Machine Learning: Overview

- Basic Idea: Design algorithms that can learn how to perform tasks based on some provided data.

  → Ex: Train an algorithm to decide whether an image is of

a cat

OR

a dog

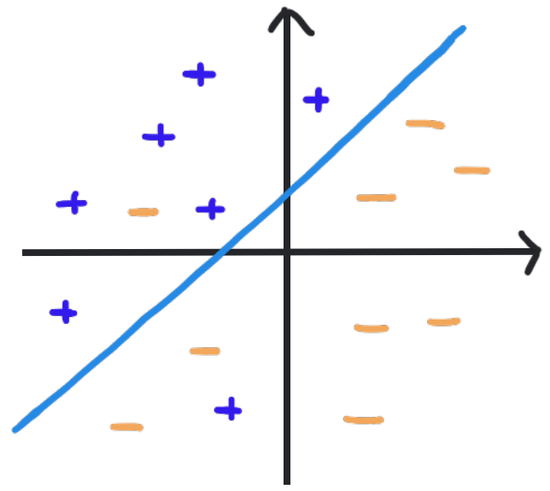
- While it might seem like a good fit for binary hypothesis testing, we do not have good probability models for cat and dog images.
- Instead, we can use a dataset of example images to design a good decision rule.

- **Supervised Learning:** The dataset is labeled, meaning that for each example we have both the desired input and the desired output, which is sometimes called the label.
- **Unsupervised Learning:** The dataset is unlabeled, meaning that for each example, we only know the input.
- A dataset is a collection of examples of the form $(\underline{X}_i, Y_i)$ for $i = 1, \ldots, n$. We can think of the vector $\underline{X}_i = \begin{bmatrix} X_{i,1} \\ \vdots \\ X_{i,d} \end{bmatrix}$ as an input whose entries are often called features. We can think of Y_i as the desired output, which is often called the label.
- These features are highly correlated so finding a good probability model involves estimating a high-dimensional function. This can require many, many examples, often more than we have available. This phenomenon is sometimes called the curse of dimensionality.

- Examples of Applications:

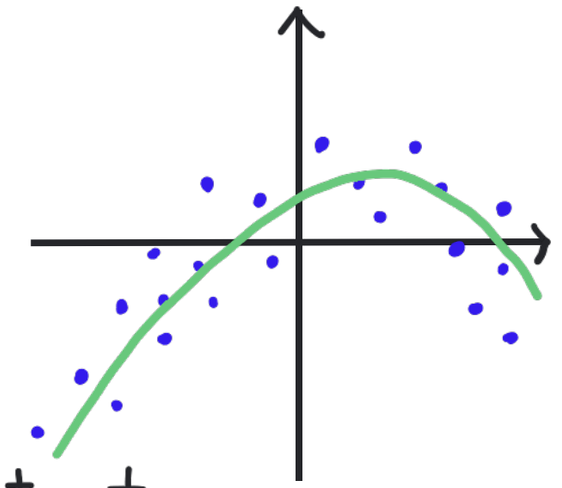
→ Classification: Place each example into a category specified by its label. Similar to detection, but without a probability model.

* Ex: Cat and Dog Images



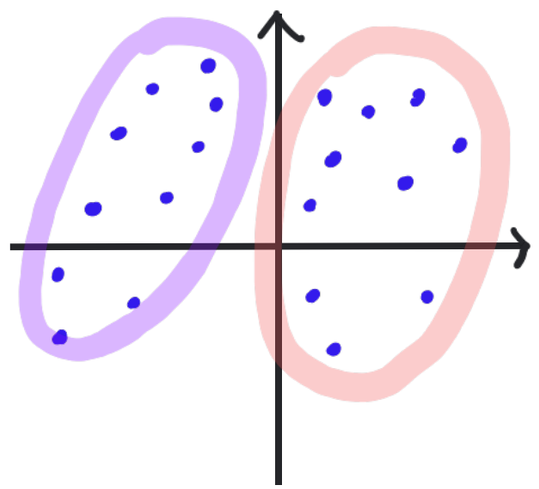
→ Regression: Use the input values to predict the output value and minimize a loss function (like mean-squared error). Similar to estimation, but without a probability model.

* Ex: Predict cholesterol levels from height, weight, etc.



