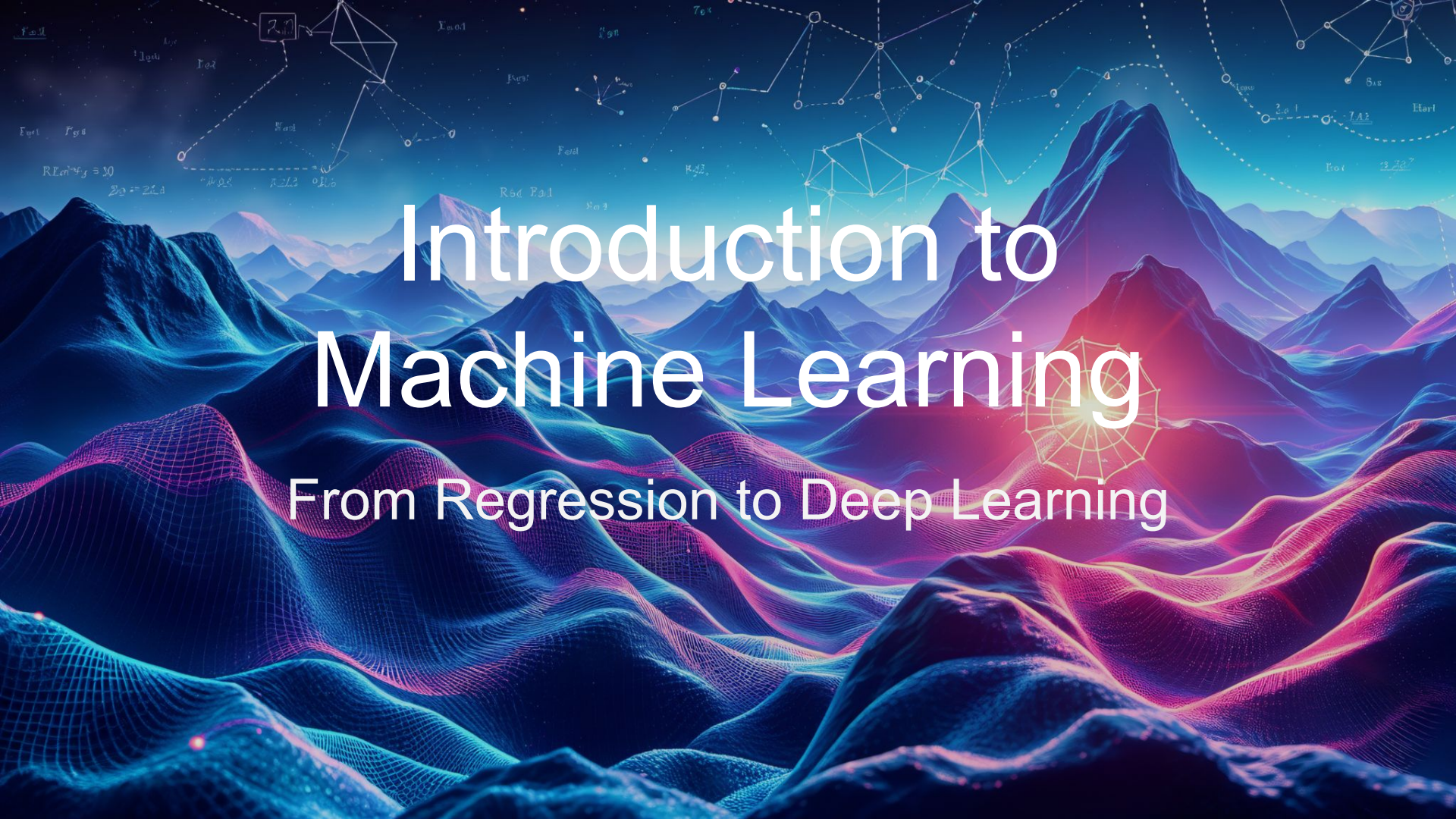
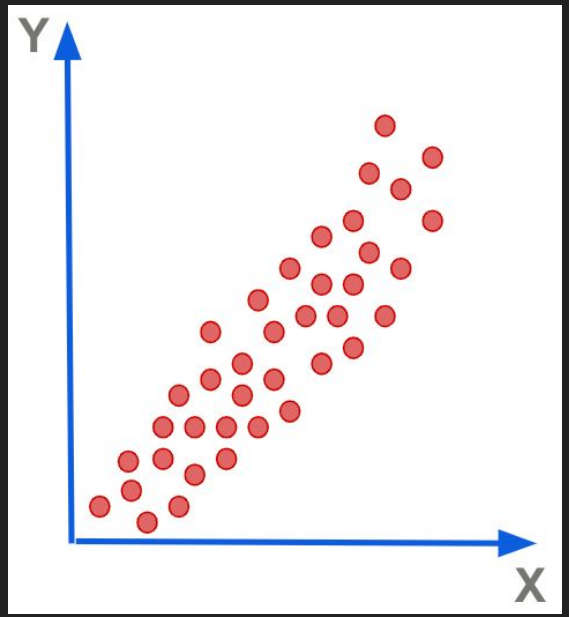

Introduction to
Machine Learning
From Regression to Deep Learning

What is machine learning?

Machine learning is a field of math / computer science that uses algorithms to learn from data (often to make predictions)

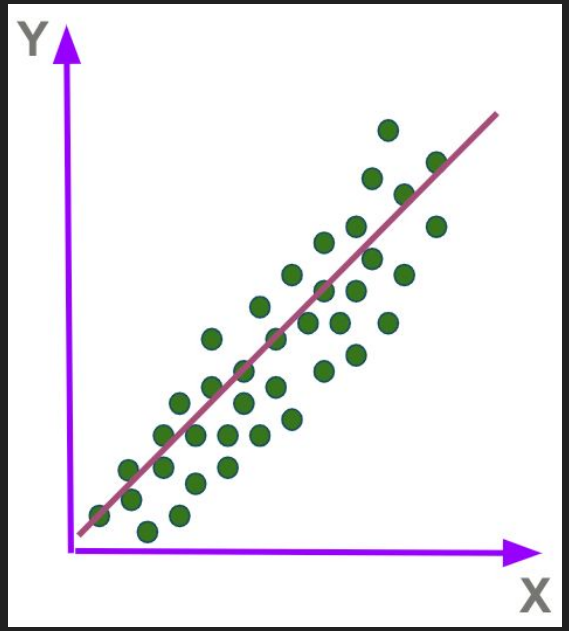
Let’s start with some data

- Let’s assume a dataset with two variables



Let’s start with some data

- This looks linear, so let’s run a linear regression on it



What we just did is a form of ML

Machine learning really extends from very simple concepts to large language models.

What is a linear regression in ML

- We want to solve Y = ax + b, and by this we implied *find a and b such that our error is minimal*
- This is a classic **statistics** problem, and a linear regression can be defined in a statistical manner
- However we can also define it as a machine learning problem

Linear regression as a machine learning problem

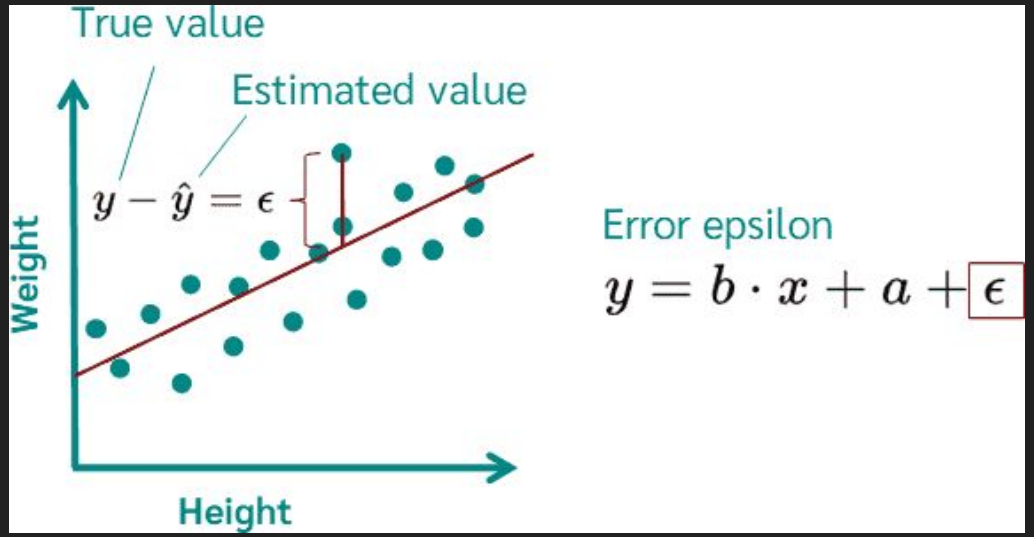
- ML is a form of statistical computation:
  - What is the **problem** we want to solve?
  - What does **Solving** mean?
  - What does **Error** mean?
  - What is the **algorithm**?
- To perform a machine learning operation we need the following elements:
  - An equation to solve
  - A list of parameters of said equation that we aim to find
  - A mathematical measure of error
  - An algorithm (a step wise computation) with a form of convergence
- Let’s see how we can find a and b in y = ax+b without stats

An algorithm to find a and b

- Machine learning applies algorithms at data in order to solve some underlying predefined equation
- This means that our algorithm should have the following properties:
  - Perform stepwise computations, which means that the final computation can be broken down into a series of small steps
  - Measure the error at each step to inform itself of its progress (if the error decrease, it's good, else it's bad)
  - Ideally know in advance in which direction it should move the parameters to decrease the error
  - Be resilient to noise
- In general we call such algorithm: **Solvers**
  - We'll get to learn more about them later

Error of linear regression

- Regression coefficient
- R and r2
- Y - y(hat)



Parameters we want to fit

- $Y = a x + b$
- We want to learn a and b

Live demo: notebook

- Notebook example: https://colab.research.google.com/drive/1c-y_KmrnVnMASyJ_zESgW_ID0B0oHBQD#scrollTo=tU_MVYBaVMSz
-

Problem, Loss and Solver

Machine Learning is always the interplay between a problem (equation), a loss function (error) and a solver (algorithm)

What is a problem?

