Introduction to Machine Learning

Machine Learning is…

- Machine learning, a branch of artificial intelligence, concerns the construction and study of systems that can learn from data.
- The phrase "machine learning" describes a wide diversity of algorithms and techniques.

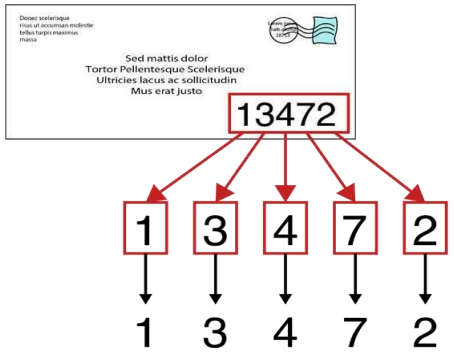
A big, expanding collection of algorithms and principles that analyze vast quantities of training data in order to extract meaning from it.

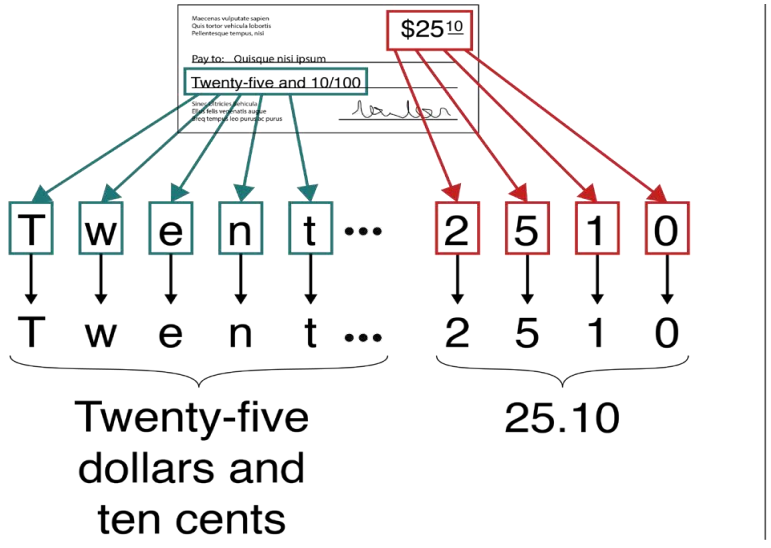
- The phrase machine learning describes a growing body of techniques that all have one goal: discover meaningful information from data.

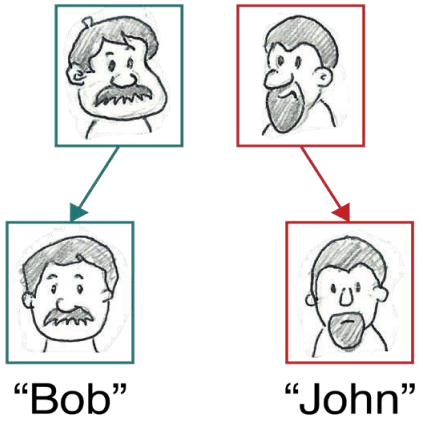
Data

- Data can be
  - raw numbers (like stock prices on successive days, the mass of different planets, the heights of people visiting a county fair).
  - sounds (the words someone speaks into their cell phone),
  - pictures (photographs of flowers or cats),
  - words (the text of a newspaper article or a novel),
  - or anything else that we want to investigate.
- "Meaningful information" is whatever we can extract from the data that will be useful to us in some way.
- We decide what's meaningful to us, and then we design an algorithm to find as much of it as possible from our data.

Example applications that use machine learning to extract meaning from data







- Left: Getting a zip code from an envelope.
- Middle: Reading numbers &letters on a cheque.
- Right: Recognizing faces from photos.

Expert systems can also find meaning from data

- Expert systems was an early approach to finding the meaning that's hiding inside of data.
- Idea:
    - Study what human experts know and automate that.
    - Make a computer mimic the human experts it was based on.
- Create a rule-based system: a large number of rules for the computer to imitate human experts.
- Example: Recognize zip codes. 7's have a horizontal line at the top, and a diagonal line that starts at the right edge of the horizontal line and moves left and down. Some people put a bar through the middle of their 7's. So now we add another rule for that special case.

But handcrafting rules is a tough job!

- This process of hand-crafting the rules to understand data is called feature engineering
- This term is also used to describe when we use the computer to find these features for us.
- It’s easy to overlook one rule, or even lots of them. It's a tough job !

How does ML compare with Expert systems?

- **Expert Systems:** Difficult to manually find right set of rules, & make sure they work properly across a wide variety of data. These difficulties have doomed expert systems.
- **ML Systems:** Beauty is they learn a dataset's characteristics automatically.
- Don't have to tell an algorithm how to recognize a cat or dog, because system figures that out for itself.
- **Flip side of ML:** To do its job well, ML system often needs a lot of data. *Enormous amounts of data.*

Common threads in ML applications

Sheer volume of work involved, and its painstaking detail.

- **We have millions of data to examine, and we want to extract some meaning from every one of them.**
- **Why can't humans do it ?**
  - **Humans get tired, bored, and distracted,**
- **What about Computers?**
  - **computers just plow on steadily and reliably.**

ML has ability to extract meaningful information quickly, so are used in many fields.

Why recent explosion in ML ?

- Why has machine learning has exploded in popularity and applications in the last few years?
- Couple of big reasons:

A. Flood of data: provided by the Internet has let these tools extract a lot of meaning from a lot of data.

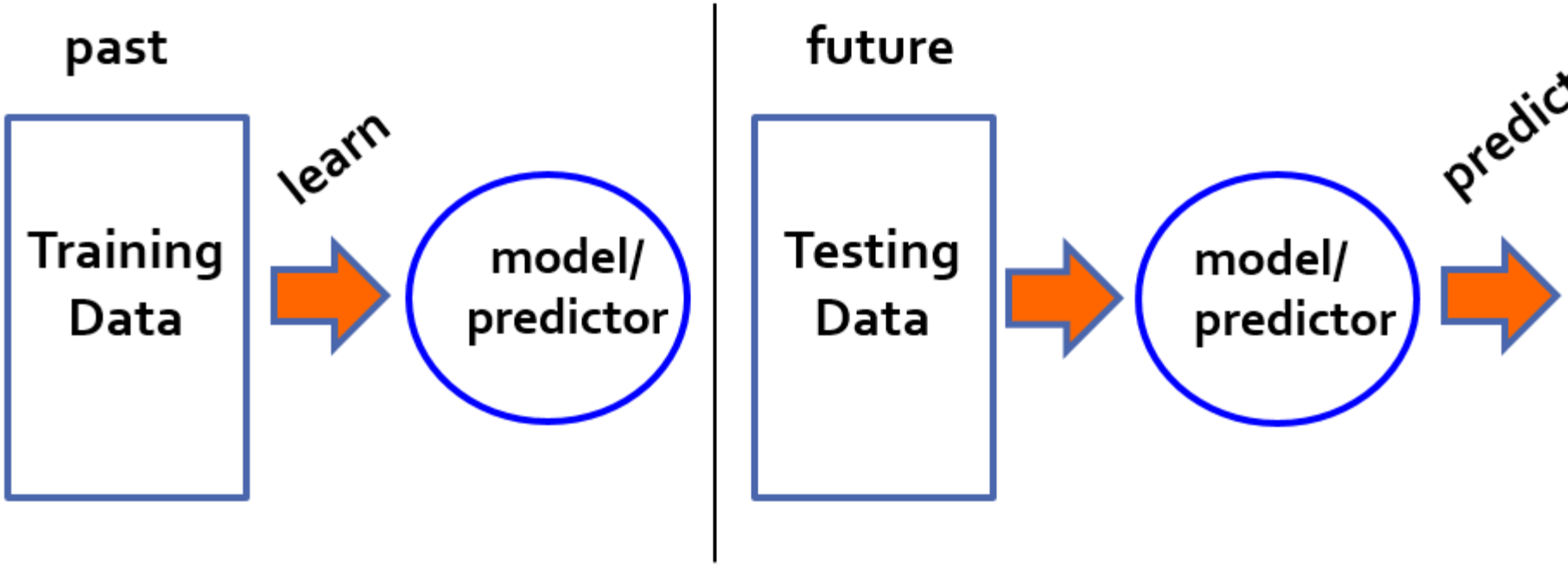
Example: Online companies make use of every interaction with every customer to accumulate more data. Then they use it as input to ML algorithms, getting more information about customers.

B. Increased Computing power - GPUs

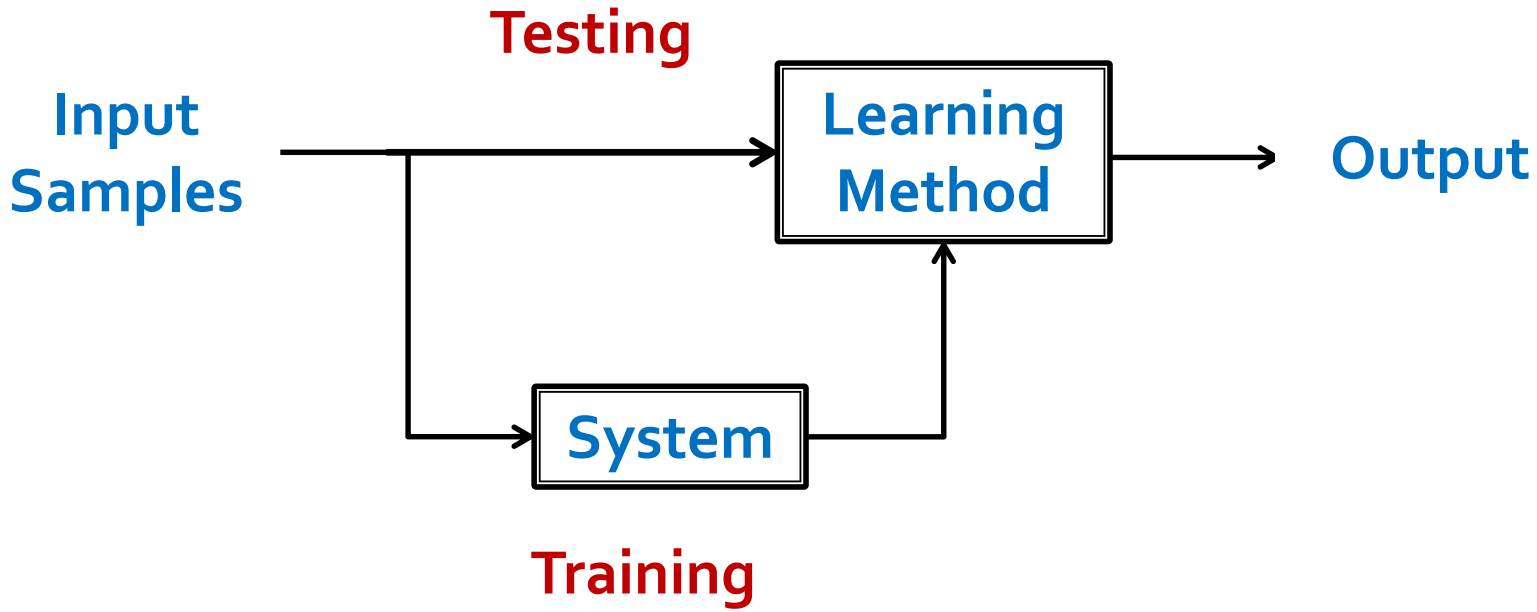
Machine Learning is…

Machine learning is about predicting the future based on the past.

-- Hal Daume III



Learning system model



Performance

- There are several factors affecting the performance:
  - Types of training provided
  - The form and extent of any initial background knowledge
  - The type of feedback provided
  - The learning algorithms used

What's a classifier ?

- A classifier assigns a label to each sample describing which category or class, that sample belongs to.