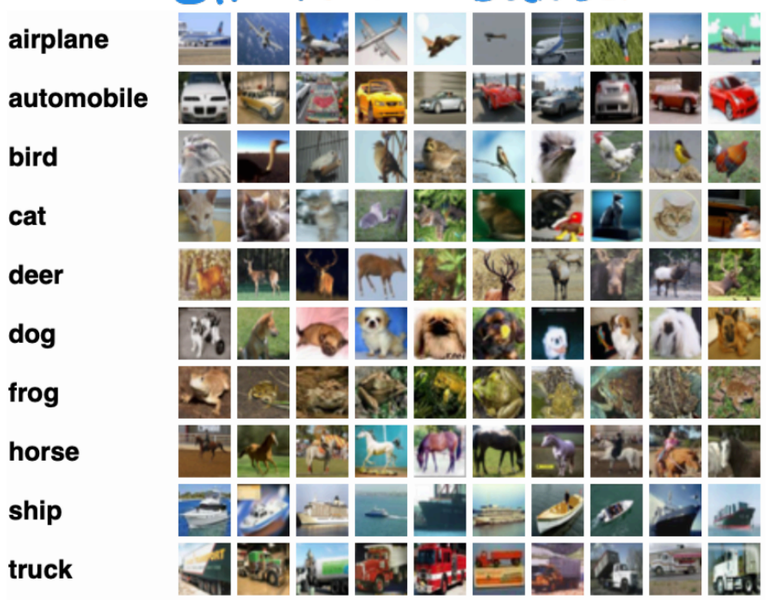
→ Clustering: Group examples into meaningful clusters without access to any labels.

* Ex: Sort online news articles into coherent topics such as sports, food, etc.



- As a first step, we need to gather and possibly label a large collection of examples to serve as our dataset.
  - → Important to decide which features to use and which to throw away.
  - → Each example should be in the same format.
  - → Labeling examples can be very time-consuming.
- Many interesting datasets are freely available via repositories and competitions. Here are a few examples:

CIFAR 10 Dataset



Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009

Kaggle

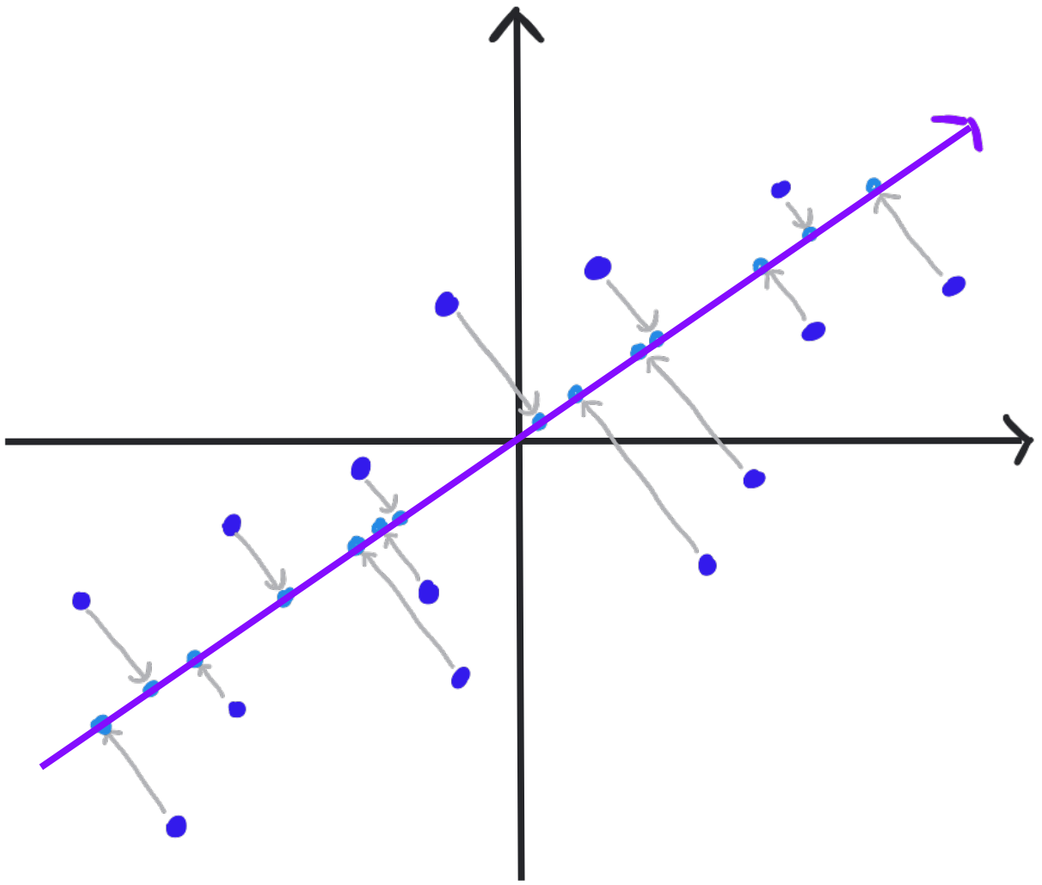
- → Hosts many interesting competitions
- → Large collection of datasets from previous competitions
- → Tutorials and other resources to get you started