If we want to learn from our data, then we must define a mathematical equation to represent it.

The landscape of potential equations is infinite

- Machine learning is not dependant on a particular equation, therefore many potential equations can be used to define a problem
- Just before we've seen a linear regression with a linear model: y = ax + b
- But we could also fit a quadratic model: y = ax^2 + bx + c
- Or an exponential model: y = exp(ax)
- Or any other equation that depends on a set of parameters: **a**, **b**, **c** etc

Machine learning techniques are often but a model

- You might have seen words like Random Forest, K-means, SVD, etc
- While learning the math behind each of those model is important and interesting, what is fundamental is to understand that they are all general purpose mathematical model that define a problem into mathematical terms
- Finally, don’t get scared by complex equations, most of those models have some nice intuitions and there exists a plethora of content online explaining them in layman’s terms

What is the error?

What we often define as the loss function, or the cost function is a measure of error

The loss function is just a measure of error

- You might have seen the words Loss function or Cost function, what the ML world means when talking about this is the deviation between the model and the data.
- In other words we are measuring the prediction error between our model and our data.
- Error is a term often used in physics to represent the deviation between a model and reality, for example, if I throw a ball, it will follow newton's law, but maybe there is some wind that changes its trajectory slightly. The landing spot of the ball (data) therefore is slightly different from my prediction (model)
- **The loss function is the mathematical formula used to measure this error.**

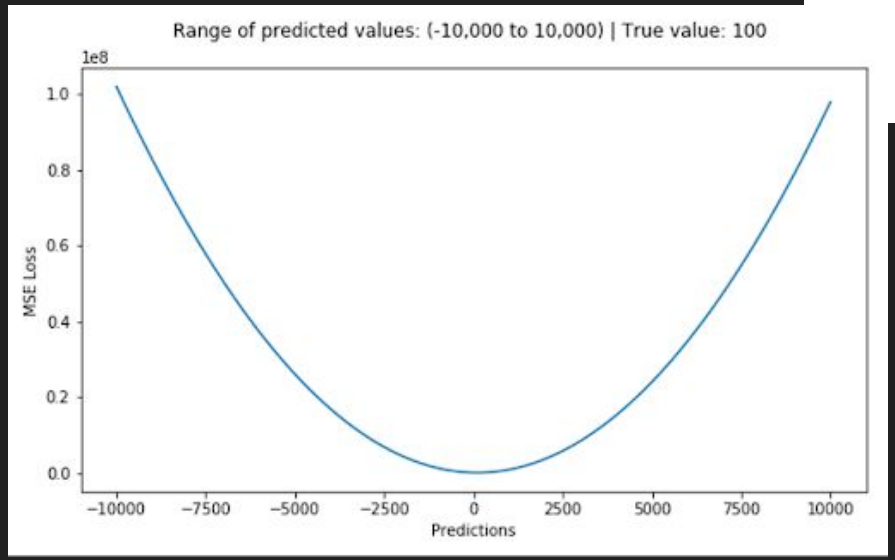
Loss Functions are defined by our problem

- Machine learning problems are often classified into two types: Regression and Classification
  - Regression: If we aim to learn a continuous function representing continuous data (height, weight, growth, etc) then we will need to use a class of loss function that are well suited for this
    - Example: Mean Squared Error
  - Classification: If we want to learn to classify items into categories (species, colors, state, etc) then we must use another type of loss function that can take into account all potential categories and measure whether we “sorted” our data correctly.
    - Example: Cross Entropy Loss

Mean square error

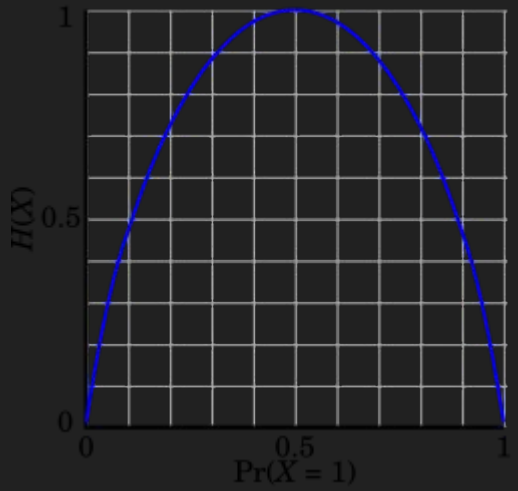
- Regression

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$



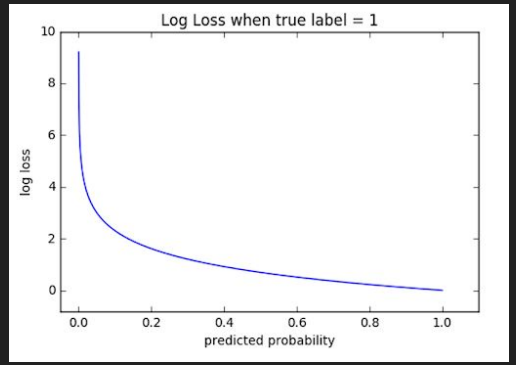
Entropy

Coin



Cross Entropy

- Classification



$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

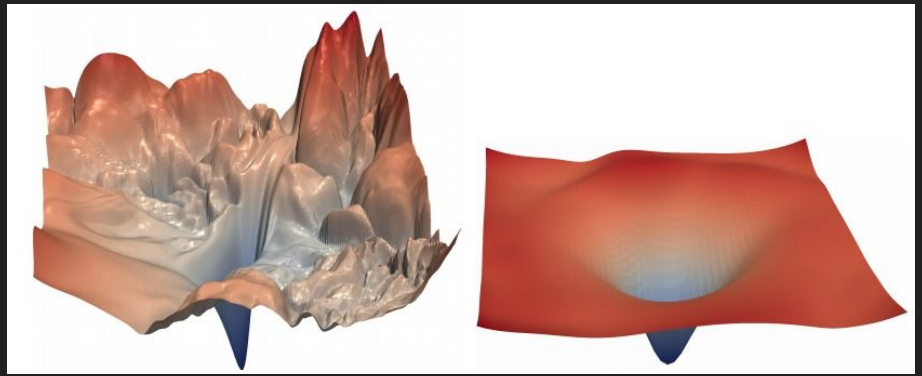
So many more

- MAE, Hinge, Log Loss, etc
- There exists many different loss functions, each of which will be best applied to a particular problem
- The only thing that matters is that you chose a loss function that approximates what you are aiming to minimize.

Loss Function	Applicability to Classification	Applicability to Regression
Mean Square Error (MSE) / L2 Loss	✗	✓
Mean Absolute Error (MAE) / L1 Loss	✗	✓
Binary Cross-Entropy Loss / Log Loss	✓	✗
Categorical Cross-Entropy Loss	✓	✗
Hinge Loss	✓	✗
Huber Loss / Smooth Mean Absolute Error	✗	✓
Log Loss	✓	✗

The loss landscape

- We define the loss landscape as the shape of all possible loss values given our parameters.
- From an intuitive point of view, we aim to find the minimum, the “lowest valley” in a mathematical landscape.
- The landscape can take on many shapes, and it is very “expensive” to compute it

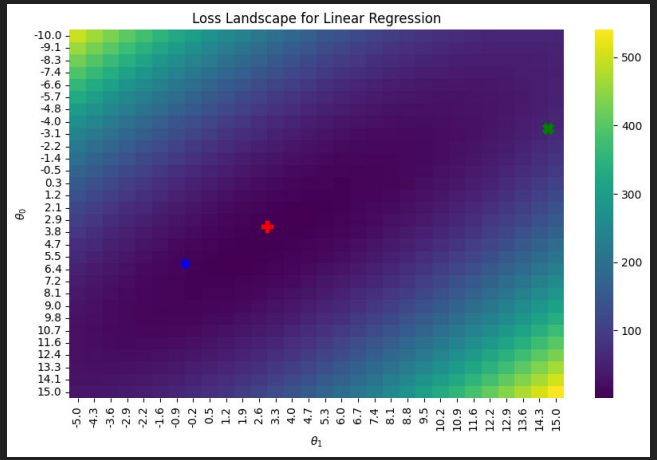


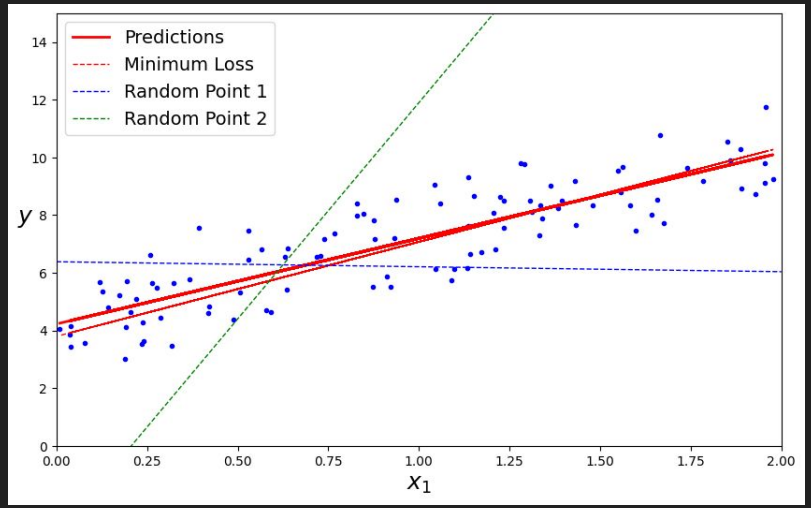
Demo: Let’s draw the loss landscape

- Notebook demo
- Let’s draw the loss landscape: simulation of many a and b and measure the loss plot it for different values