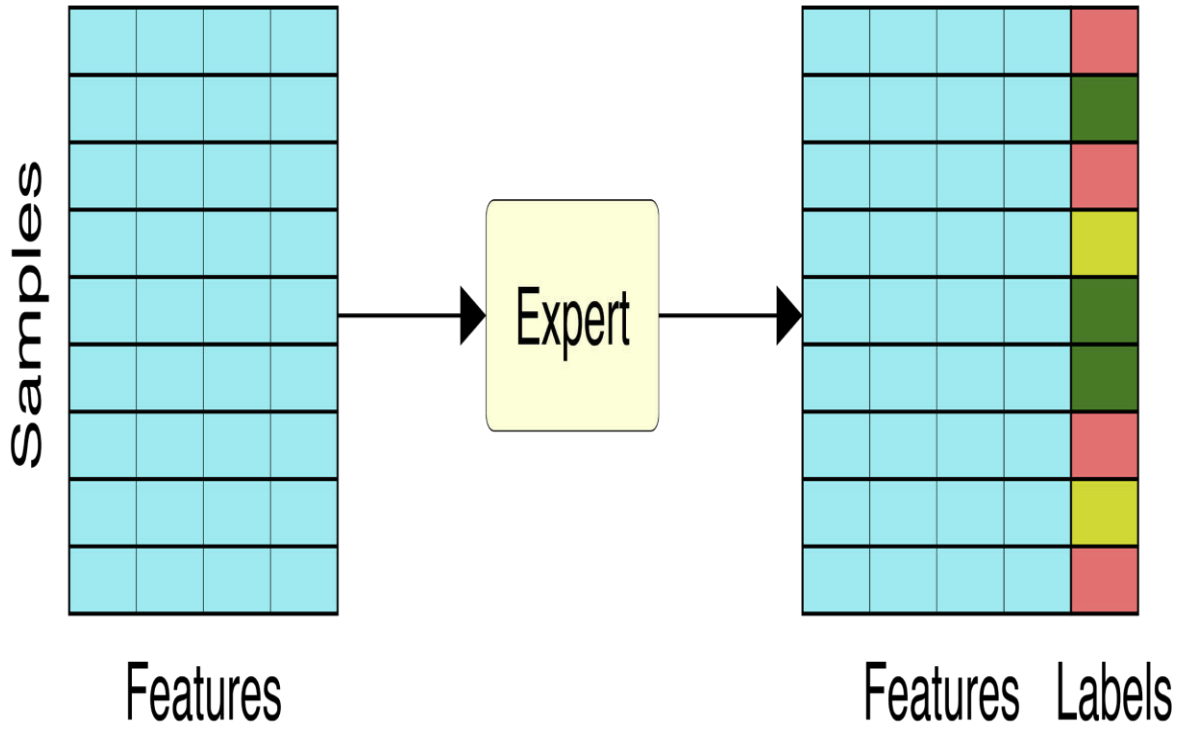
- Example of classifiers
  - If the input is a song, classifier assigns the label as the genre (e.g., rock or classical).
  - If it's a photo of an animal, the classifier assigns the label as the name of the animal shown (e.g., a tiger or an elephant).
  - In mountain weather for hiking, classifier may label the hiking experience into 3 categories: Lousy, Good, and Great.

Example of Samples, Features, Labels

- Weather measurements on a mountain for hiking
- Sample is weather at a given moment.
- Features are measurements: temperature, wind speed, humidity, etc.
- Hand over each sample (with a value for each feature) to a human expert.
- Expert examines features and provides a label for that sample.
- Expert's opinion, using a score from 0 to 100, tells how the day's weather would be for good hiking.
- Labels can be "Lousy", "Good", "Excellent" (weather for hiking)

Example of Samples, Features, Labels (Contd.)

- To label a dataset, we start with a list of samples, or data items.
- Each sample is made up of a list of features that describe it.
- We give the dataset to a human expert, who examines the features of each sample one by one, and assigns a label for that sample.

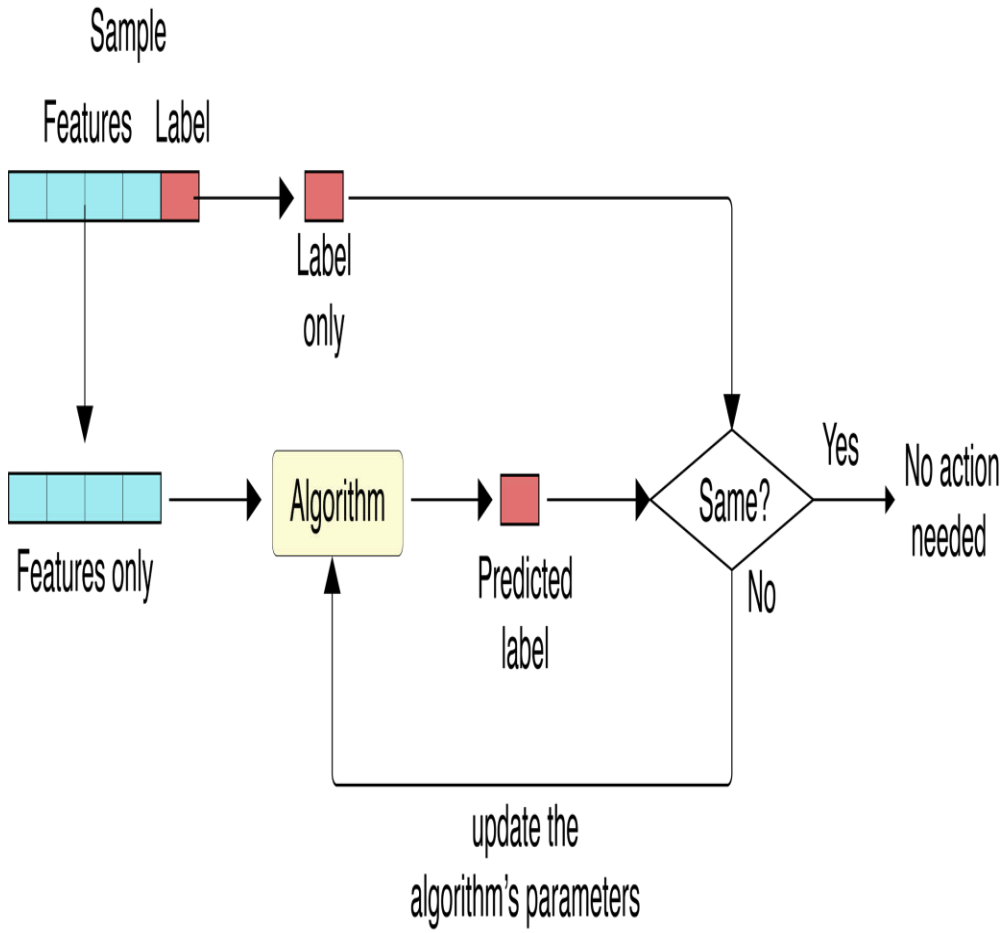


A Computerized Learning Strategy

- First, collect as much data as possible.
- Call each piece of observed data (say, the weather at a given moment) as sample.
- Call the names of the measurements that make it up (the temperature, wind speed, humidity, etc.) as features.
- Hand over each sample (with a value for each feature) to a human expert.
- Expert examines features and provides a label for that sample.
- Example: if our sample is a photo, the label might be the name of the person or the type of animal in the photo.

A Computerized Learning Strategy

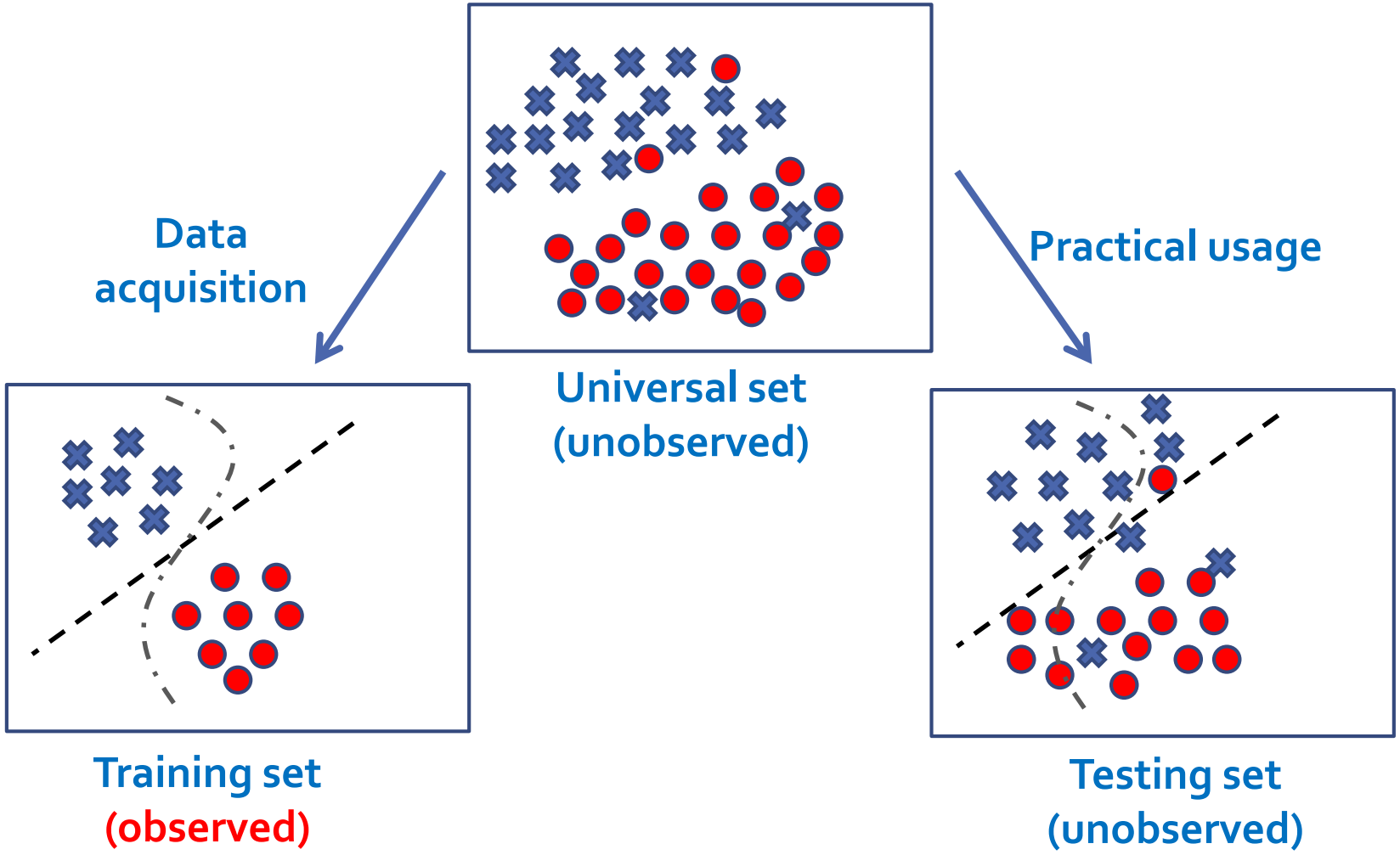
- Figure shows the idea of a learning strategy - one step of training or learning.
- Split the sample's features and its label. From the features, algorithm predicts a label.
- Compare prediction with truth label.
- If predicted label matches truth label, don't do any thing.
- Otherwise, tell algorithm to update itself
- The process is basically trial and error.



First, split Data for Training & Validation

- First, set aside some of these labeled samples for time being (use them later for validation).
- Give remaining labeled data to our computer, and ask it to find a way to come up with the right label for each input.
- We do not tell it how to do this.
- Instead, we give labelled data to an algorithm with a large number of parameters it can adjust (perhaps even millions of them).
- Different types of learning will use different algorithms.

Training and Testing

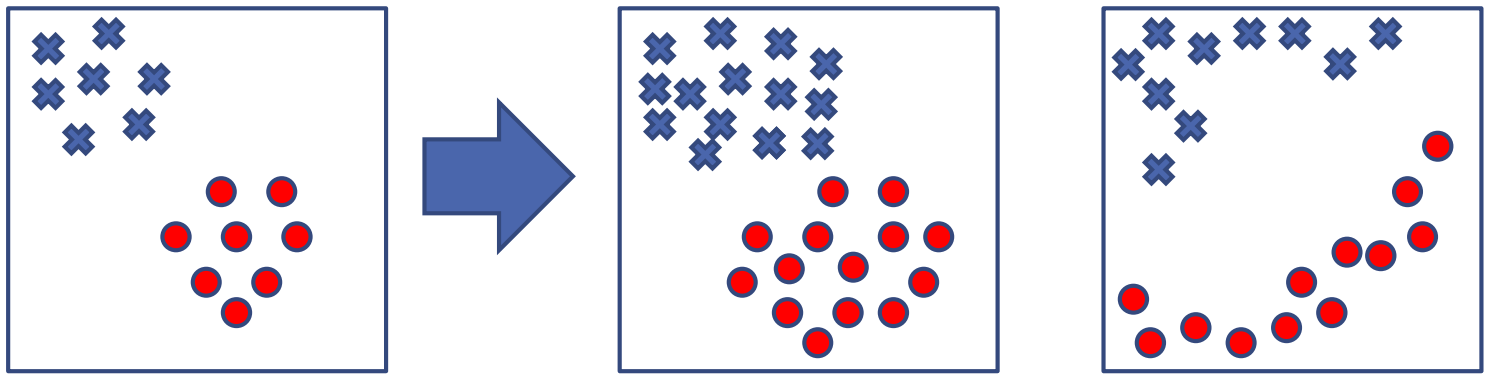


Training data

- Training is the process of taking a system that's been initialized with default or random values, and reconfiguring it so that it's tuned to the data we want to understand.
- Training data is data used to train the system.
- Real-world data is data the classifier will see in the real world, once it's been released, or deployed.
- Real-world data will only arrive after system has been deployed.

Training and testing

- **Training** is the process of making the system able to learn.
  - Training set and testing set come from the same distribution
  - Need to make some assumptions or bias



Parameters and hyper parameters

- Learning algorithm modifies itself its own parameter values, over time.
- Learning algorithm are also controlled by values that we set (such as the learning rate we saw above).
- These are called hyper parameters.

Difference between parameters and hyper parameters

- Computer adjusts its own parameter values during the learning process, while we specify the hyper parameters when we write and run our program.

Training step and Learning rate

- Each algorithm learns by changing the internal parameters it uses to create its predictions.
- Big change: risk of changing them so much that it makes other predictions worse.
- Small change: Cause learning to run slower.
- We have to find by trial and error for each type of algorithm the right trade-off between these extremes.
- We call the amount of updating the learning rate,
- A small learning rate is cautious and slow,
- A large learning rate speeds things up but could backfire.