UCI Machine Learning Repository

Google Dataset Search Engine

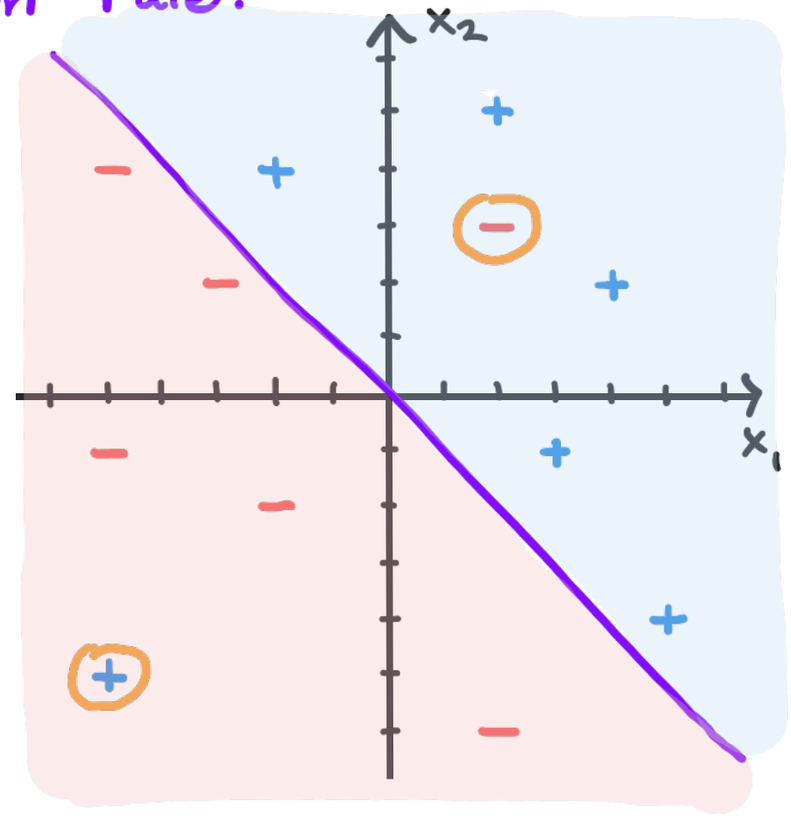
github Awesome Public Datasets

- Dimensionality Reduction:
  - → The curse of dimensionality means that it can become more challenging to estimate probability models, select good decision rules, etc., as the number of features increases.
  - → One way to overcome this barrier is to transform our data from its original high-dimensional space to a low-dimensional space that preserves the important relationships between examples.



- → Initially, data lives in a high-dimensional space.
- → Find a good low-dimensional space.
- → Project each example onto the low-dimensional space.
- → Train algorithm on this space.

- As a case study, consider a binary classification problem.
- Our goal is to select a good decision rule.
- Consider a dataset with two features, $\underline{X}_i = \begin{bmatrix} X_{i,1} \\ X_{i,2} \end{bmatrix}$, and +/− labels, $Y_i \in \{+1, -1\}$



- Here is one possible decision rule.
- What is a good performance measure?
  → Consider the error rate, which is the fraction of misclassified points. (Sometimes this is reported as a percentage.)

Total # Examples = 12

Errors = 2

Error Rate = $\frac{2}{12} \approx 16.7\%$

- One approach to designing a decision rule is to estimate the conditional probability models $f_{\underline{X}|Y}(\underline{x}|y=+1)$ and $f_{\underline{X}|Y}(\underline{x}|y=-1)$ and then apply the ML Rule.

- How can we construct probability models from data?

→ Parametric: Assume the data comes from a certain family of random variables and estimate its parameters. Converges faster but might not be the best fit.