The loss landscape

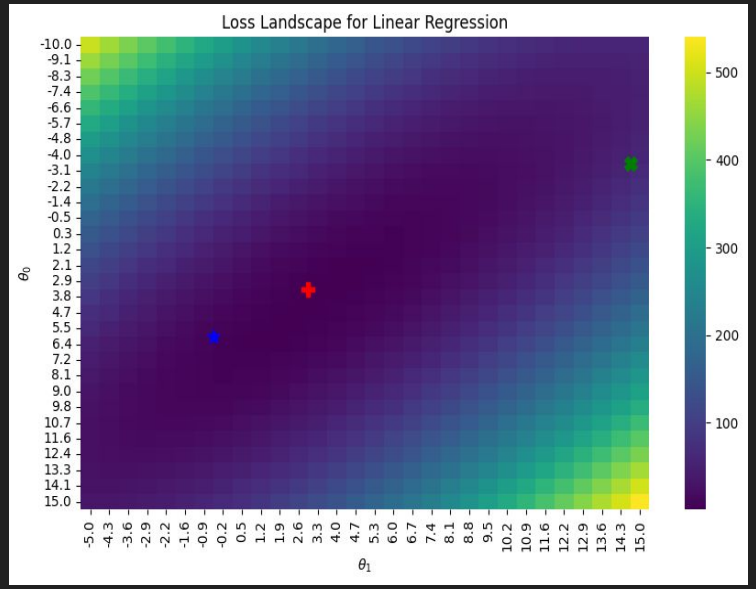
- $Y = \Theta_0 x + \Theta_1$

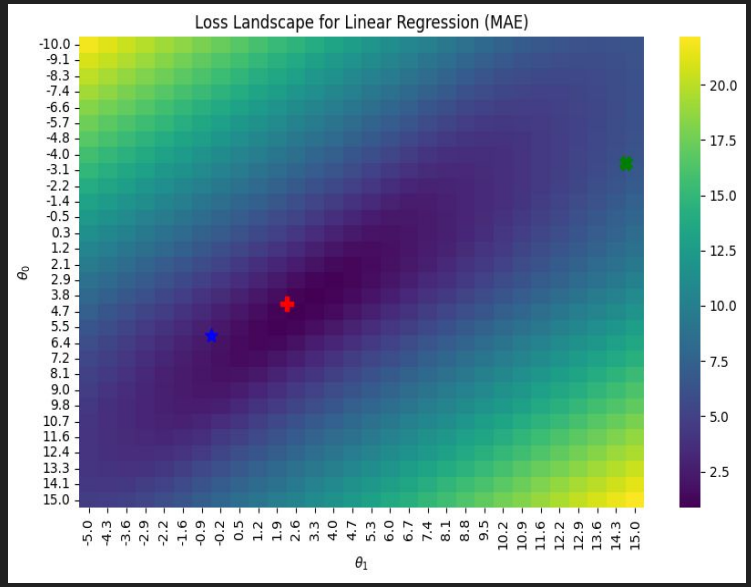




The loss landscape

- $Y = \Theta_0 x + \Theta_1$





What are solvers?

Algorithms that can take an equation and change the parameters so that the equation minimizes an error

Math of finding the correct path in the landscape

- Get some intuition about the math behind navigating a high dim landscape
- A ball rolls using gravity, can we find a way to know which way is down?
- The Derivative
- Intuition remains when it becomes high dimensional even if the math is different
- With higher dimensions, the landscape flattens

Gradient descent

The blind man on a foggy mountain of loss landscape, how to find the right path to navigate efficiently a loss function.

Go through Gradient Descent

- The derivative and its sign
- Step (jumping distance)
    - Example with rough and smooth landscape

Demo of Gradient Descent

- Notebook break to test gradient descent step by step

Deep Learning

From carbon to silicon, 60 years of progress in modeling brains in a computer.

Neurons

- Biology insight
- Historical insight ?
- Explainer on neurons as a function
- Activation function

More than one Neuron

- Non Linearity emergence
- Different architecture
- Deep Learning as a DAG (directed acyclic graph)

Back Propagation

- Remember Gradient Descent
- Back Prop is a way to compute the gradient across all neurons in one pass
- It is very efficient to compute
- Forget the math, the concept is just to find the gradient for each neuron

The curse of High Dimensionality

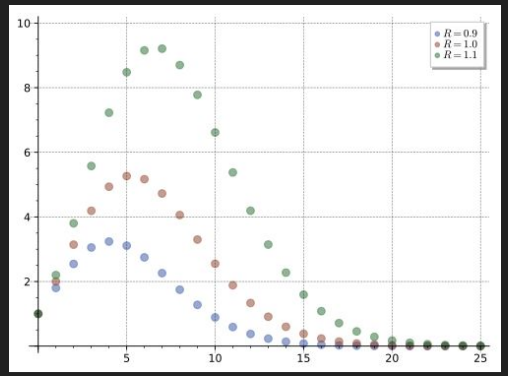
Distance is meaningless

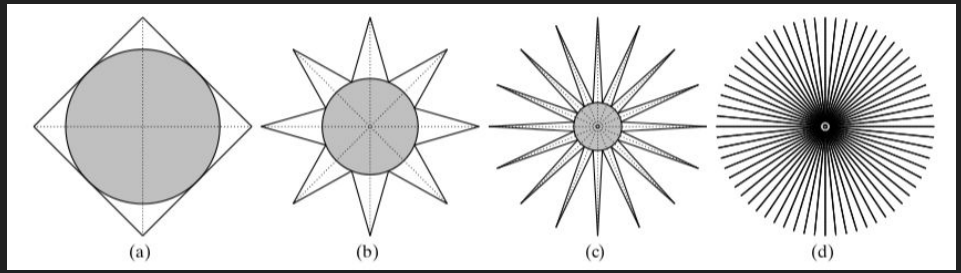
High Dimentions

[O]ur intuitions, which come from a three-dimensional world, often do not apply in high-dimensional ones. In high dimensions, most of the mass of a multivariate Gaussian distribution is not near the mean, but in an increasingly distant “shell” around it; and most of the volume of a high-dimensional orange is in the skin, not the pulp. If a constant number of examples is distributed uniformly in a high-dimensional hypercube, beyond some dimensionality most examples are closer to a face of the hypercube than to their nearest neighbor. And if we approximate a hypersphere by inscribing it in a hypercube, in high dimensions almost all the volume of the hypercube is outside the hypersphere. This is bad news for machine learning, where shapes of one type are often approximated by shapes of another.