Testing or Validation step

- We now return to the labeled data kept aside in the last section.
- This is called as test data.
- We evaluate how the system can generalize what it learned, by showing these samples that it's never seen before.
- This test set shows how the system performs on new data.

Validation

- Q: What is validation?
  - A: Validation is evaluating how well system is learning.
- Q: How do you validate a system?
  - A: By showing it new data(from training set, not real-world)it hasn't seen before, and measuring how accurately it works
- Q: What happens after validation?
  - A: If predictions on validating data are good, deploy system.
    - - If not good, then go back and train some more.

Cross-validation

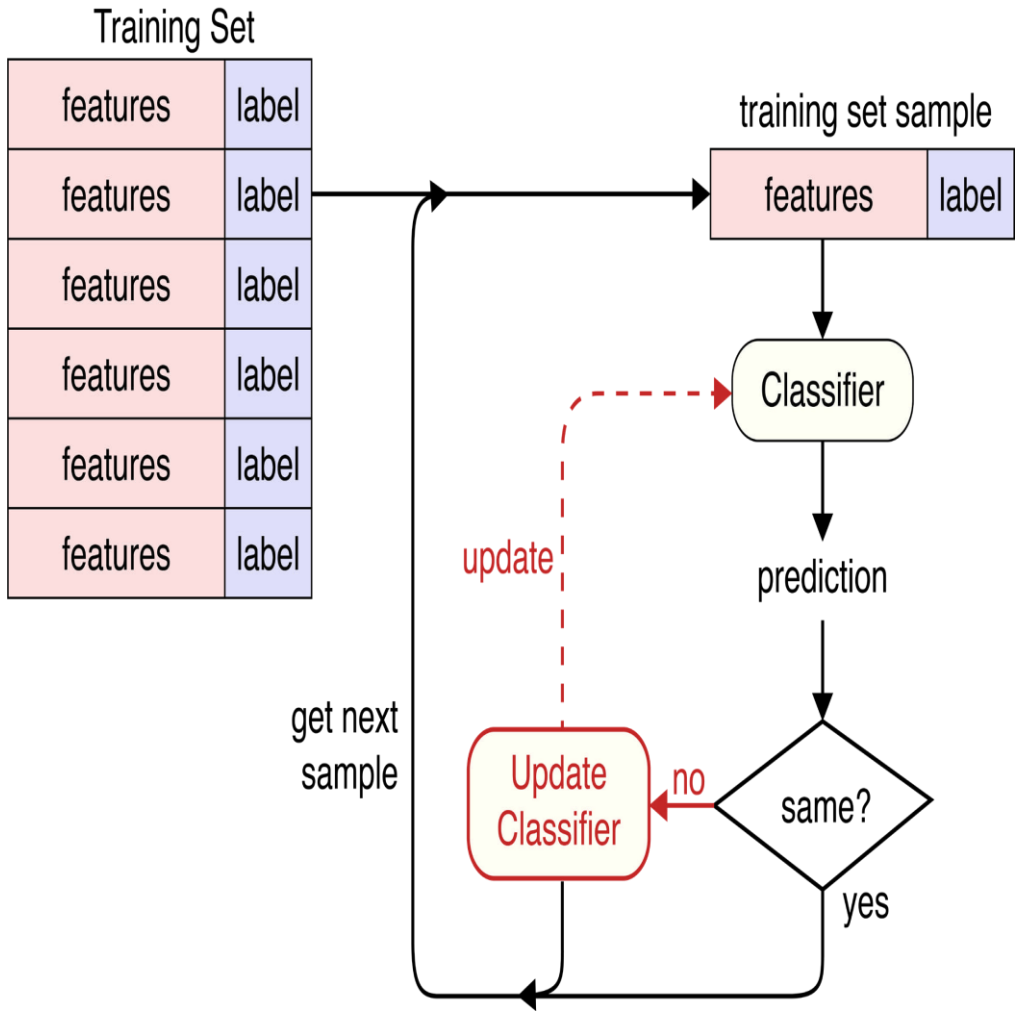
- Cross-validation is a validation tool
- It squeezes every bit of learning from training data.
- Cross-validation is specially useful when dataset is small.

Lets train and test a Categorizer (for example)

- We'll illustrate the ideas in this module using a supervised categorizer, which we'll teach with labeled data.
- In particular, we'll use a neural network type of classifier, which learns by repeated exposure to the training data.
- The techniques we'll discuss are general, and can be applied to all types of learners.

Lets train and test a Categorizer (for example)

- Present each sample in training set to classifier, one at a time.
- For each sample, give the sample's features and ask it to predict its category.
- If prediction is correct, then move to next sample.
- If prediction is wrong, feed classifier's output and correct label back to classifier.
- ML algorithm uses correct label, predicted label, and the current state of classifier to modify the classifier's internal parameters - so that it's likely to predict correct label if it sees this sample again.



How long to train? Epochs

- Epoch is a run through the entire training set.
- Usually you must train through many epochs.
- Keep training as long as the system is still learning and improving its performance on the test data.
- Stop when system's performance is good enough for the task.
- Stop also if you are out of time.

Testing performance – separate issue performance on training data is not enough

- Aim: Find how well categorizer will perform on real-world data before it is deployed. Categorizer should meet a quality threshold.

- Q: How to estimate quality of predictions before categorizer is released?
- Q: Is it enough if our categorizer does really well on training data?
  - A: No. If we just use the quality of the classifier's results only from training data, we'll be misled.

Testing performance – separate issue performance on training data is not enough

- Q: Why is good performance on training data alone not enough?

- A: Because we don't really know what the system learned from the training data.
- It might have learnt about some weird idiosyncrasies in training data and then used that as a rule, only to be foiled by new data that doesn't happen to have those quirks.

Run experiments to check performance

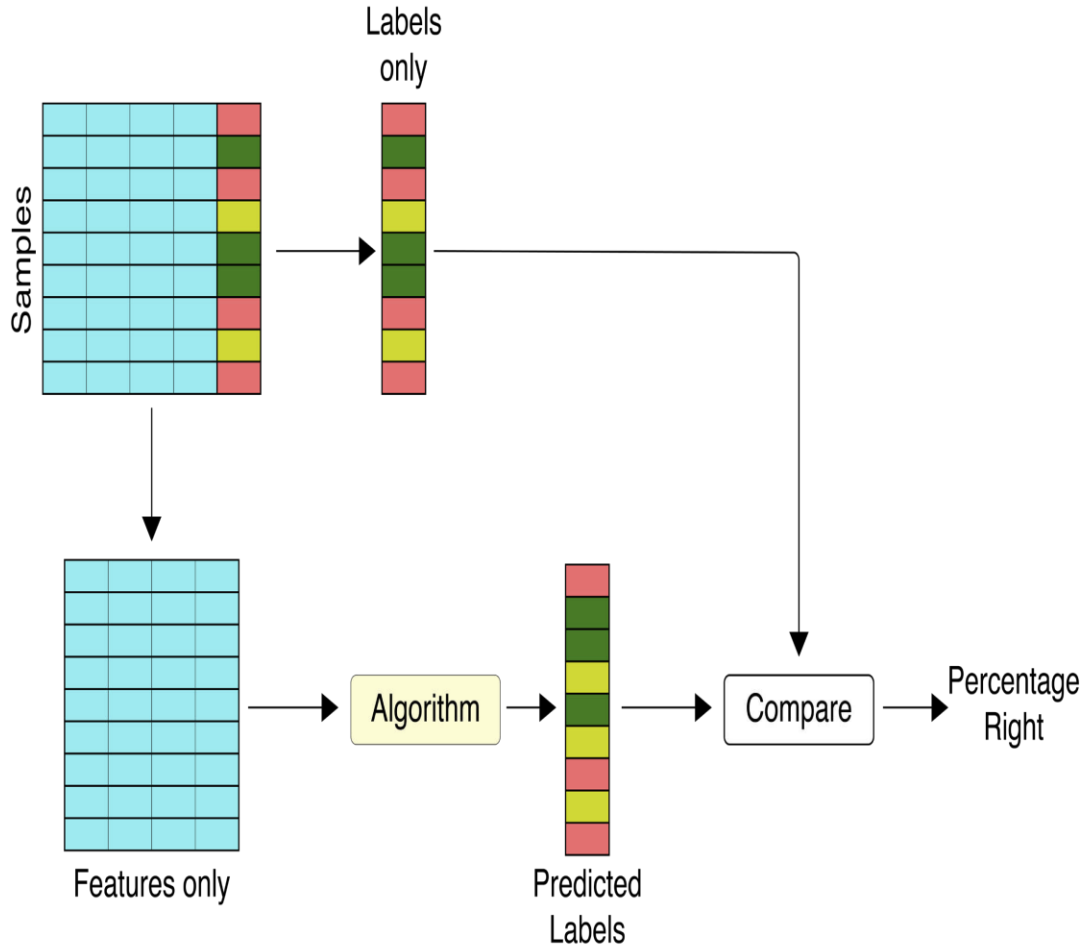
- Q: How do we then evaluate performance of the categorizer?
- A: There's no way for us to know how good our categorizer is without trying it out and seeing.
- We must run experiments to see how well our system performs.
- The best way anyone has found to work out how well a system will do on new, unseen data is to give it new, unseen data and see how well it does.
- There's no shortcut to this kind of experimental verification.

Test data – a representation for real-world data

- Test data is the new unseen data used for testing.
- We hope test data represents real-world data that system will see when deployed.
- If performance on test data isn't good, then train it more.
- Irrespective of how well it might have done on the training data.

Testing – Procedure for evaluating a classifier

- Split the test data (not training data) into features and labels.
- Algorithm predicts a label for each set of features.
- Compare predictions with the
- truth labels to get a measurement of accuracy.
- If it's good enough, deploy the system.
- If the results aren't good enough, go back and train some more.
- In this evaluation process there is no feedback and no learning.



Never learn from test data

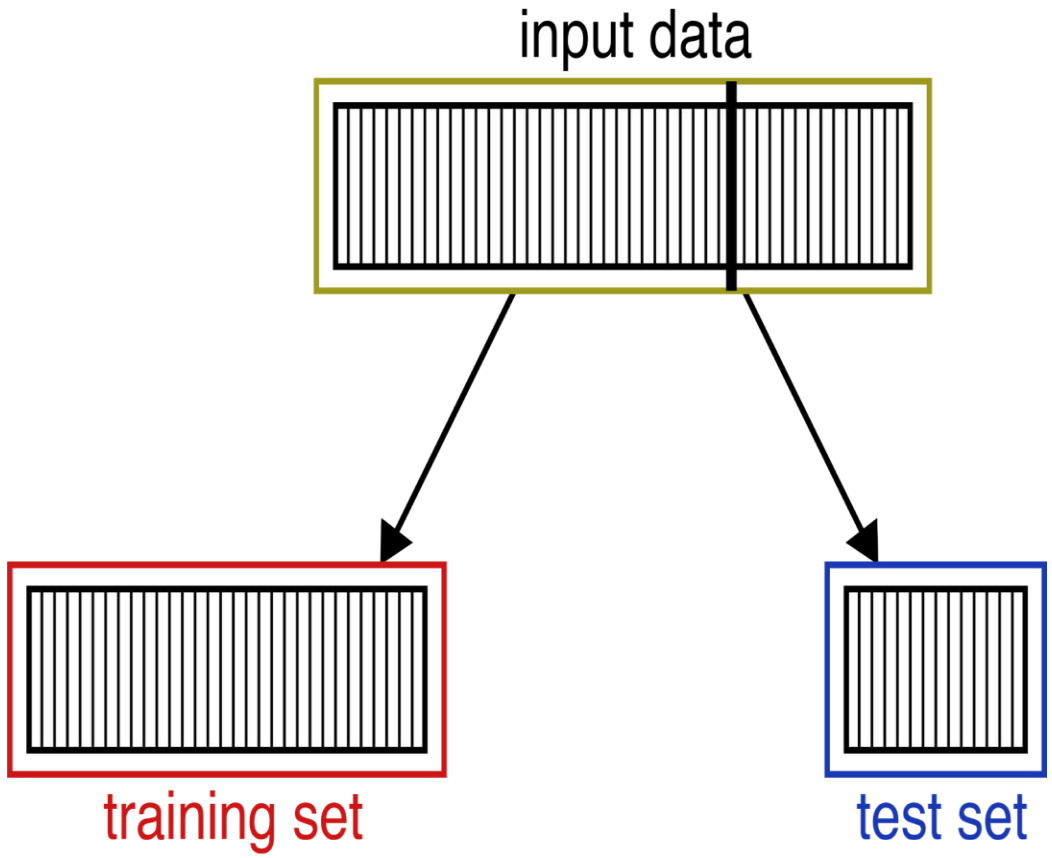
- Never learn from the test data.
- If categorizer learns from test data, it loses its special and valuable quality as a way to measure the performance of the system on new data.
- This would ruin the reliability of our measure of the general ability of our system.
- So, this would be "cheating," because it's already learned from test data.
- So keep test set hidden from the system until training is over, and then use it just once to estimate its performance.

Lookout for Data leaks

- Data leakage, or data contamination happens when system learns from test data.
- Watch out for this !
- Data leaks can sneak in, as they are hard to notice.

