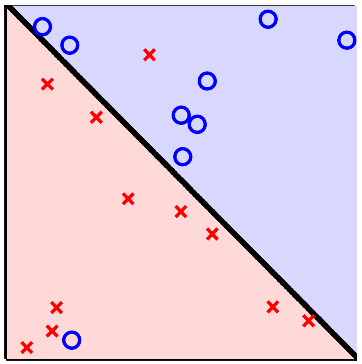


Linear Separability

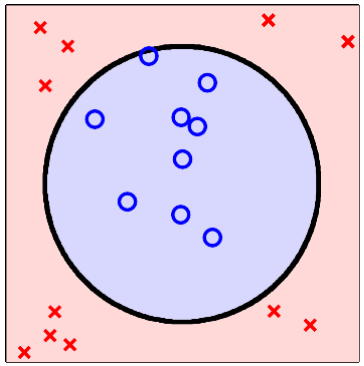
- If PLA stops (i.e., no more mistakes),
  (necessary condition) D allows some **w** to make no mistake
- Call such D linearly separable



(linearseparable)



(notlinear separable)



(notlinear separable)

A Tour of Machine Learning Algorithms

Models: Instance Methods

Compare instances in data set with a similarity measure to find best matches.

- Suffers from curse of dimensionality.
- Focus on feature representation and similarity metrics between instances

- **k-Nearest Neighbors (kNN)**
- **Self-Organizing Maps (SOM)**
- **Learning Vector Quantization (LVQ)**

Models: Regression

Model relationship of independent variables, X to dependent variable Y by iteratively optimizing error made in predictions.

- **Ordinary Least Squares**
- **Logistic Regression**
- **Stepwise Regression**
- **Multivariate Adaptive Regression Splines (MARS)**
- **Locally Estimated Scatterplot Smoothing (LOESS)**

Models: Regularization Methods

Extend another method (usually regression), penalizing complexity (minimize overfit)

- simple, popular, powerful
- better at generalization

- **Ridge Regression**
- **LASSO (Least Absolute Shrinkage & Selection Operator)**
- **Elastic Net**

Models: Decision Trees

Model of decisions based on data attributes. Predictions are made by following forks in a tree structure until a decision is made. Used for classification & regression.

- **Classification and Regression Tree (CART)**
- **Decision Stump**
- **Random Forest**
- **Gradient Boosting Machines (GBM)**

Models: Bayesian

Explicitly apply Bayes' Theorem for classification and regression tasks. Usually by fitting a probability function constructed via the chain rule and a naive simplification of Bayes.

- **Naive Bayes**
- **Averaged One-Dependence Estimators (AODE)**
- **Bayesian Belief Network (BBN)**

Models: Kernel Methods

Map input data into higher dimensional vector space where the problem is easier to model.
Named after the “kernel trick” which computes the inner product of images of pairs of data.

- **Support Vector Machines (SVM)**
- **Radial Basis Function (RBF)**
- **Linear Discriminant Analysis (LDA)**

Models: Clustering Methods

Organize data into into groups whose members share maximum similarity (defined usually by a distance metric). Two main approaches: centroids and hierarchical clustering.

- **k-Means**
- **Affinity Propagation**
- **OPTICS (Ordering Points to Identify Cluster Structure)**
- **Agglomerative Clustering**

Models: Artificial Neural Networks

Inspired by biological neural networks, ANNs are nonlinear function approximators that estimate functions with a large number of inputs.

- System of interconnected neurons that activate
- Deep learning extends simple networks recursively

- **Perceptron**
- **Back-Propagation**
- **Hopfield Network**
- **Restricted Boltzmann Machine (RBM)**
- **Deep Belief Networks (DBN)**

Models: Ensembles

Models composed of multiple weak models that are trained independently and whose outputs are combined to make an overall prediction.

- **Boosting**
- **Bootstrapped Aggregation (Bagging)**
- **AdaBoost**
- **XGBoost**
- **Stacked Generalization (blending)**
- **Gradient Boosting Machines (GBM)**
- **Random Forest**

Models: Other

The list before was not comprehensive, other algorithm and model classes include:

- **ARIMA (**AutoRegressive Integrated Moving Average)

- **Conditional Random Fields (CRF)**
- **Markovian Models (HMMs)**
- **Dimensionality Reduction (PCA, PLS)**
- **Rule Learning (Apriori, Brill)**
- **Recommender system**
- **More ...**

What is Scikit-Learn?

Extensions to SciPy (Scientific Python) are called SciKits. SciKit-Learn provides machine learning algorithms.

- Algorithms for supervised & unsupervised learning
- Built on SciPy and Numpy
- Standard Python API interface
- Open Source: BSD License (part of Linux)

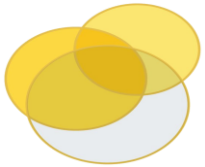
Probably the best general ML framework out there.

Where did it come from?

Started as a Google summer of code project in 2007 by David Cournapeau, then used as a thesis project by Matthieu Brucher.

In 2010, INRIA pushed the first public release, and sponsors the project, as do Google, Tinyclues, and the Python Software Foundation.