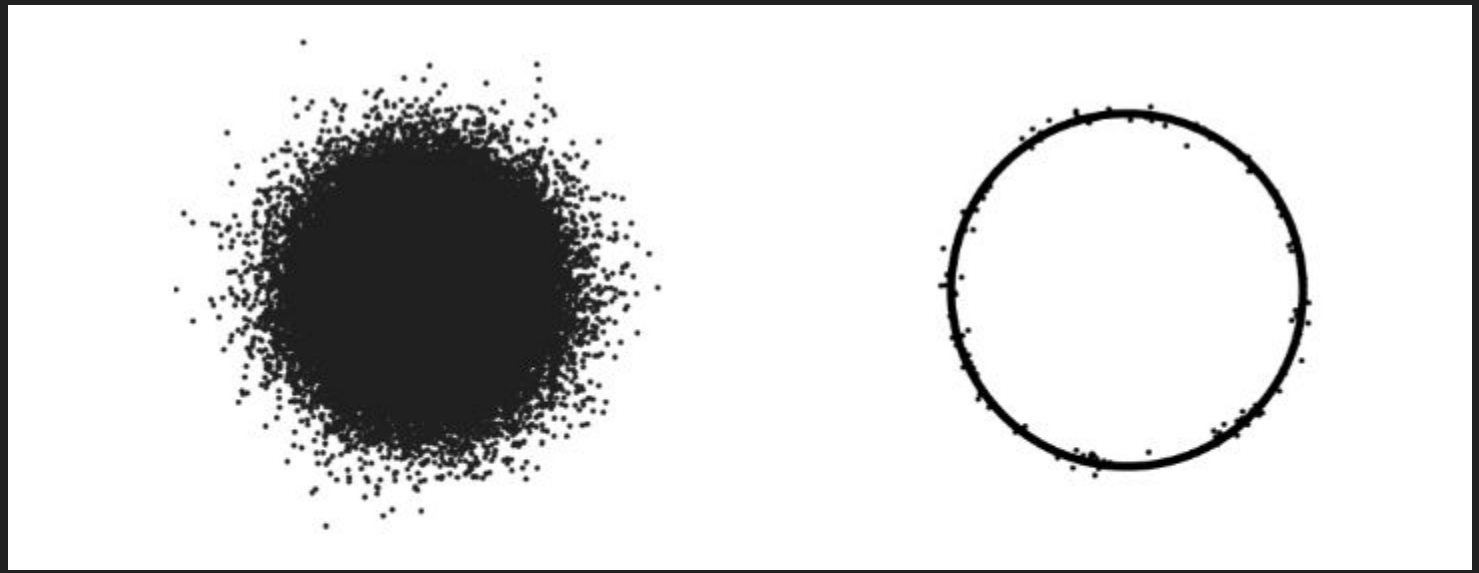
Pedro Domingos

Volume of a hyperSphere

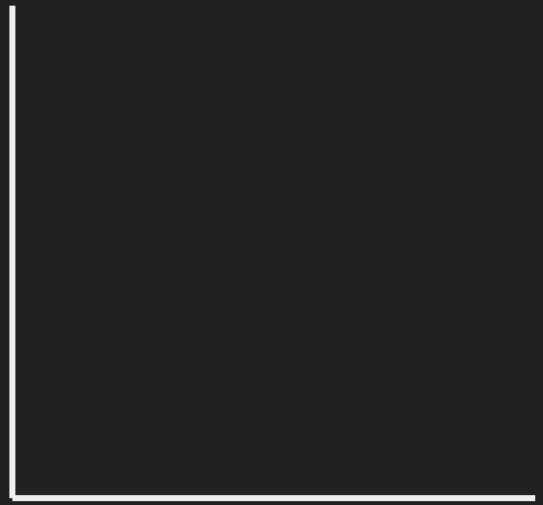


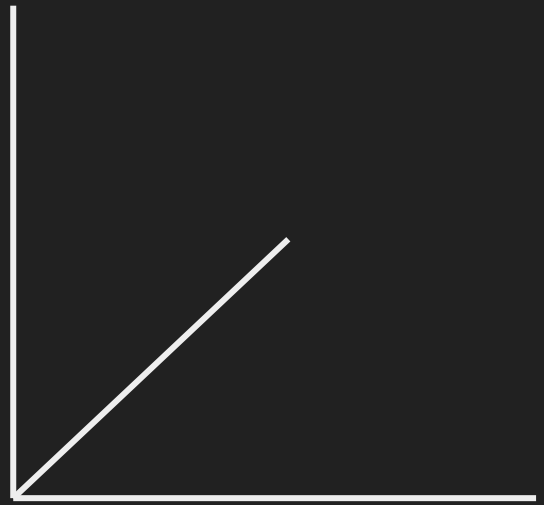


Gaussian in High Dimension

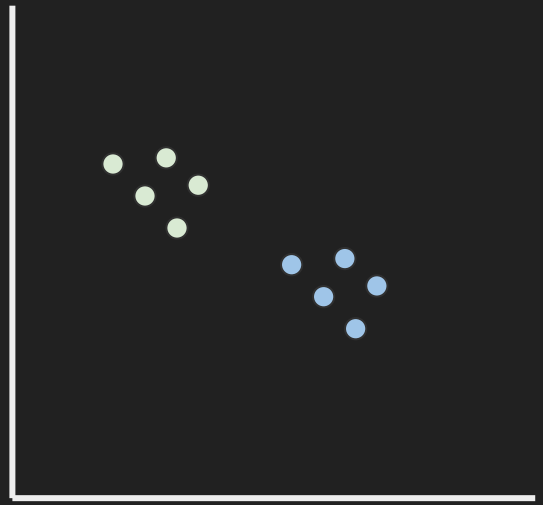


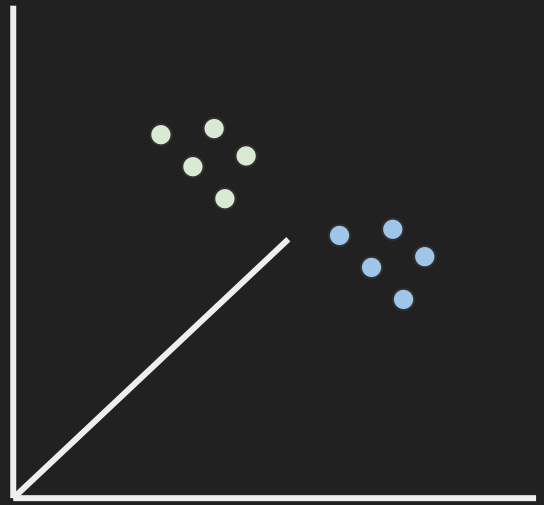
Euclidean space



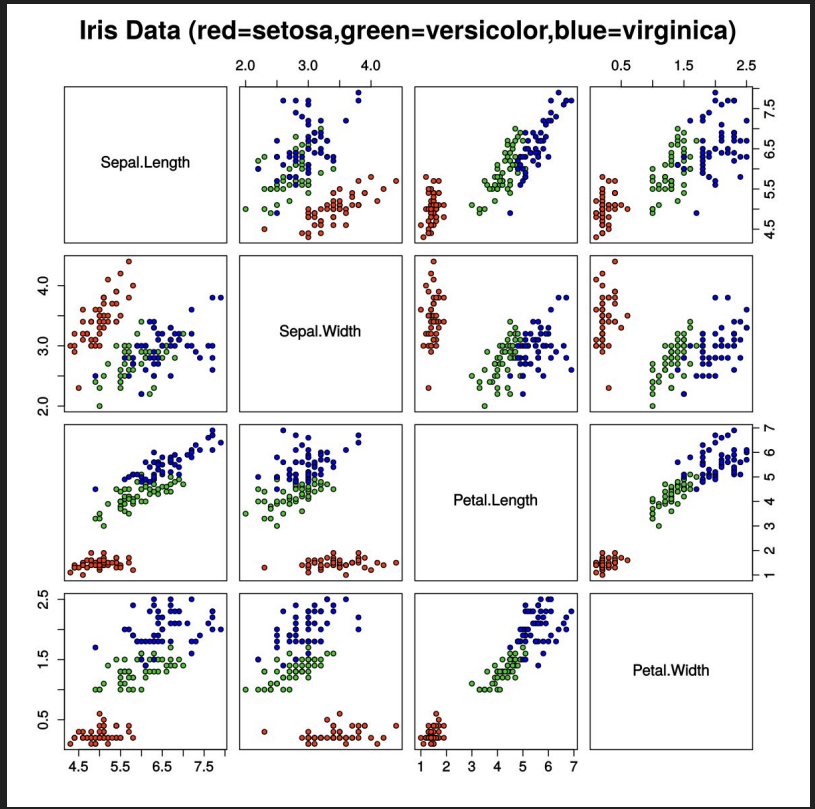


Euclidean space

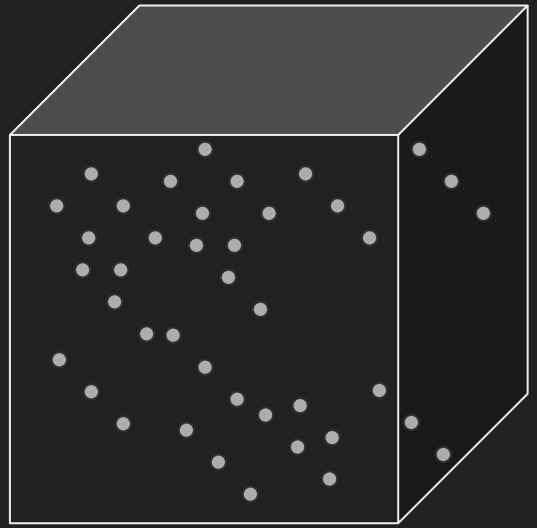




Neighbours are related?

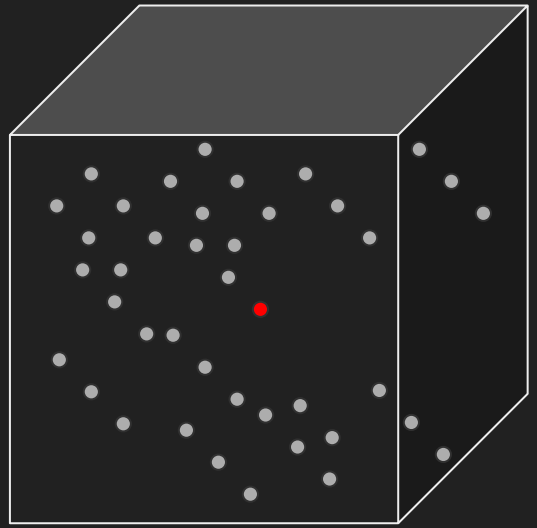


Finding Neighbours



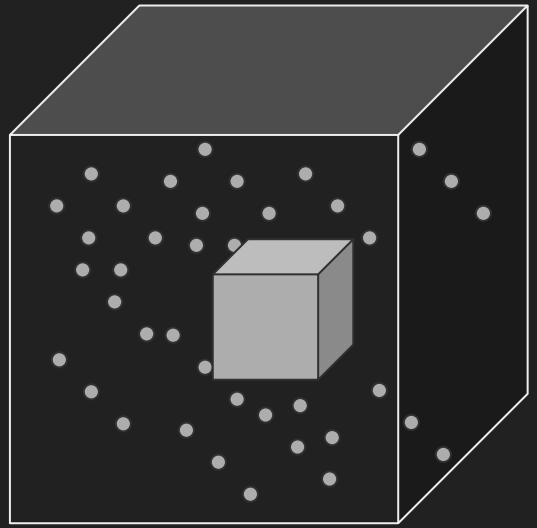
Let’s assume a unit cube, and uniformly distributed points in the cube