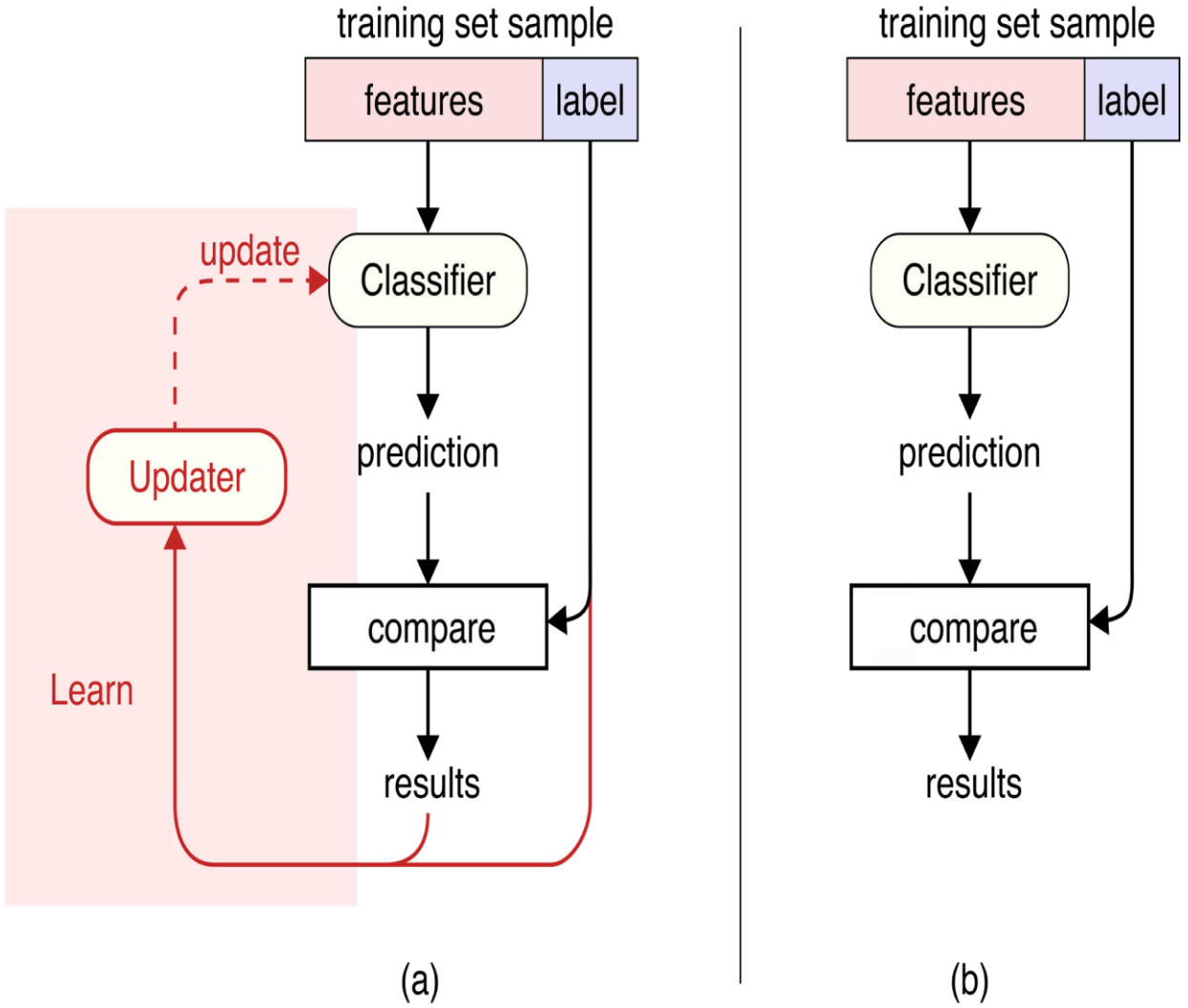
How to create training and test data sets?

- Create test data by splitting original data collection into two pieces: training set and test set.
- Split to give about 75% of the samples to training set.



Difference between training and testing

- In training, compare the prediction and the real label, and use any error between them to make classifier learn to do better.
- In testing, omit learning step.

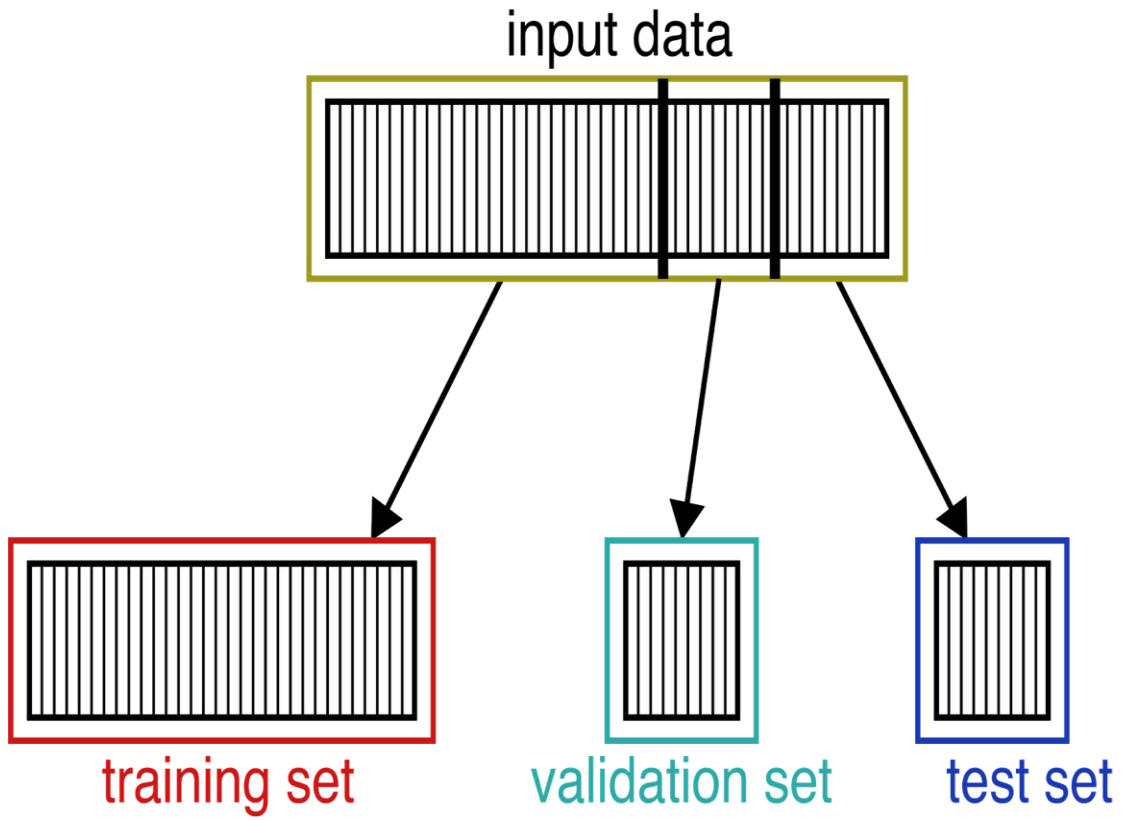


To try hyperparameters, use validation data

- To try out many values of hyperparameters, create and use validation data.
- For each variation of hyperparameters, train on the training set, and evaluate the system's performance on the validation set.
- Select the hyperparameters that gave us the best results.
- Run that through the test set to evaluate the system's performance on data that it's never seen before.
- In other words, use the validation set as part of the learning process.
- The test set is still held aside for that one-time final evaluation.

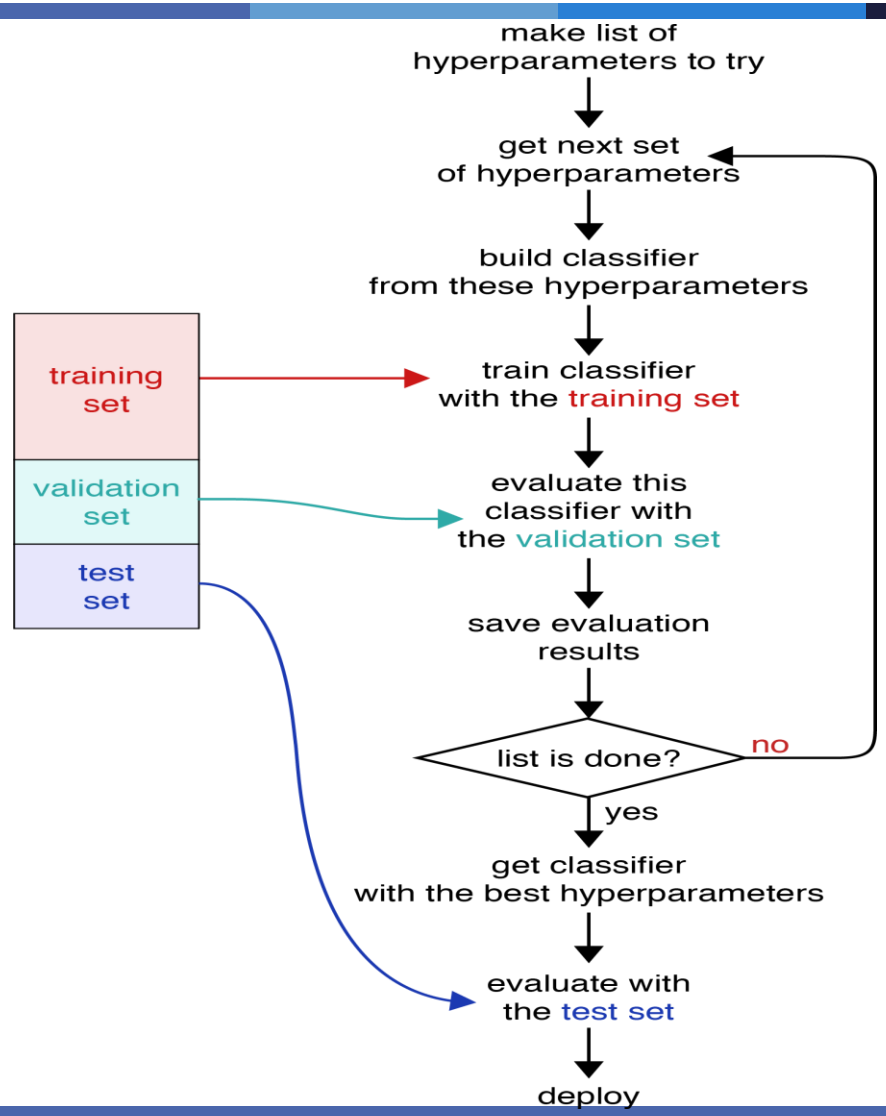
Creating Validation data set

- Splitting input data into a training set, validation set, and test set.
- Split original data into three pieces, 60% for training, 20% for testing, and 20% for validation.



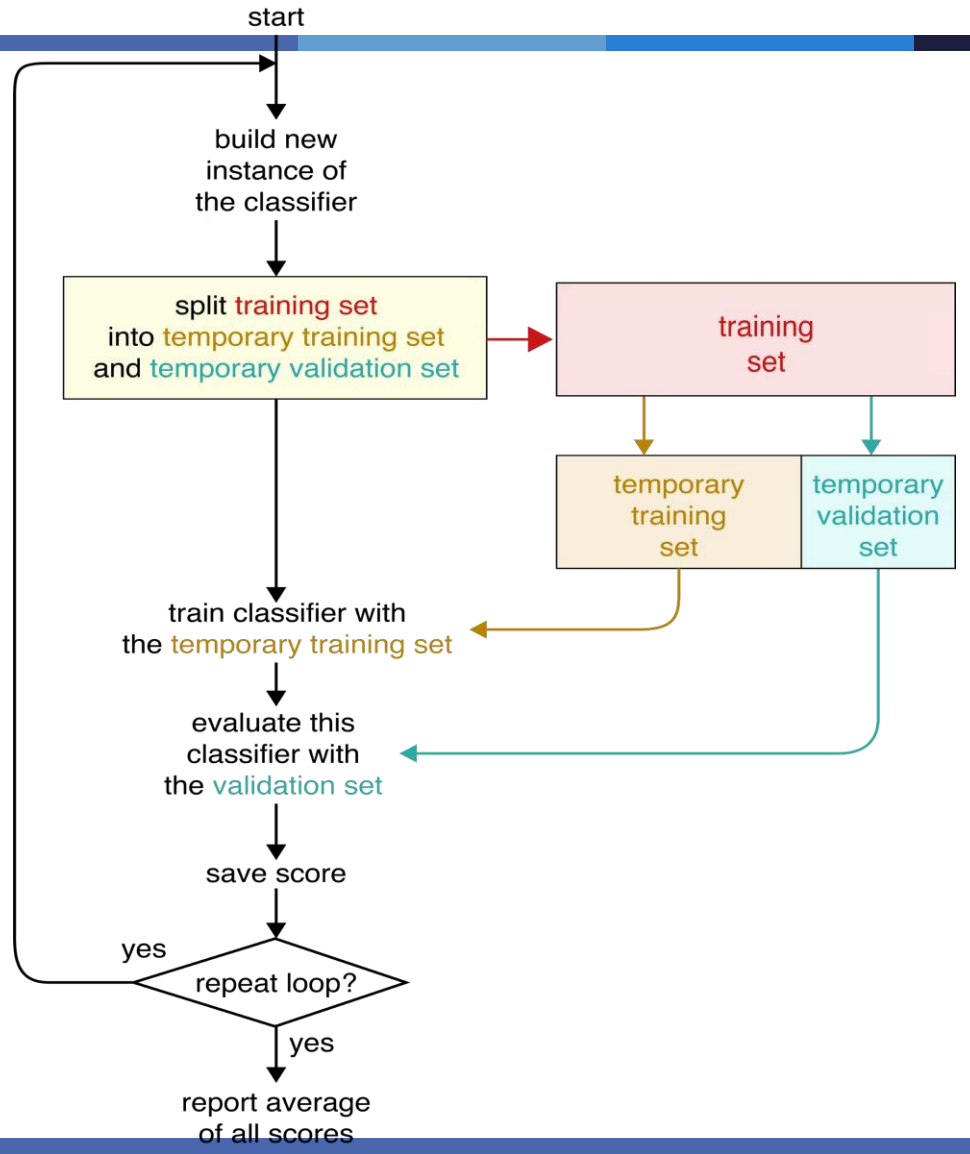
Validation Process- for trying lots of hyperpairs

- Run the Loop:
  - Select a set of hyperparameters, Train the system,
  - Test it with validation set.
  - Repeat with new set of hyperpars
- Select set of hyperpars that gave best performance.
- Take that system, apply it to testing data. & check how good its predictions are.
- If good, deploy. Else, retrain
- We'll see next a more sophisticated method called cross-validation that does not re-use the same validation set over and over in this way.