Who uses Scikit-Learn?



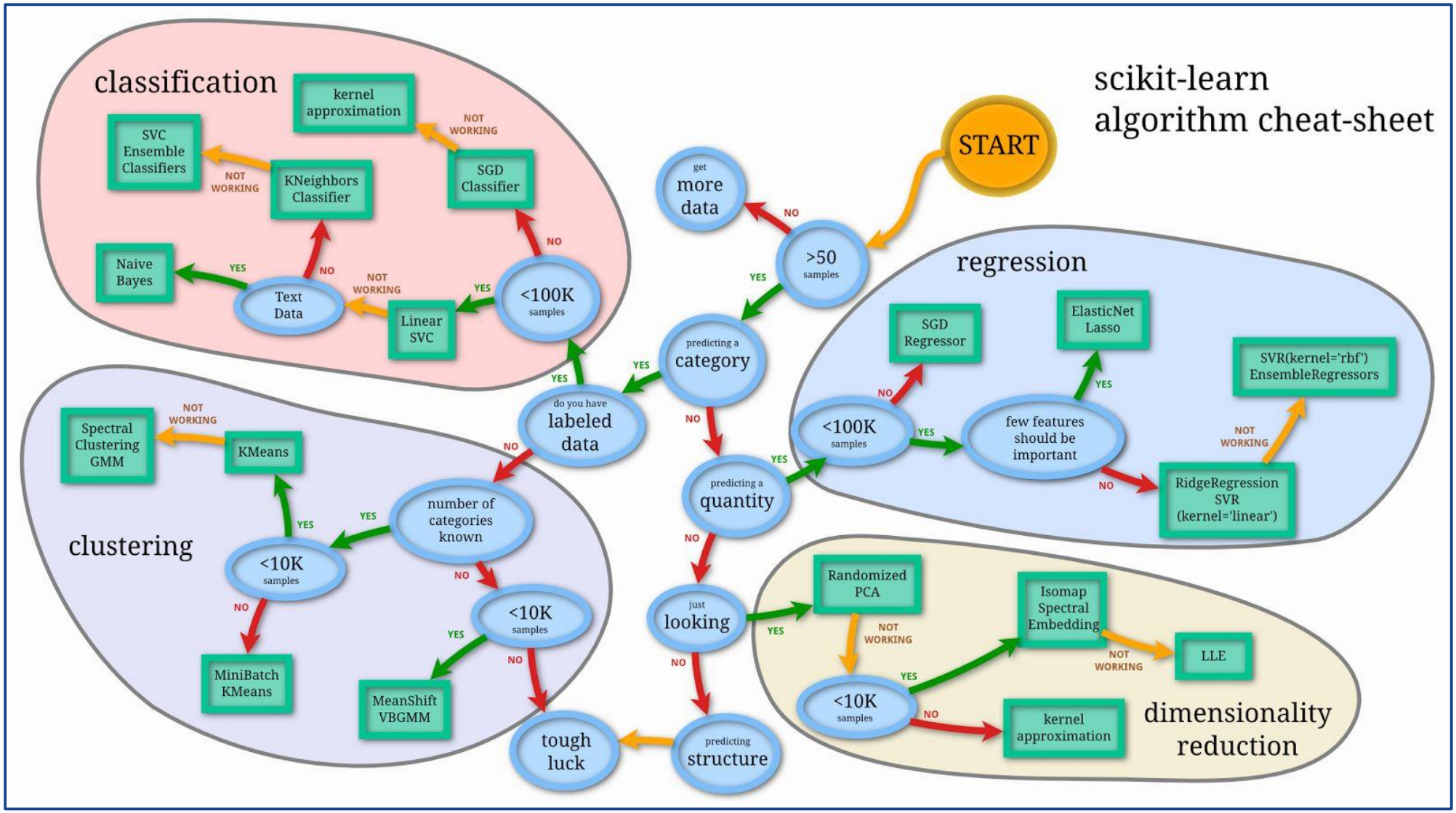
SUNLIGHT FOUNDATION

DATA PUBLICA

DataRobot

Primary Features

- Generalized Linear Models
- SVMs, kNN, Bayes, Decision Trees, Ensembles
- Clustering and Density algorithms
- Cross Validation
- Grid Search
- Pipelining
- Model Evaluations
- Dataset Transformations
- Dataset Loading



A Guide to Scikit-Learn

https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-Learn API

O*bject-oriented* interface centered around the concept of an *Estimator*:

> “An estimator is any object that learns from data; it may be a classification, regression or clustering algorithm or a transformer that extracts/filters useful features from raw data.”

- Scikit-Learn Tutorial

https://scikit-learn.org/stable/tutorial/index.html

```
class Estimator(object):

    def fit(self, X, y=None):
        """Fits estimator to data. """
        # set state of ``self``
        return self

    def predict(self, X):
        """Predict response of ``X``. """
        # compute predictions ``pred``
        return pred
```

The Scikit-Learn Estimator API

```
from sklearn import svm

estimator = svm.SVC(gamma=0.001)
estimator.fit(X, y)
estimator.predict(x’)
```