Finding Neighbours



I want to find the closest points of the red point

Finding Neighbours

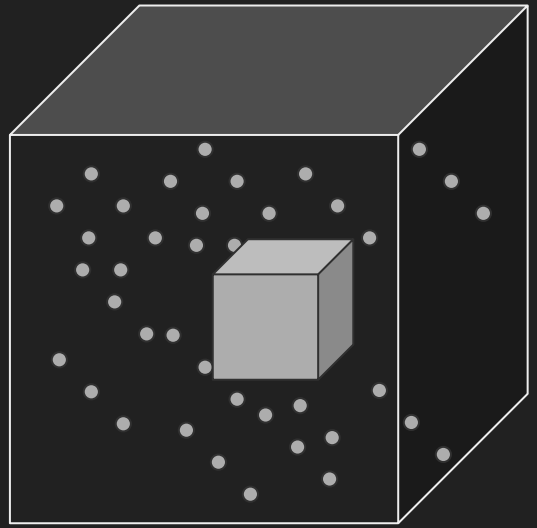


I can draw a cube (or a sphere but the math is easier with a cube)

Finding Neighbours

$$l^d = \frac{k}{n}$$

$$l^3 = \frac{k}{n}$$

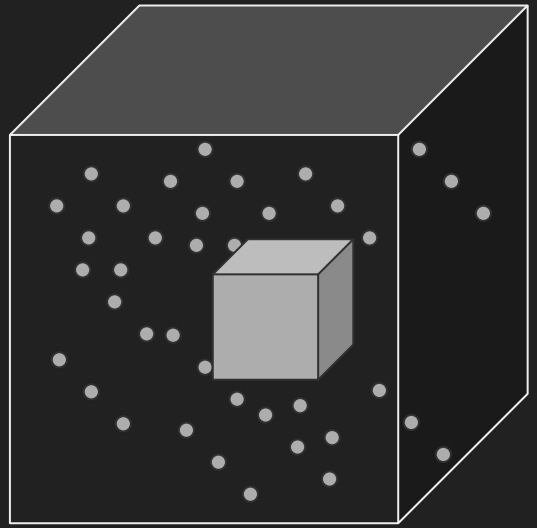


Everything in my cube is a neighbour

Finding Neighbours

$$l^d = \frac{k}{n}$$

$$l^3 = \frac{k}{n}$$



$$l = (\frac{k}{n})^{\frac{1}{d}}$$

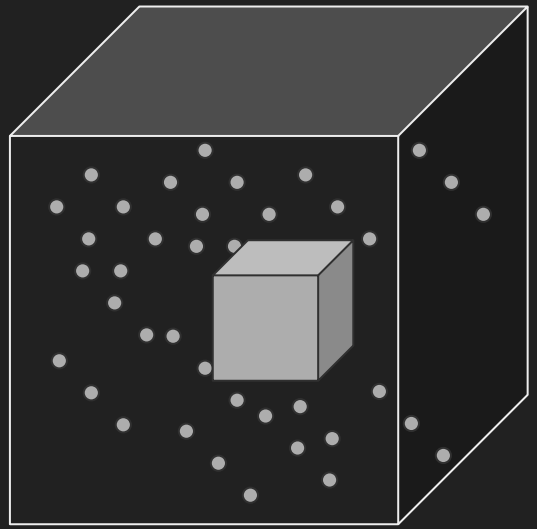
Everything in my cube is a neighbour

Finding Neighbours

$$l = (\frac{k}{n})^{\frac{1}{d}}$$

$$l^d = \frac{k}{n}$$

$$l^3 = \frac{k}{n}$$



d	l
2	0.1
3	0.21
10	0.63
100	0.955
1000	0.995

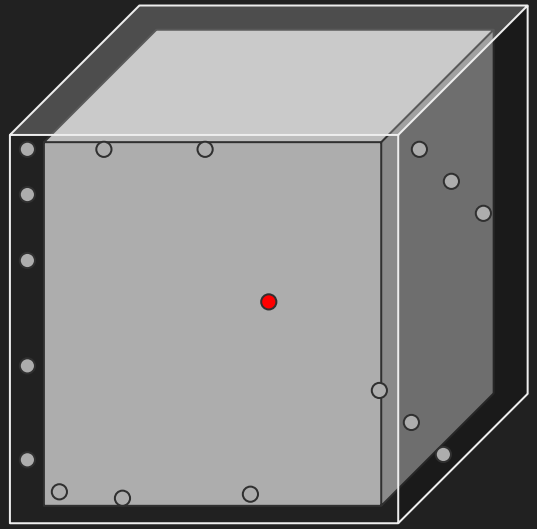
For k=1 & n=100, so a cube that has one element in it among 100 elements

Finding Neighbours

$$l = (\frac{k}{n})^{\frac{1}{d}}$$

$$l^d = \frac{k}{n}$$

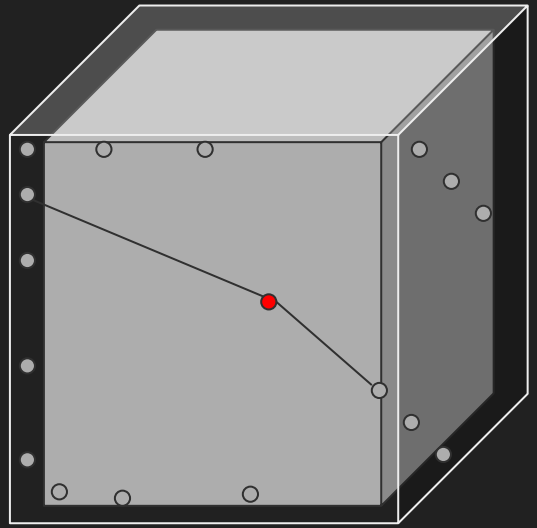
$$l^3 = \frac{k}{n}$$



d	l
2	0.1
3	0.21
10	0.63
100	0.955
1000	0.995

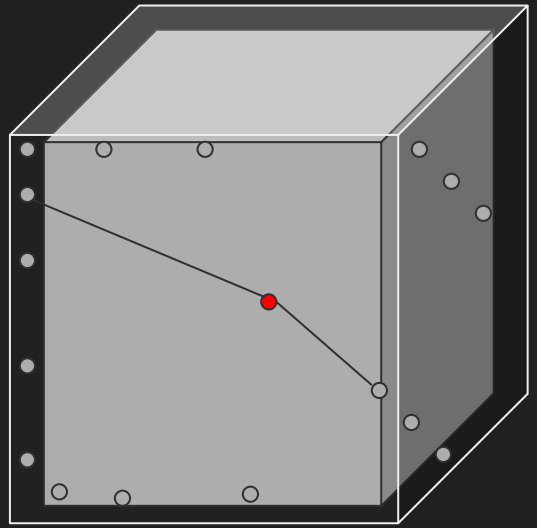
For k=1 & n=100, so a cube that has one element in it among 100 elements

Finding Neighbours



The distance to any other point becomes meaningless

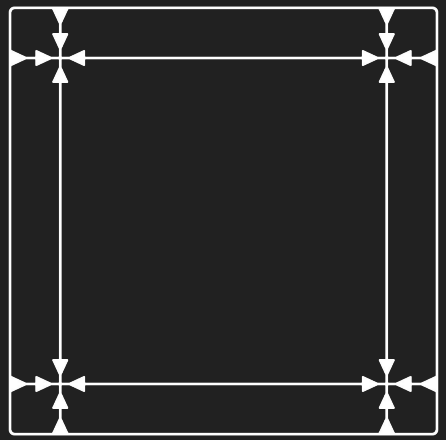
Finding Neighbours



The concept of “Interior” loses meaning, everything is far away

What is the “Border”

$$p = 1 - 2\epsilon$$

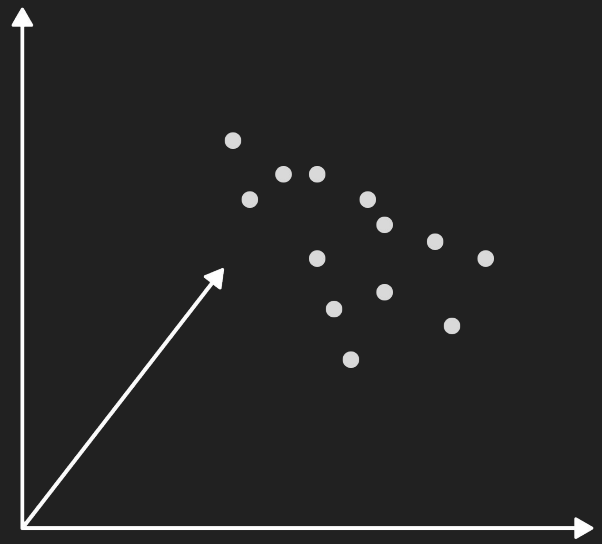


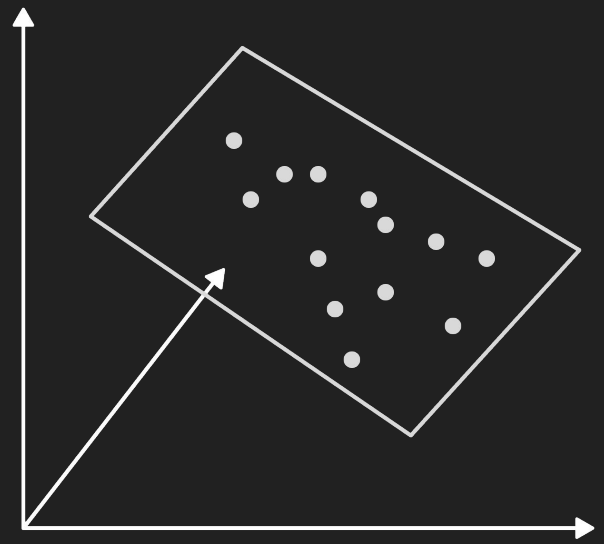
$$p = (1 - 2\epsilon)^2$$

What is the “Border”