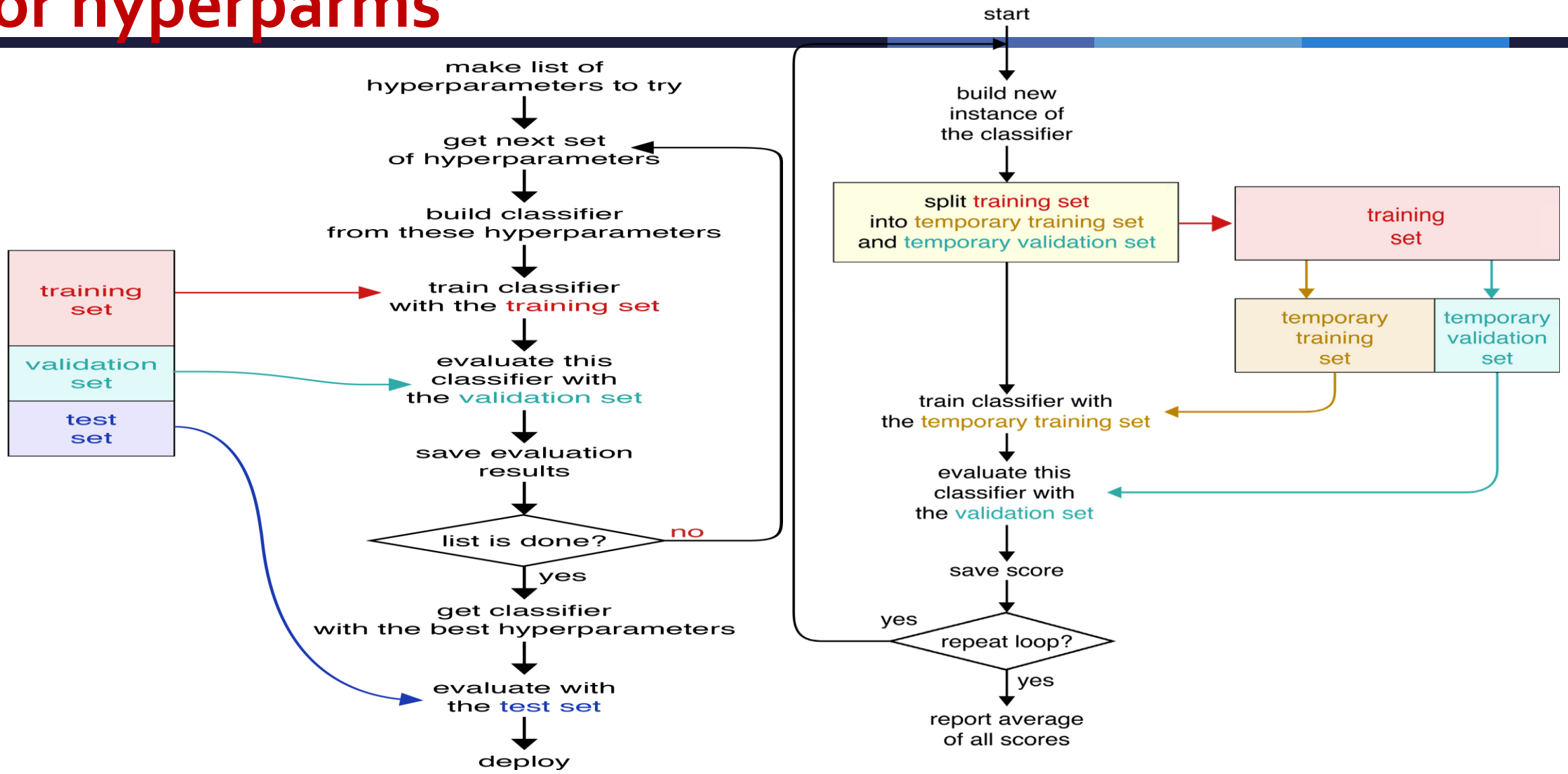


Validation Process- for trying lots of hyperpairs

- Before starting, pull out the test data and set it aside.
- Work with rest of data- training data set
- Each time through loop, split training data into a temporary training set and temporary validation set.
- Begin by building a fresh instance of our classifier.
- Train on temporary training set, and evaluate with temporary validation set.
- Get score for current classifier's performance.
- Go through loop again, split data into new sets, different from any we've tried before.
- Loop over and over, splitting data into fresh sets, training and validating, and getting a score.
- Find average of all scores of our classifier.



Inner loop of Validation- k-fold Cross validation - for hyperparms

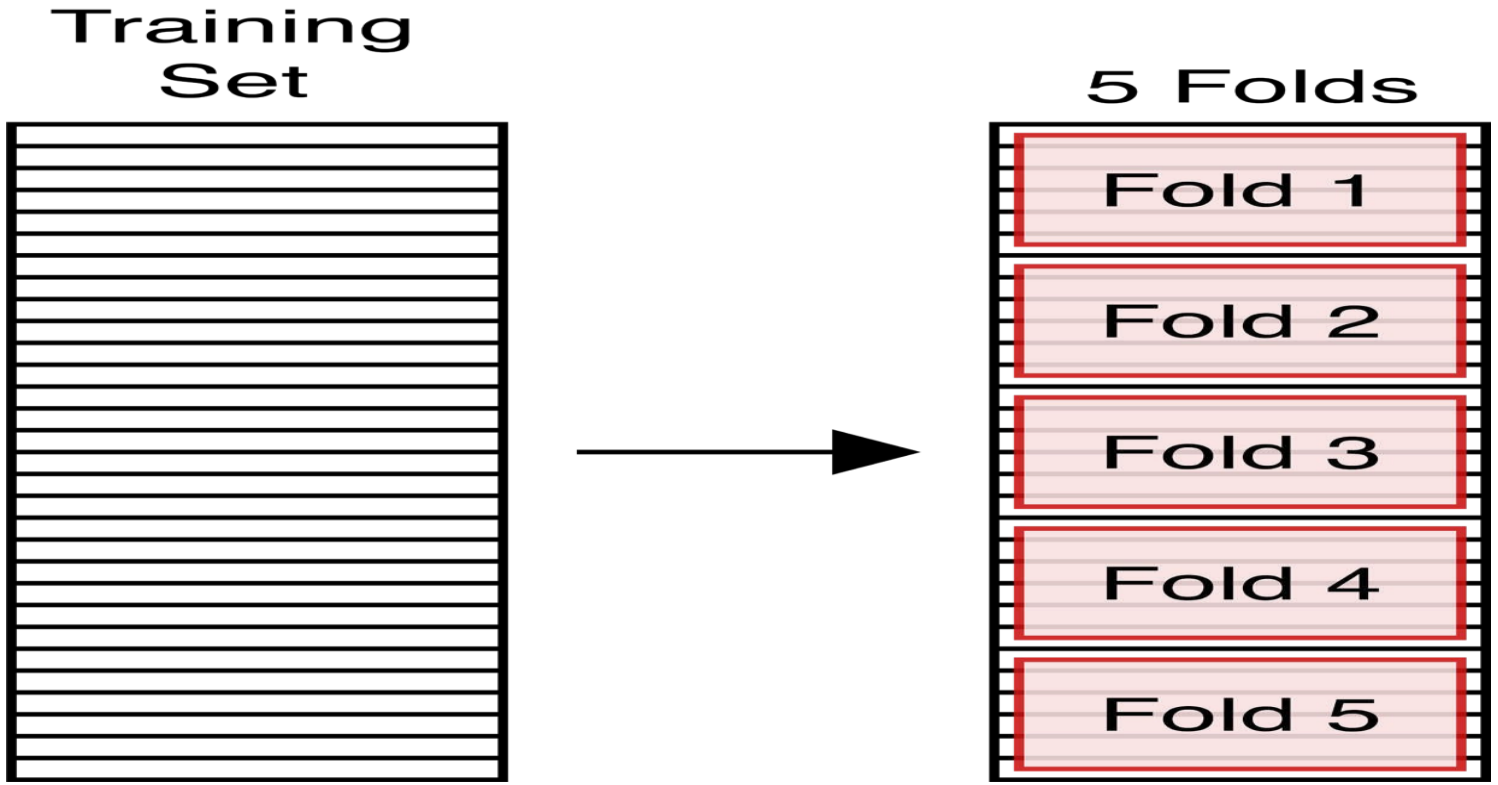


K-fold cross- validation

- There are a variety of different algorithms available for constructing the temporary training and validation sets.
- Let's look at a popular approach for constructing temporary data sets. Called as K-fold cross-validation
- Popular way to build the temporary datasets for cross-validation is called k-fold cross-validation.
- For example, "2-fold cross-validation" or "5-fold cross- validation". Typically, the value of k is the number of times we want to go through the loop.
- The algorithm starts before the cross-validation loop of Figure begins.
- We take our training data and split it up into a series of equal-sized chunks. We call each chunk a fold.

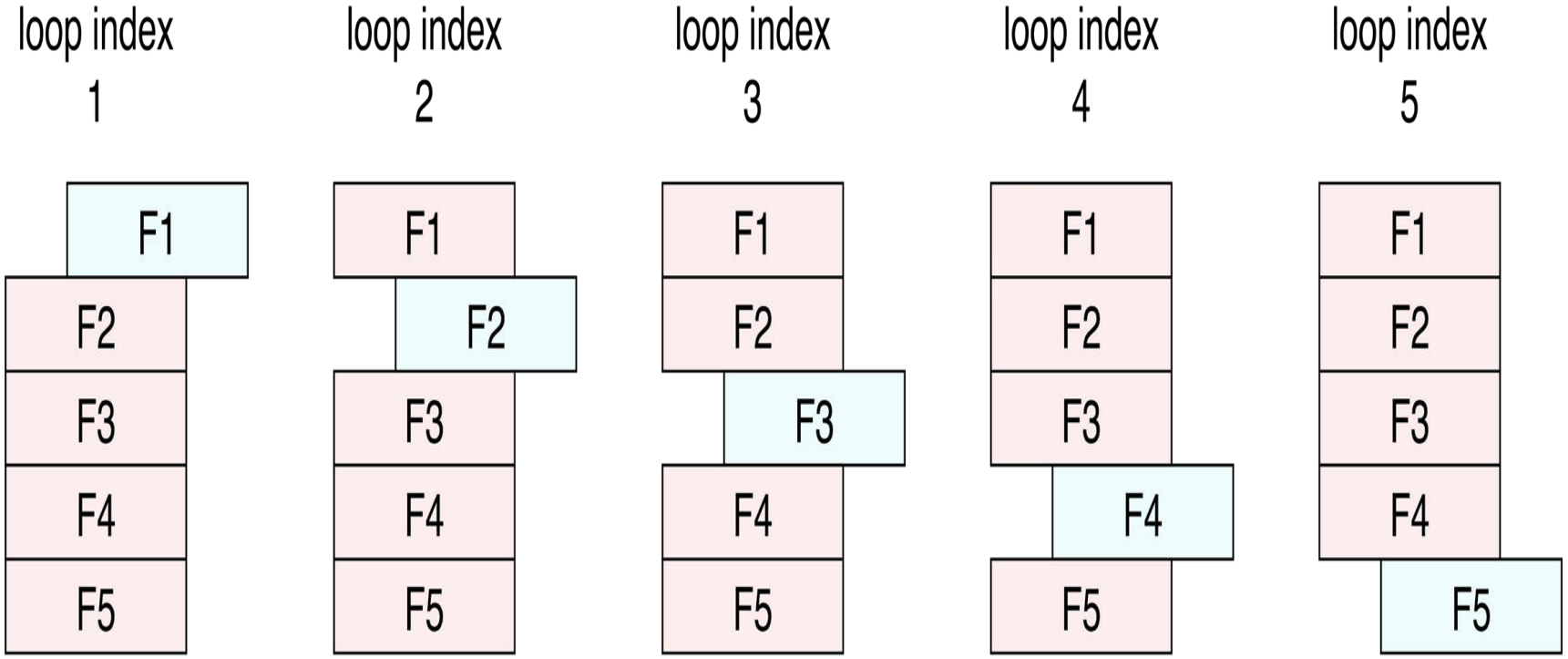
5-fold – splitting data set

Splitting our training set into 5 equally-sized folds, named Fold 1 through Fold 5.