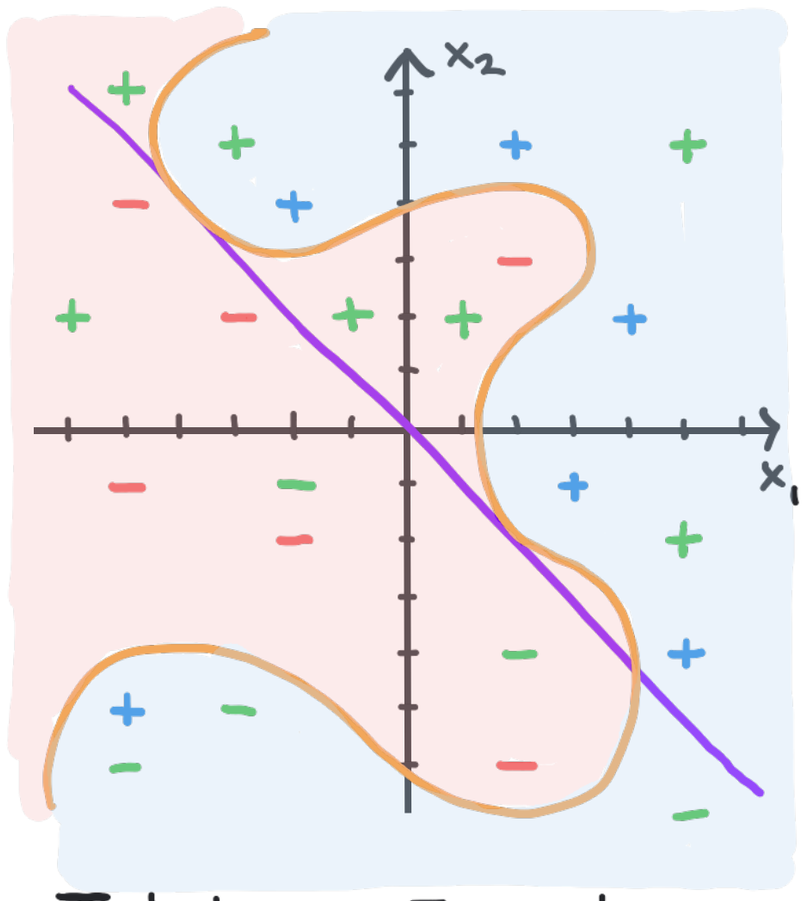
* Ex: Assume distributions are Gaussian. Estimate means and variances.

→ Non-Parametric: Make no assumptions about the distribution family. Converges more slowly but to a good fit.

* Ex: Kernel Density Estimation. Place a "kernel" at each point

$$\hat{f}_x(x) = \frac{1}{n} \sum_{i=1}^{n} K(x - x_i)$$

- Can we improve upon the error rate of our simple decision rule?
- Yes! In fact, we can always get the error rate all the way down to 0 by either memorizing the dataset or design a more complex decision rule.
- However, this may lead to overfitting the data and lower performance in the real world.
  - → For the new data points, the simple decision rule is clearly better.
  - → How can we avoid overfitting using only the dataset we have?



Total # Examples = 12

Errors = 2

Error Rate = $\frac{2}{12} \approx 16.7\%$

Errors = 0

Error Rate = $\frac{0}{12} = 0\%$

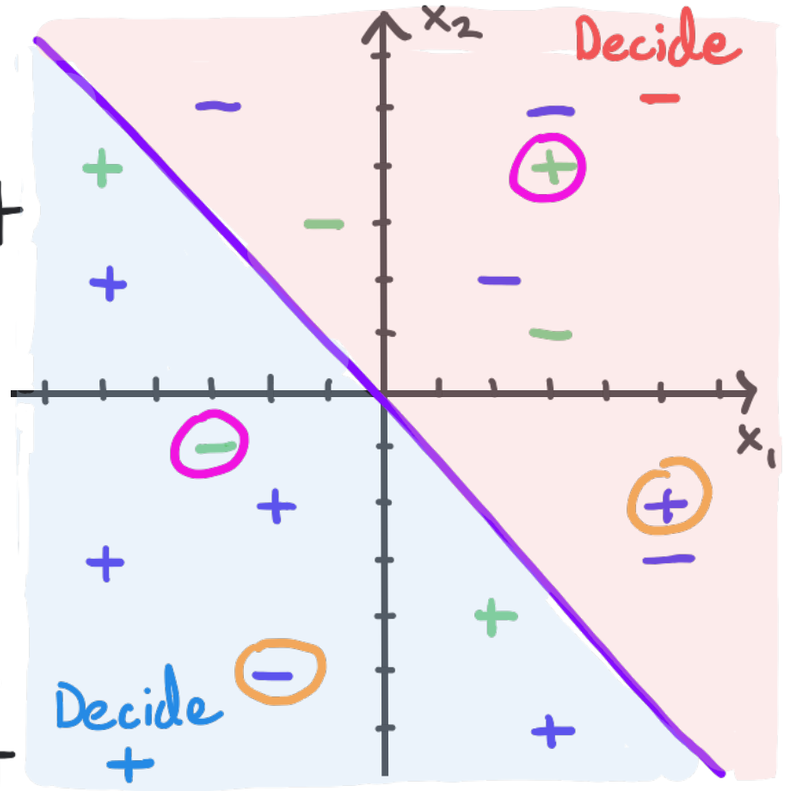
- The most common approach to avoid overfitting is to split our data set into training data and test data.

  → Training Data: We use this to select our decision rule and optimize all of its parameters.

  → Test Data: We use this to estimate the error rate of our decision rule. This enables us to compare different approaches (but not optimize parameters!)

→ The training error rate is the error rate calculated only on training data.

→ The test error rate is the error rate calculated only on test data.



Training Examples = 10

Training Errors = 2

Training Error Rate = $\frac{2}{10}$

Test Examples = 6

Test Errors = 2

Test Error Rate = $\frac{2}{6}$