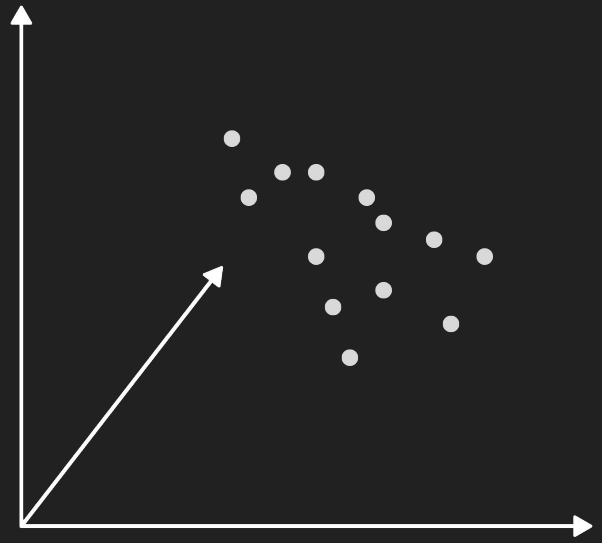
$$p_{interior} = (1 - 2\epsilon)^d \to 0$$

True Dimensionality

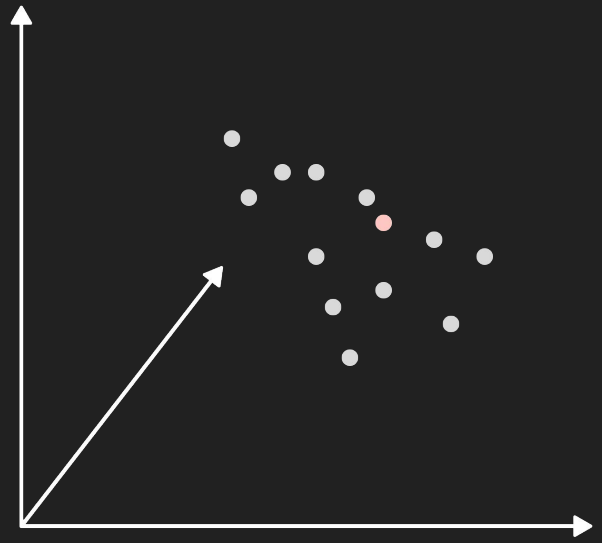




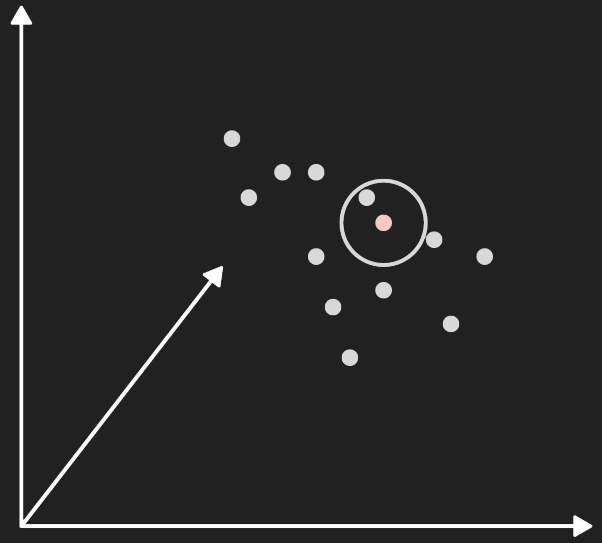
Estimating Intrinsic Dimensionality



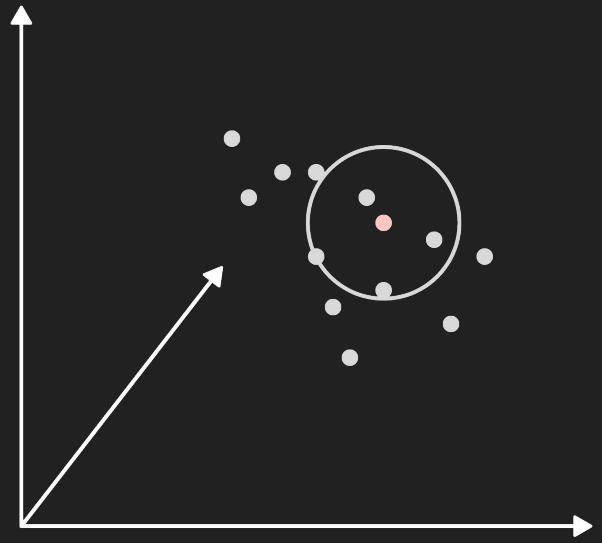
Estimating Intrinsic Dimensionality



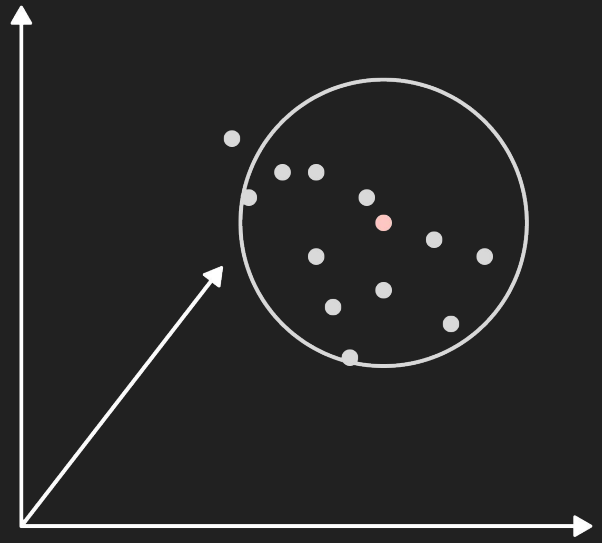
Estimating Intrinsic Dimensionality



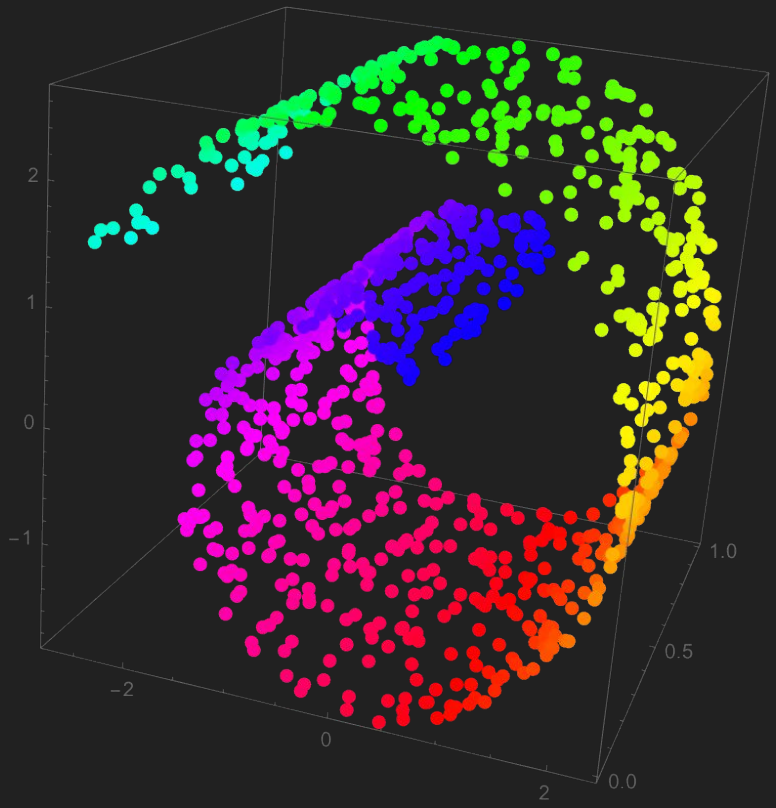
Estimating Intrinsic Dimensionality



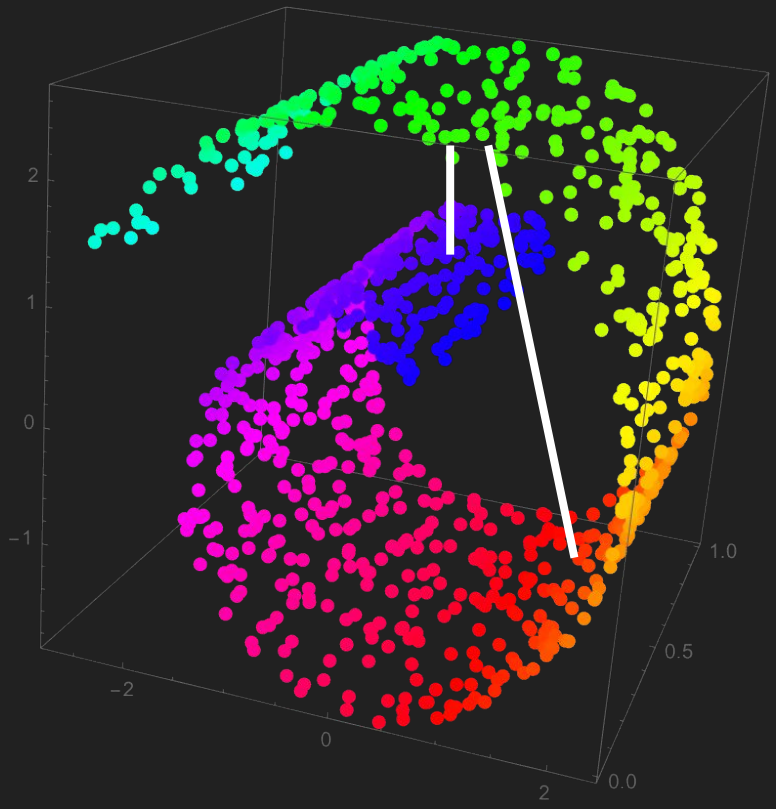
Estimating Intrinsic Dimensionality



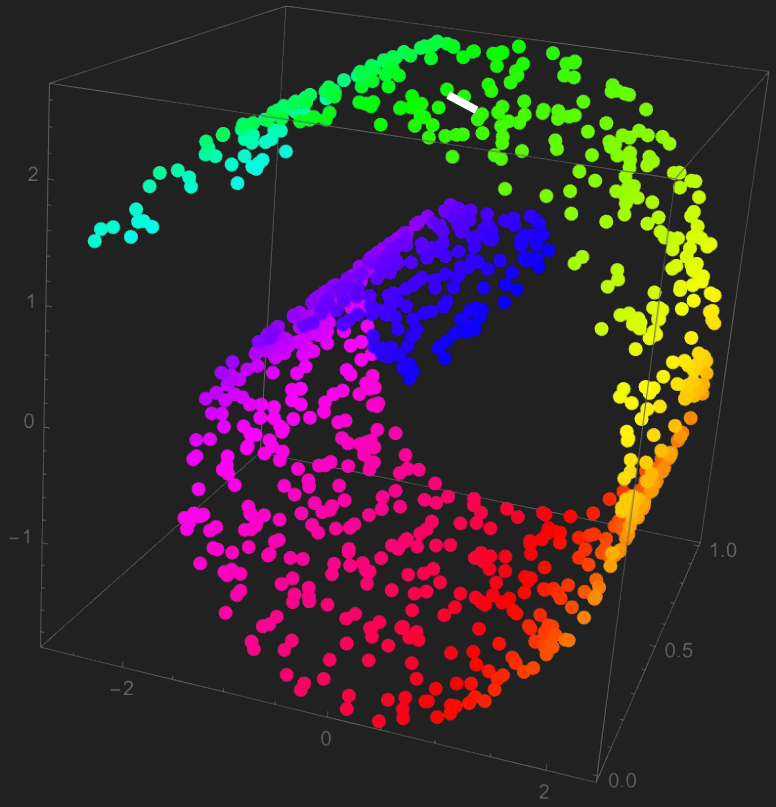
Manifolds



Manifolds



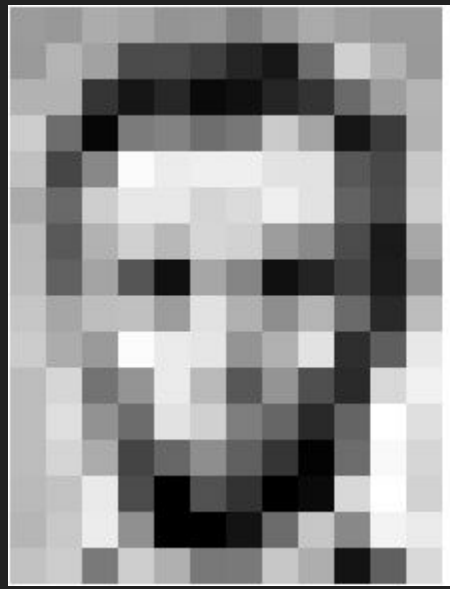
Manifolds



Why is this important