Example of 5-fold

- In each pass through the loop, we choose one fold for validation (in blue), and train with the others (in pink).
- If we go through the loop more than 5 times, we just repeat the pattern.



How to Retrain, if testing results are not good

- Use again original training set data. Note that these are the same samples.
- Shuffle this data first - but no new information.
- Show every sample again, letting it learn along the way again.
- Computer learns over and over again from the very same data.
- Now, show test data set .
- Ask algorithm to predict labels for the test set again.
- If the performance isn't good enough, go back to original training set again, and then test again.
- Repeat this process often hundreds of times. Let it learn just a little more each time.
- Computer doesn't get bored or cranky seeing the same data over and over.

Learning – Good and Bad News

- **Bad News:**
  - No guarantee that there's a successful learning algorithm for every set of data,
  - No guarantee that if there is one, we'll find it.
  - May not have enough computational resources to find the relationship between the samples and their labels.
- **Good news:**
  - Even without a mathematical guarantee, in practice we can often find solutions that generalize very well, sometimes doing even better than human experts.

When do we deploy the System ?

- When the algorithm has learned enough to perform well enough on the test set that we're satisfied, we're ready to deploy, or release, our algorithm to the world.
- Users submit data and our system returns the label it predicts.
- That's how pictures of faces are turned into names, sounds are turned into words, and weather measurements are turned into forecasts.

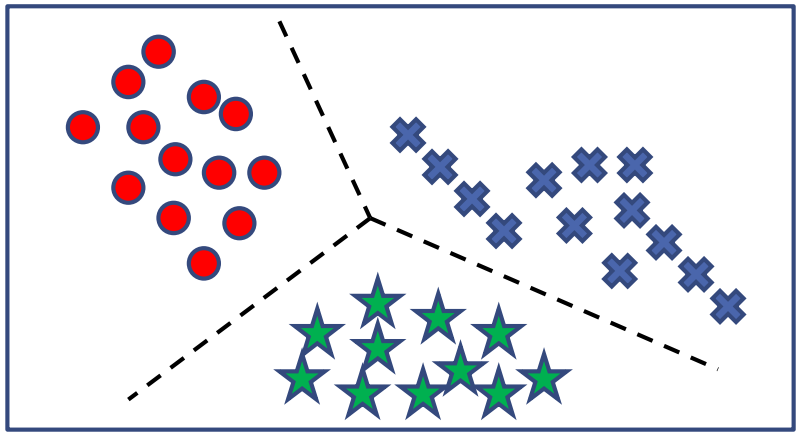
Machine Learning – Major categories-Algorithms

- The success of machine learning system also depends on the algorithms.
- The algorithms control the search to find and build the knowledge structures.
- The learning algorithms should extract useful information from training examples.

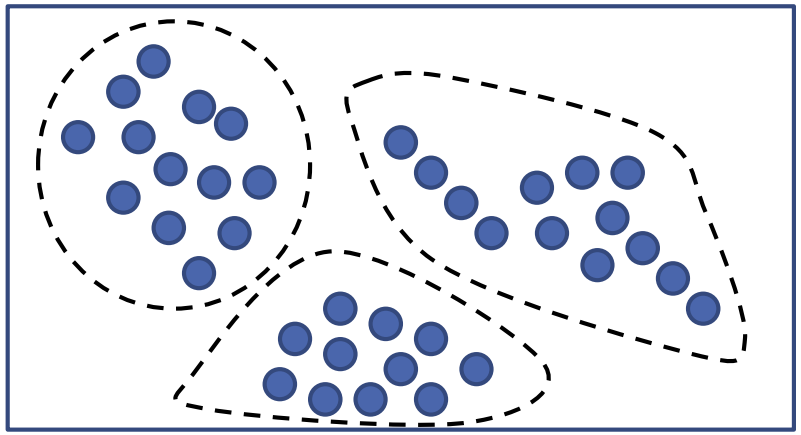
Algorithms

- Supervised learning
  - Prediction
  - Classification (discrete labels), Regression (real values)
- Unsupervised learning
  - Clustering
  - Probability distribution estimation
  - Finding association (in features)
  - Dimension reduction
- Semi-supervised learning
- Reinforcement learning
  - Decision making (robot, chess machine)

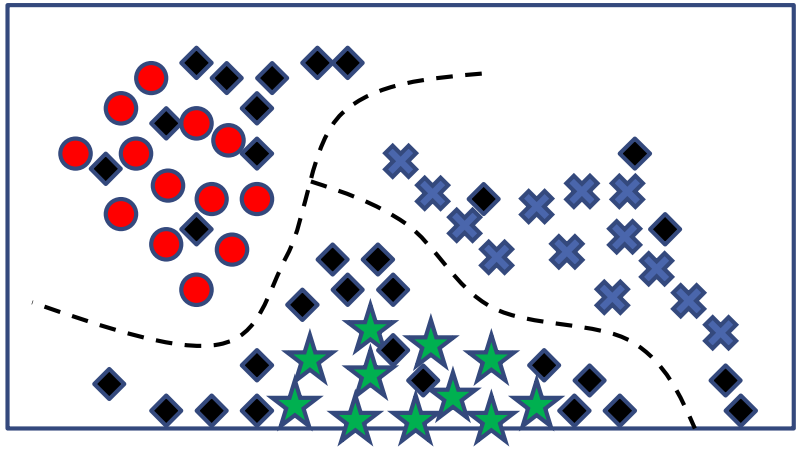
Algorithms



Supervised learning



Unsupervised learning



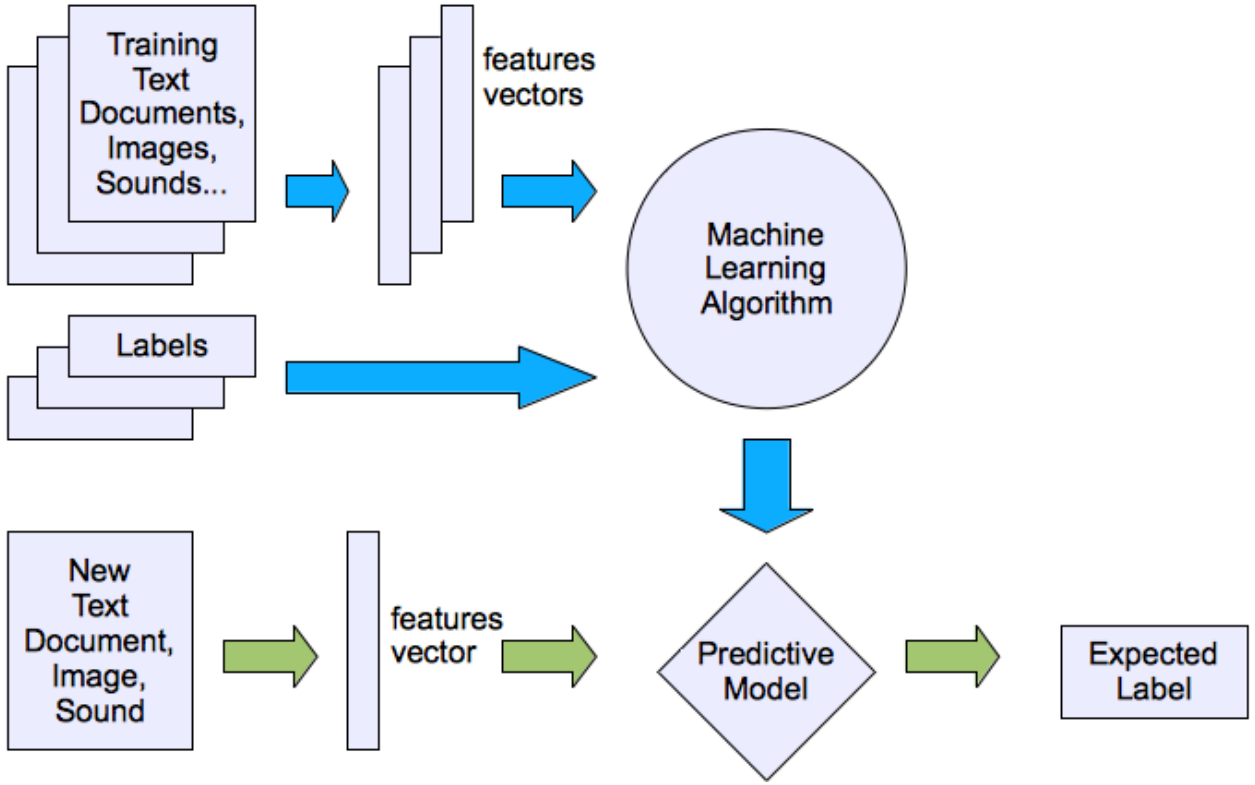
Semi-supervised learning

Supervised Learning (ML)

- Supervised learning (SL) is done for samples with pre-assigned labels.
- Supervision comes from the labels.
- Labels guide the comparison step.
- There are two general types of supervised learning, called classification and regression.
- Classification: look through a given collection of categories to find the one that best describes a particular input.
- Regression: take a set of measurements and predict some other value

Machine learning structure

- Supervised learning



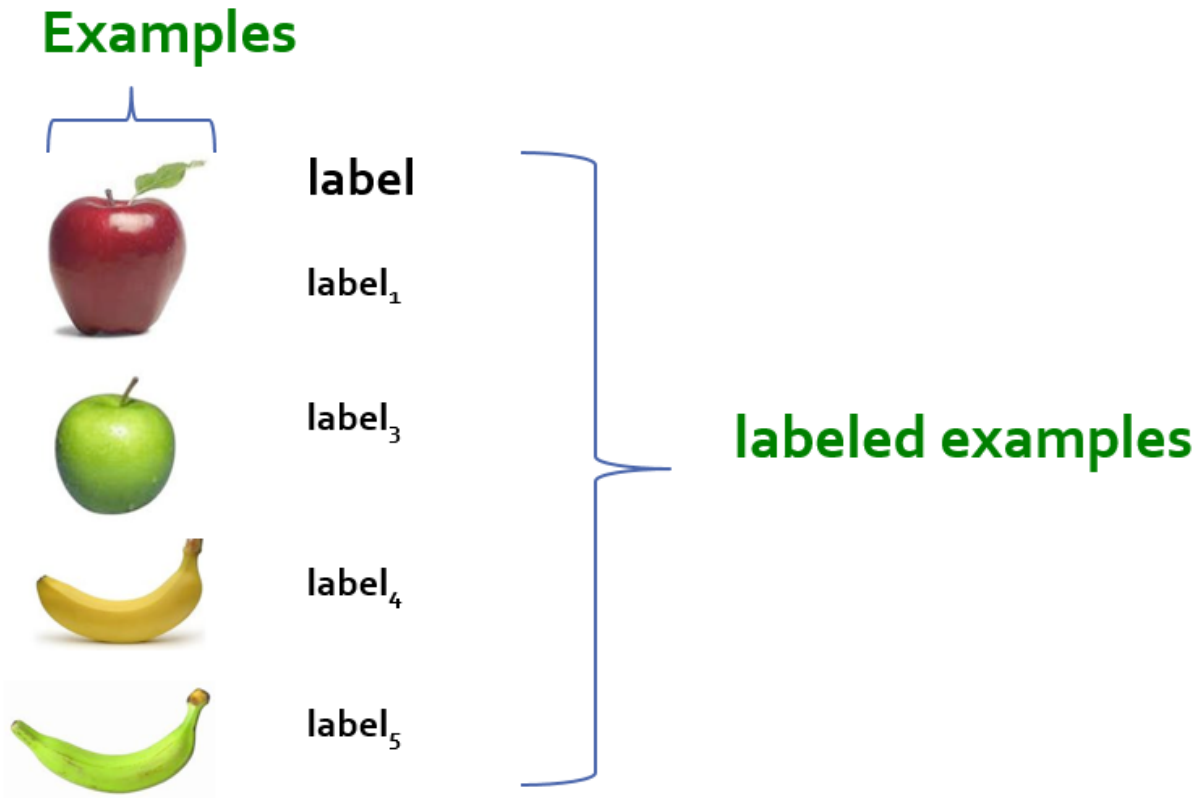
SL – Classification

- Start training by providing a list of all the labels (or classes, or categories) that we want it to learn.
- Make the list so that it has all the labels for all the samples in the training set, with the duplicates removed.
- Train the system with lots of photos and their labels, until it does a good job of predicting the correct label for each photo.
- Now, turn the system loose on new photos it hasn't seen before.
- For those objects it saw during training, It should properly label images.
- Caution: For those objects it did not see during training, the system will try to pick the best category from those it knows about.
- Next Figure shows the idea.