- Data with no intrinsic lower dimensionality is just a random sample
- This means that you probably should no analyze it
- Lower intrinsic dimensionality represents features, those are what one is interested in

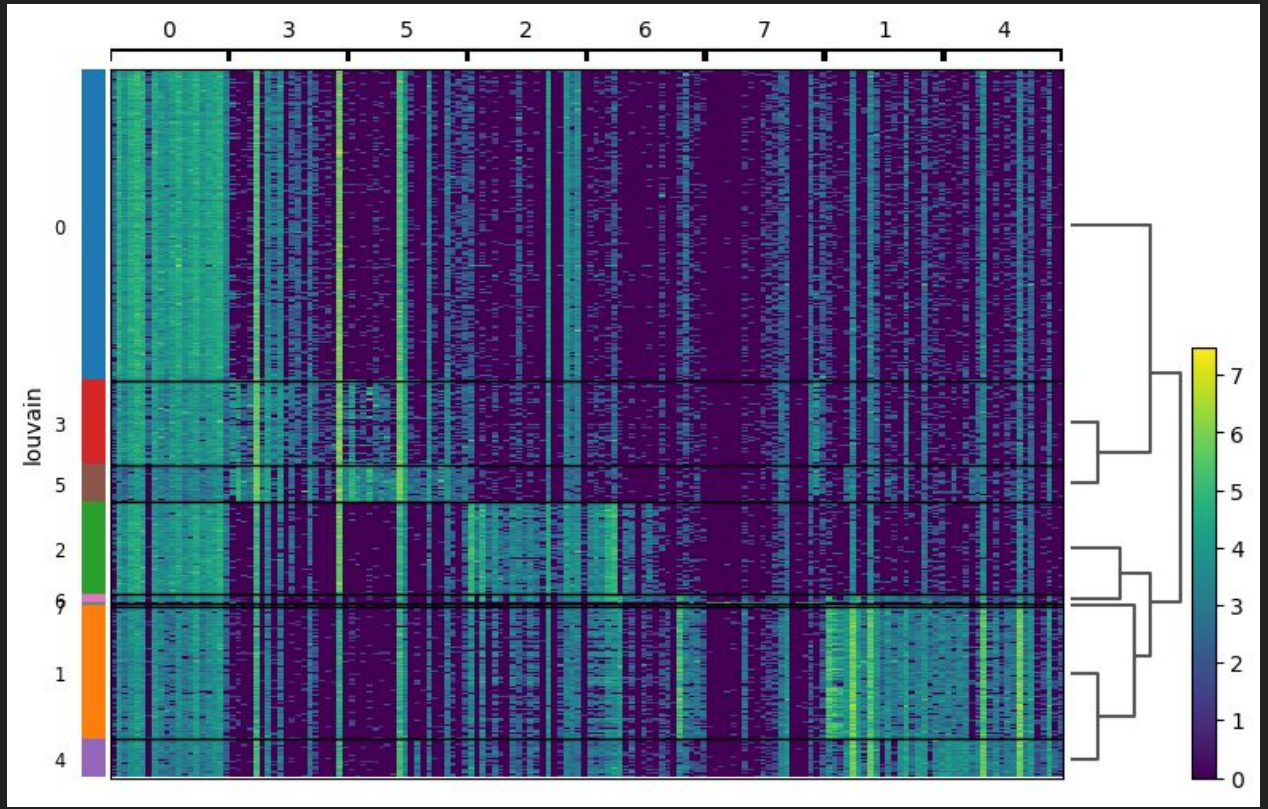
Example: Images



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

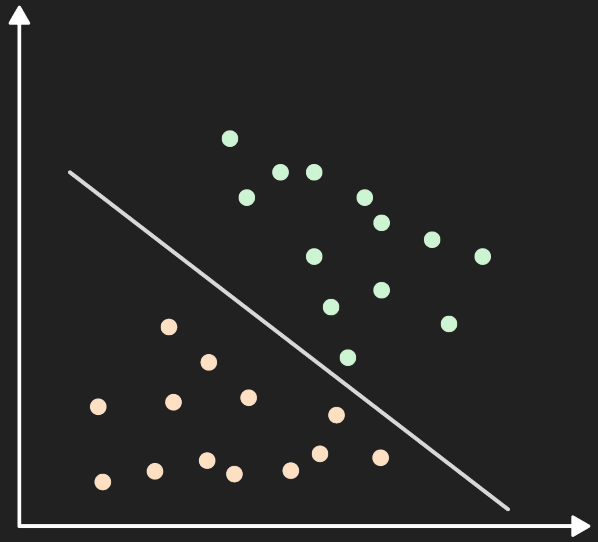
Example: Gene expression data

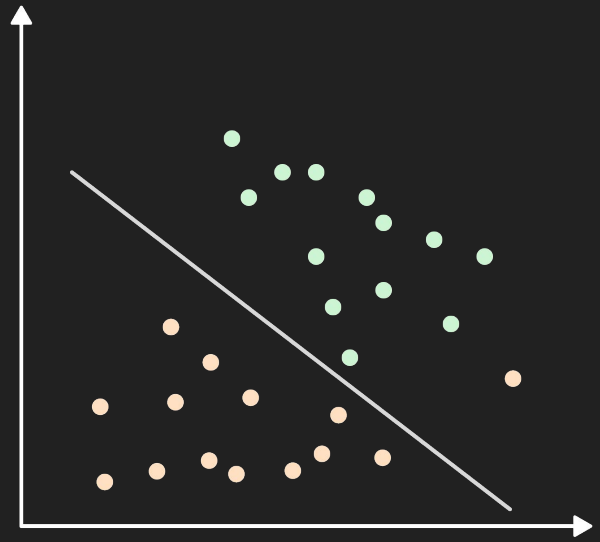


Dimensionality Reduction as a first step

- A lot of algorithm, and intuitions do not work in high dimensionality
- Dimensionality reductions like PCA, or UMAP will reduce the dataset to their intrinsic dimensionalities
- Applying a classifier to this reduced space often works better