SL – Classification Example

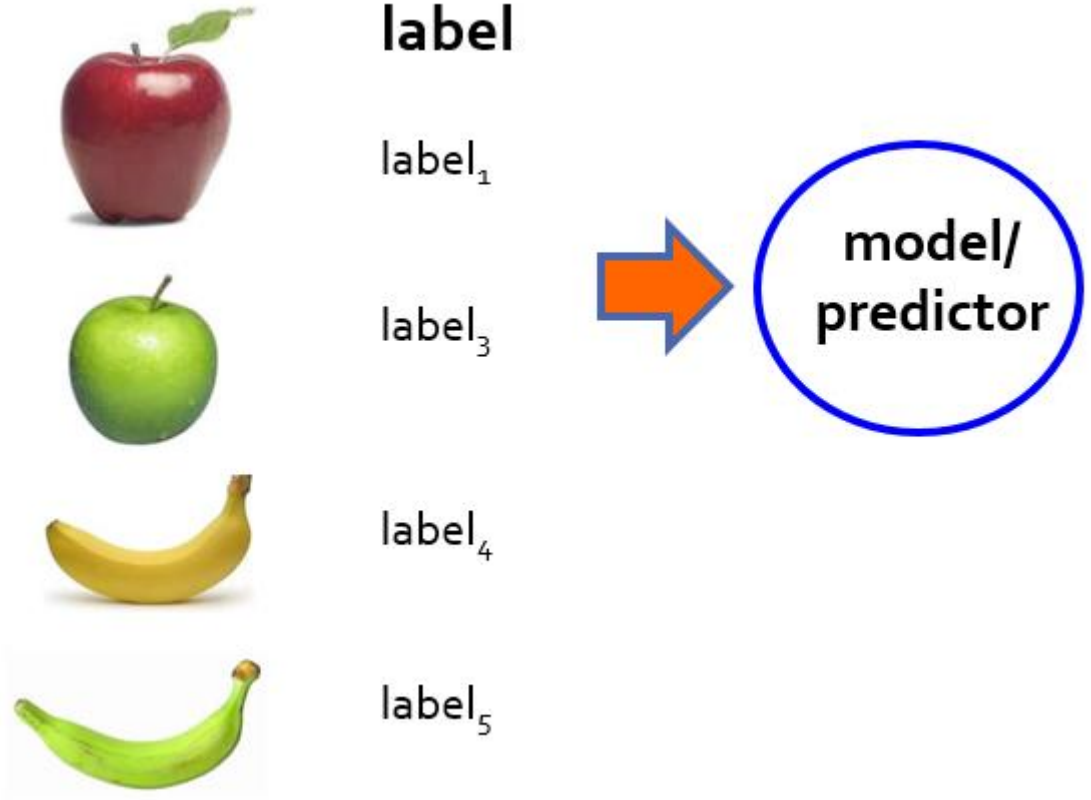
- **Example:** Sort and label photos of everyday objects.
- We want to sort them : an apple peeler, a salamander, a piano, and so on.
- We want to classify or categorize these photos.
- The process is called classification or categorization.

SL – Classification Example



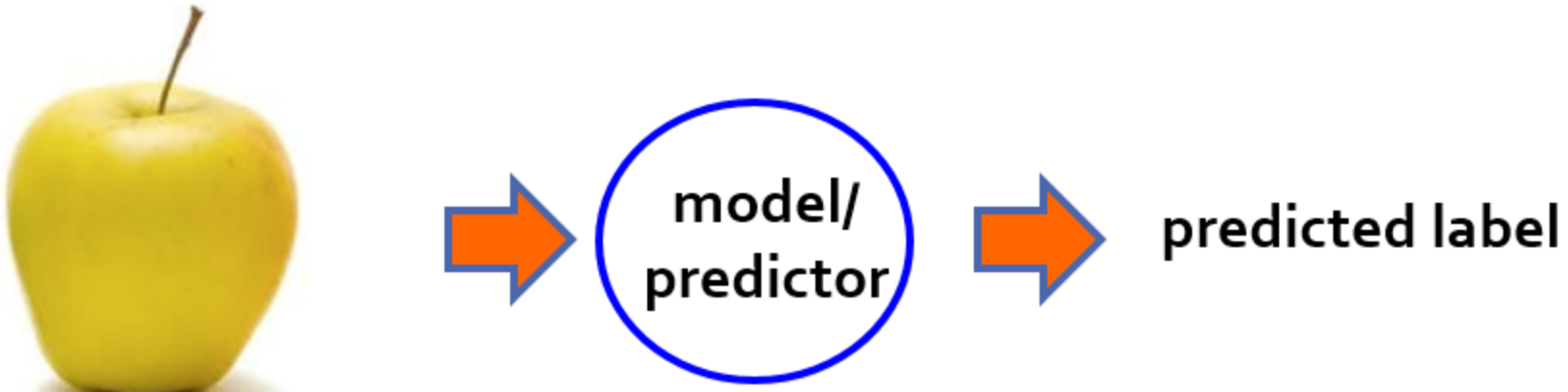
Supervised learning: given labeled examples

SL – Classification Example



Supervised learning: given labeled examples

SL – Classification Example



Supervised learning: learn to predict new example

SL – Classification Example

label

apple

apple

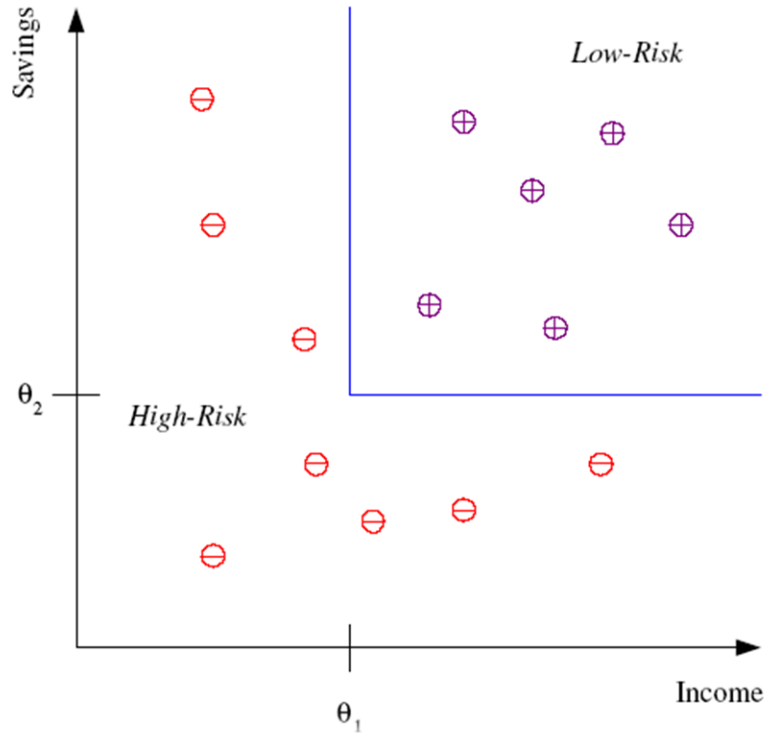
banana

banana

Classification: a finite set of labels

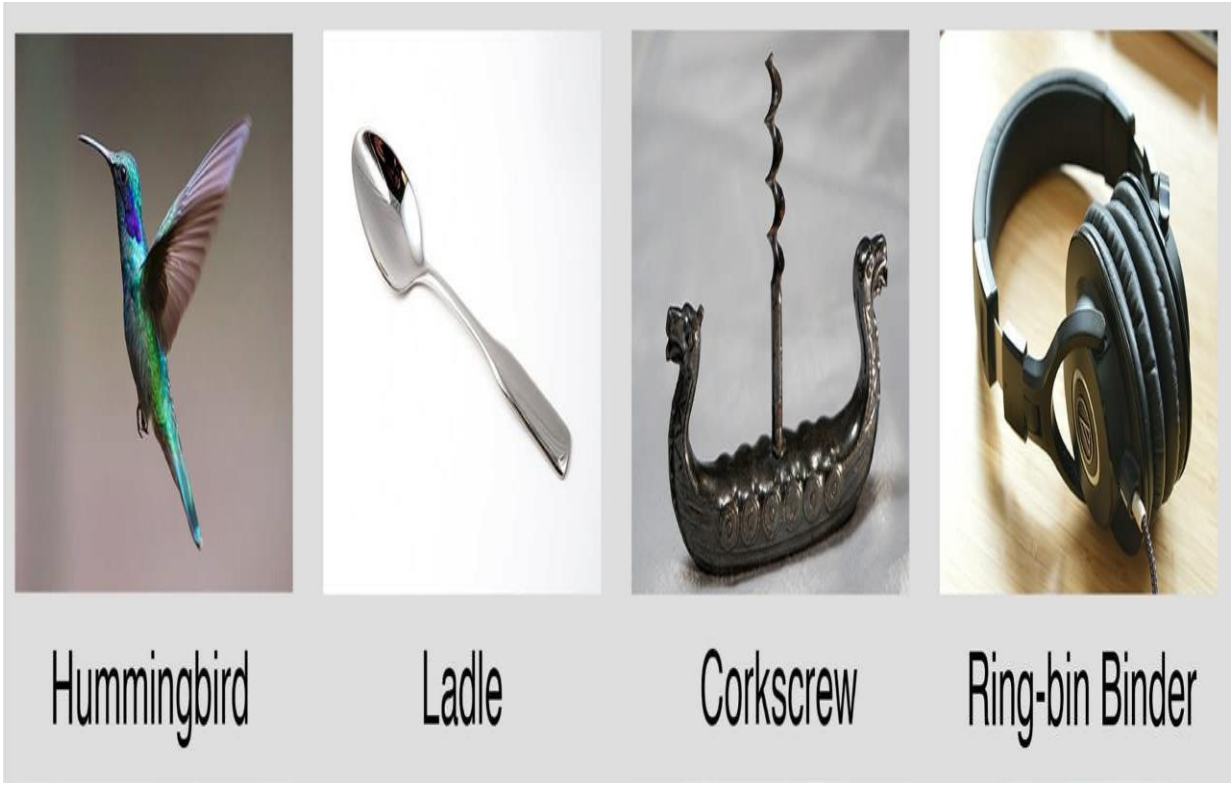
SL – Classification Example

Differentiate between low-risk and high-risk customers from their *income* and *savings*



SL – Classification Example

- In Figure, we used a trained classifier to identify four images never seen before.
- The system had not been trained on metal spoons or headphones,
- In both cases it found best match it could.
- To correctly identify those objects, the system needs to see multiple examples of them during training.

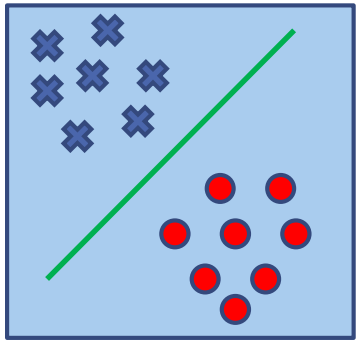


Learning techniques

- **Supervised learning categories and techniques**
  - Linear classifier (numerical functions)
  - Parametric (Probabilistic functions)
    - Naïve Bayes, Gaussian discriminant analysis (GDA), Hidden Markov models (HMM), Probabilistic graphical models
  - Non-parametric (Instance-based functions)
    - *K*-nearest neighbors, Kernel regression, Kernel density estimation, Local regression
  - Non-metric (Symbolic functions)
    - Classification and regression tree (CART), decision tree
  - Aggregation
    - Bagging (bootstrap + aggregation), Adaboost, Random forest

Learning techniques

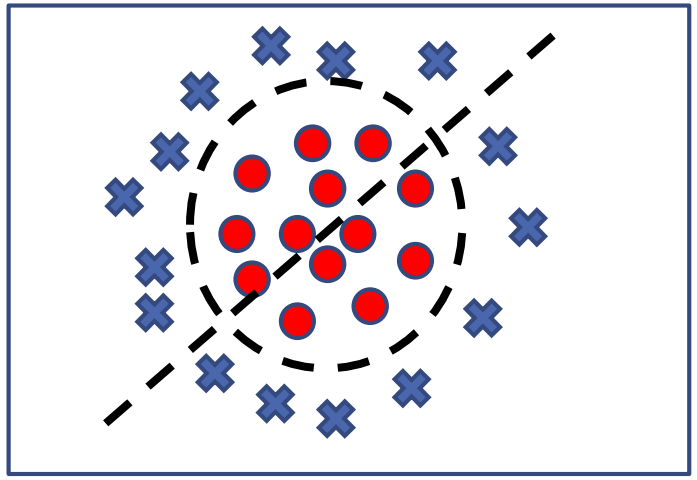
- Linear classifier



- Techniques:
  - Perceptron
  - Logistic regression
  - Support vector machine (SVM)
  - Ada-line
  - Multi-layer perceptron (MLP)

Learning techniques

- **Non-linear case**



- **Support vector machine (SVM):**
  - **Linear to nonlinear:** Feature transform and kernel function