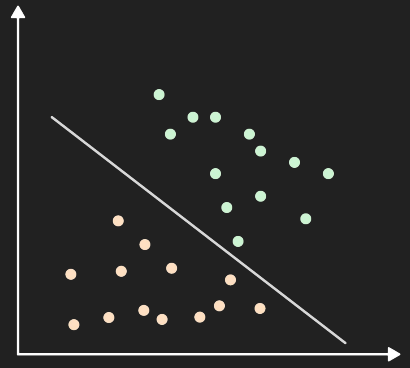
The blessing of high dimensionality





The blessing of high dimensionality

- In high dimensionality, every point is very far from each other point
- You can prove that there always exist a hyperplane that will separate the data

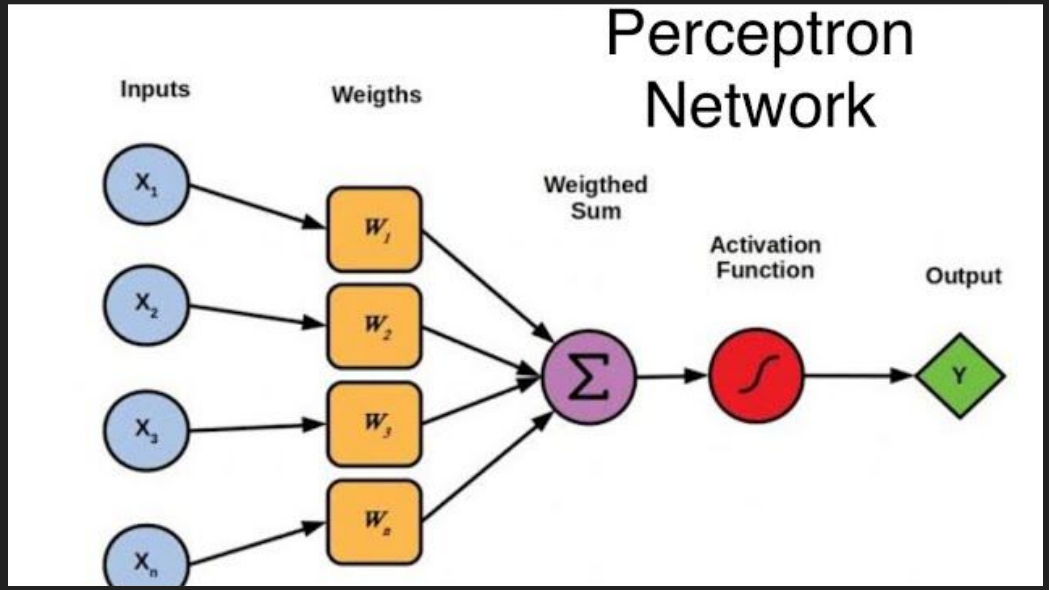


$$plane = w^T x = 0$$

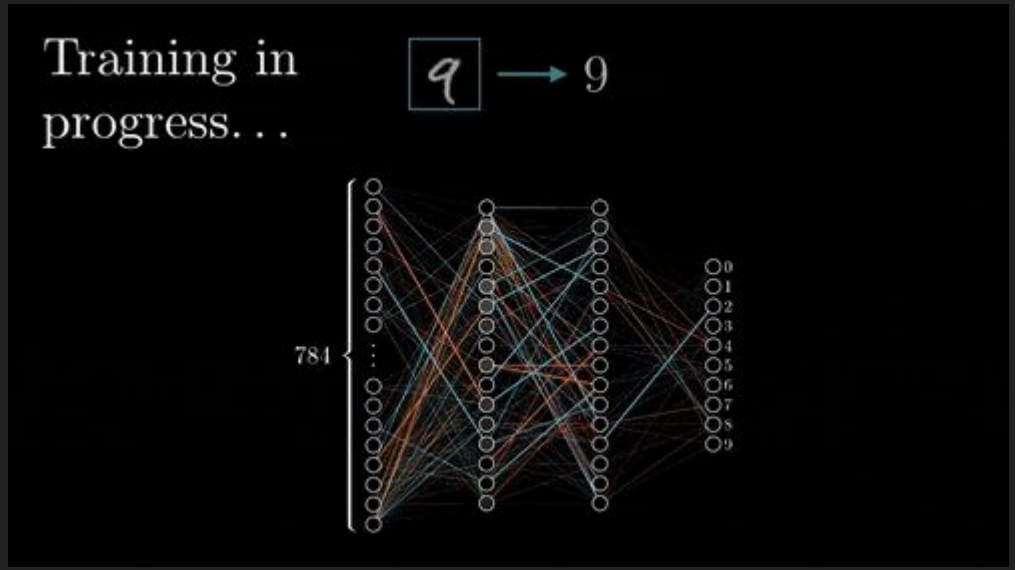
The perceptron (1957)

- Given a high dimensional data, I want to classify it into two binary classes
  - Cat or Dog
- As we saw before, an image is a high dimensional data
- How can we exploit this into an algorithm?

The perceptron (1957)



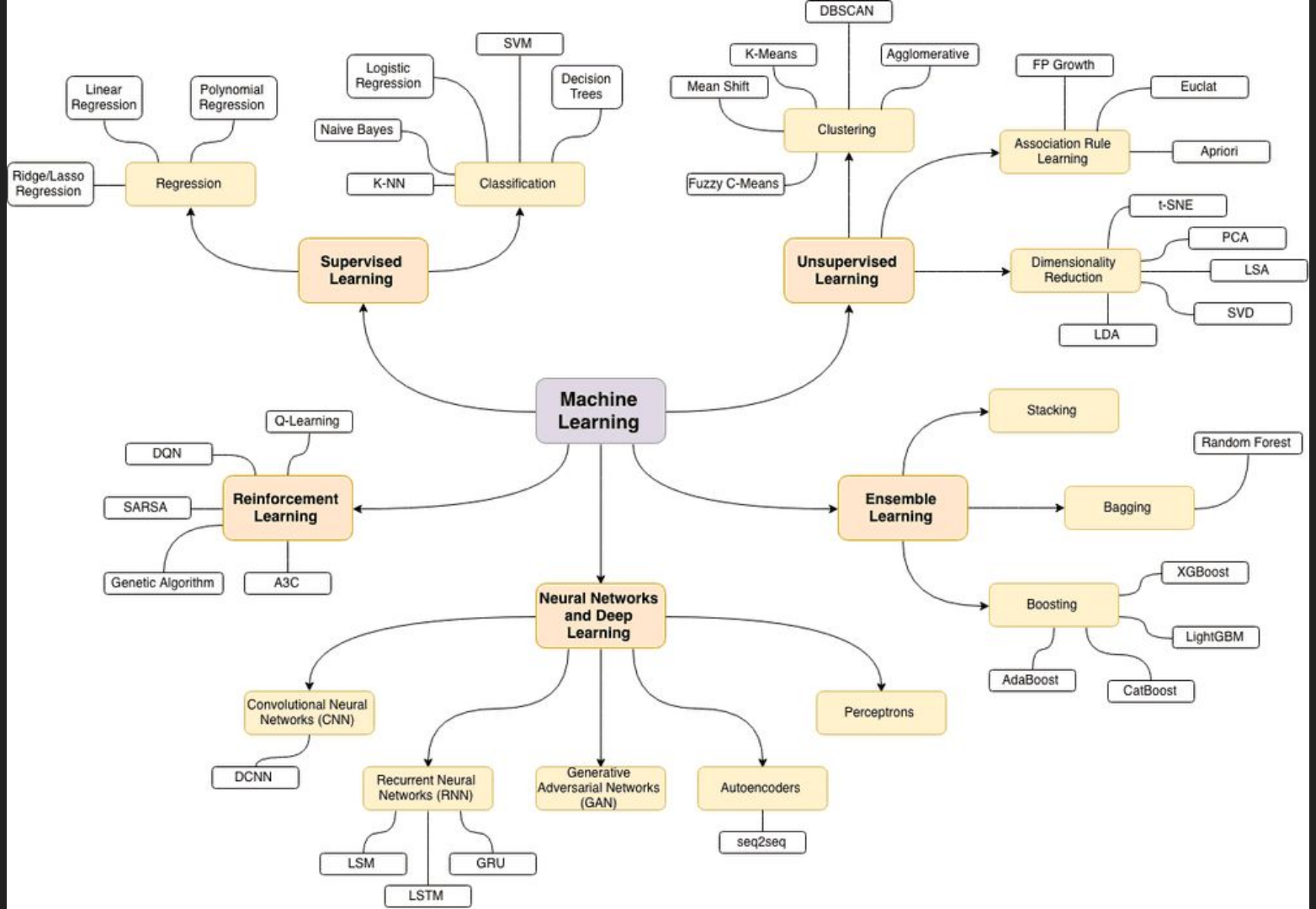
The perceptron (1957)



https://perceptrondemo.com/

Pre-Made

Think before you use a model


Machine Learning
Supervised Learning
Regression
Linear Regression
Polynomial Regression
Ridge/Lasso Regression
Classification
Logistic Regression
SVM
Decision Trees
Naive Bayes
K-NN
Unsupervised Learning
Clustering
K-Means
DBSCAN
Agglomerative
Mean Shift
Fuzzy C-Means
Association Rule Learning
FP Growth
Euclat
Apriori
Dimensionality Reduction
t-SNE
PCA
LSA
SVD
LDA
Reinforcement Learning
Q-Learning
DQN
SARSA
Genetic Algorithm
A3C
Ensemble Learning
Stacking
Bagging
Random Forest
Boosting
XGBoost
LightGBM
AdaBoost
CatBoost
Neural Networks and Deep Learning
Convolutional Neural Networks (CNN)
DCNN
Recurrent Neural Networks (RNN)
LSM
GRU
LSTM
Generative Adversarial Networks (GAN)
Autoencoders
seq2seq
Perceptrons

Your data's shape should inform your choice