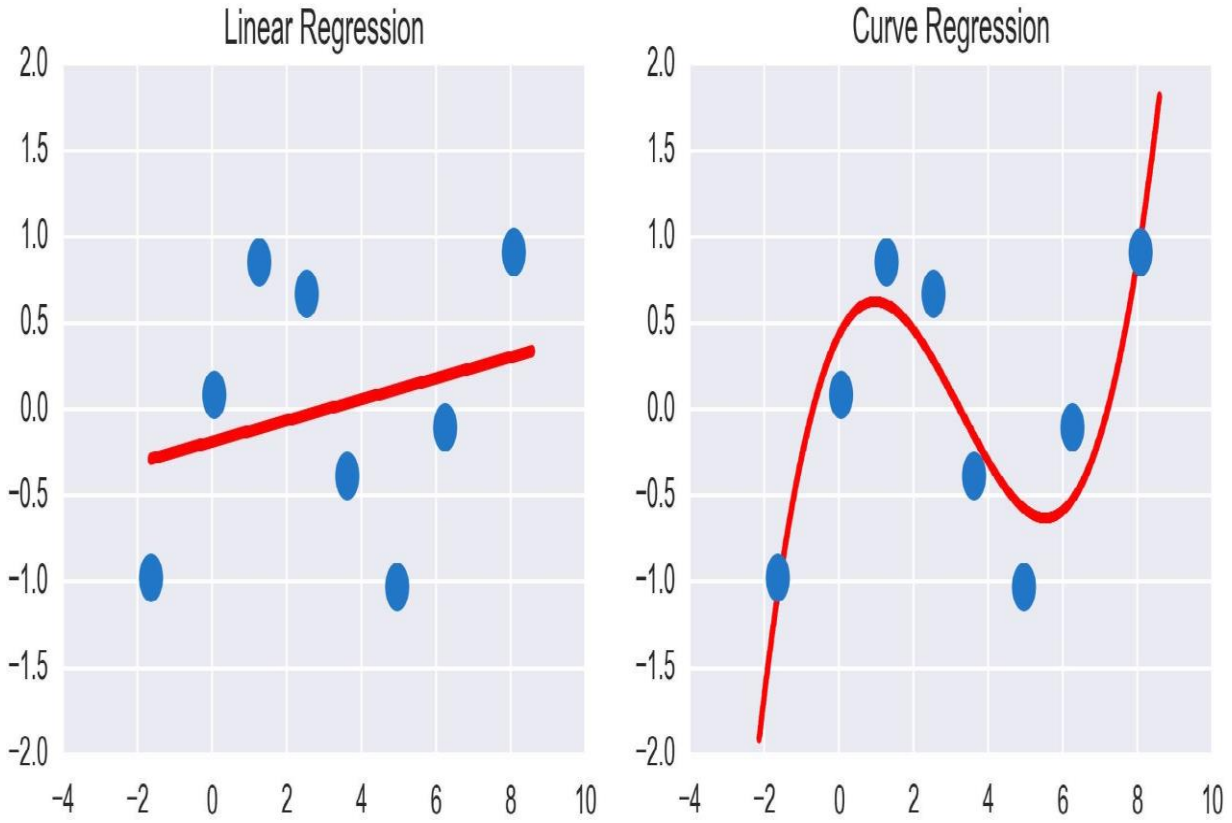
Classification Applications

- **Face recognition**: Pose, lighting, occlusion (glasses, beard), make-up, hair style

- **Character recognition**: Different handwriting styles.
- **Speech recognition**: Temporal dependency.
- **Medical diagnosis**: From symptoms to illnesses
- **Web Advertizing**: Predict if a user clicks on an ad on the Internet.
- **Biometrics**: Recognition/authentication using physical and/or behavioral characteristics: Face, iris, signature, etc

SL - Regression

- Regression is process of filling in or predicting data .
- "Regression" uses statistical properties of the data to estimate missing or future values.
- Most famous kind of regression is linear regression.
- Left: Linear regression fits a straight line (red) to the data (blue). The line is not a very good match to the data, but it has the benefit of being simple.
- Right: Nonlinear regression fits a curve to same data. This is a better match to the data, but has more complicated form and requires more work (and thus more time)



Supervised learning: Regression

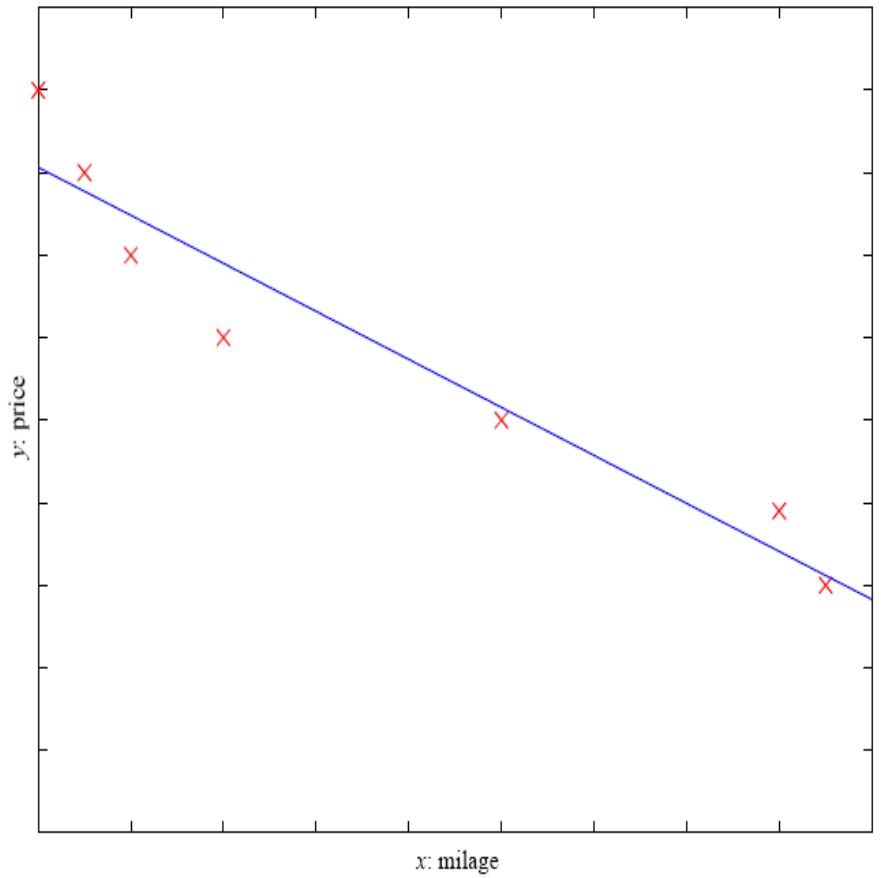
	label
	-4.5
	10.1
	3.2
	4.3

Regression: label is real-valued

Regression Example

- Price of a used car

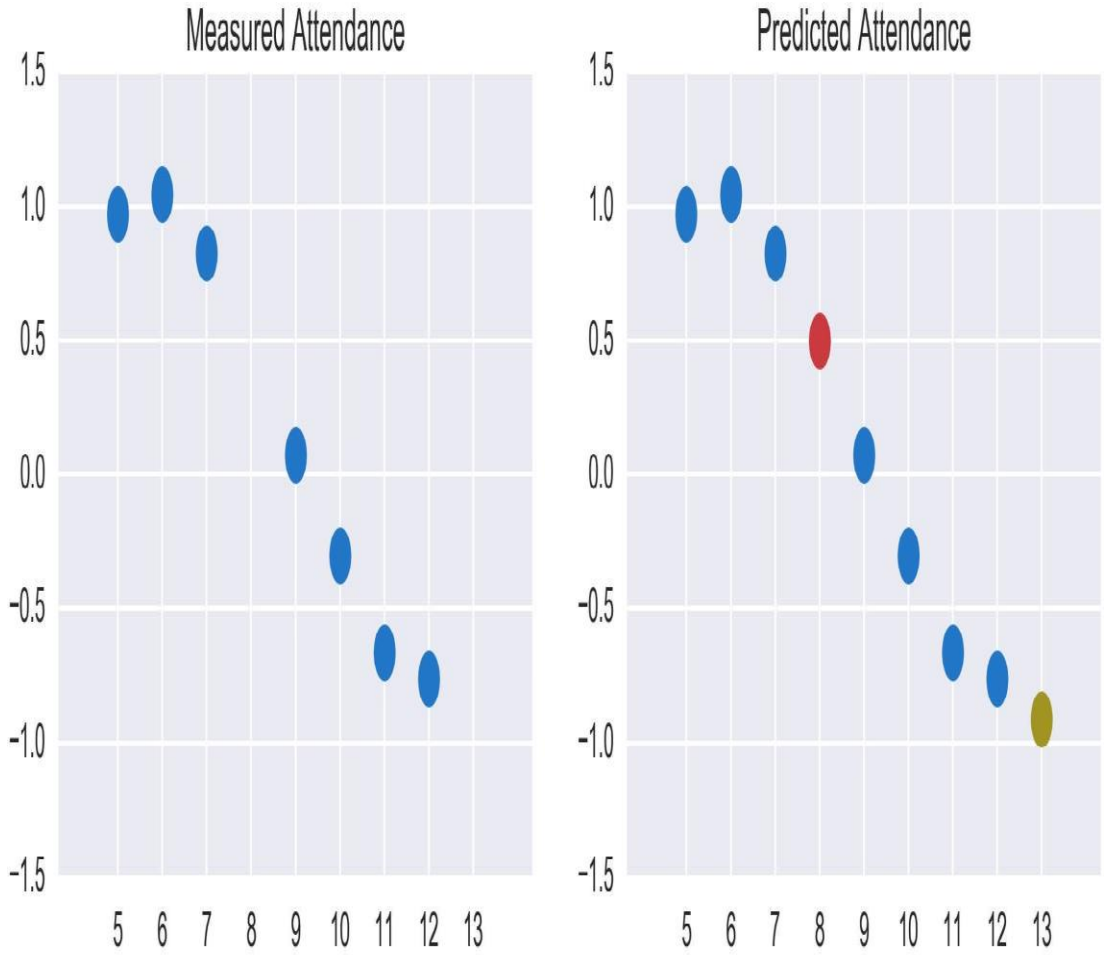
x : car attributes (e.g. mileage)
y : price



Regression Example

Example: Music band attendance

- Problem: An incomplete collection of measurements of attendance.
- Estimate missing values of attendance.
- Data: Attendance at a series of concerts at a local arena.
- Problem: Unfortunately, we lost count for one evening's performance.
- Also want to know what tomorrow's attendance is likely to be.



Regression Applications

- Economics/Finance: predict the value of a stock
- Epidemiology
- Car/plane navigation: angle of the steering wheel, acceleration
- Temporal trends: weather over time

Supervised learning: Ranking

label

1

4

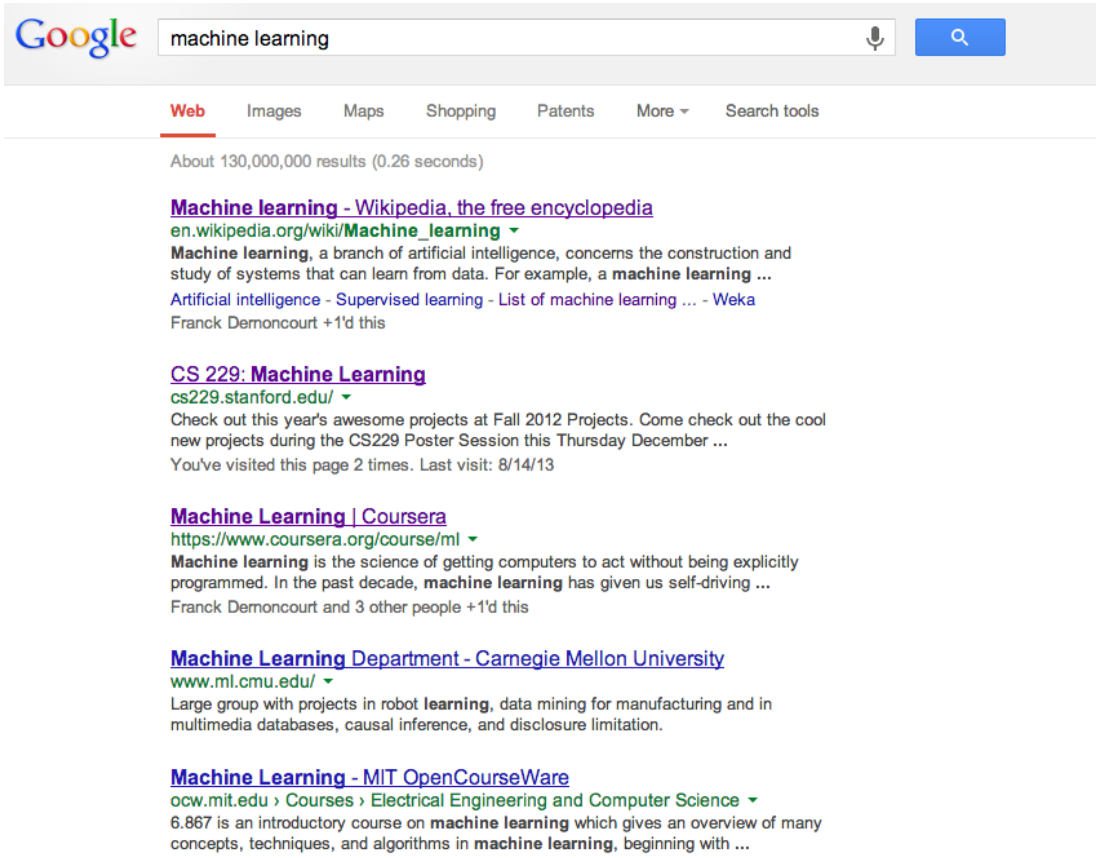
2

3

Ranking: label is a ranking

Ranking example

- **Given a query and a set of web pages, rank them according to relevance**



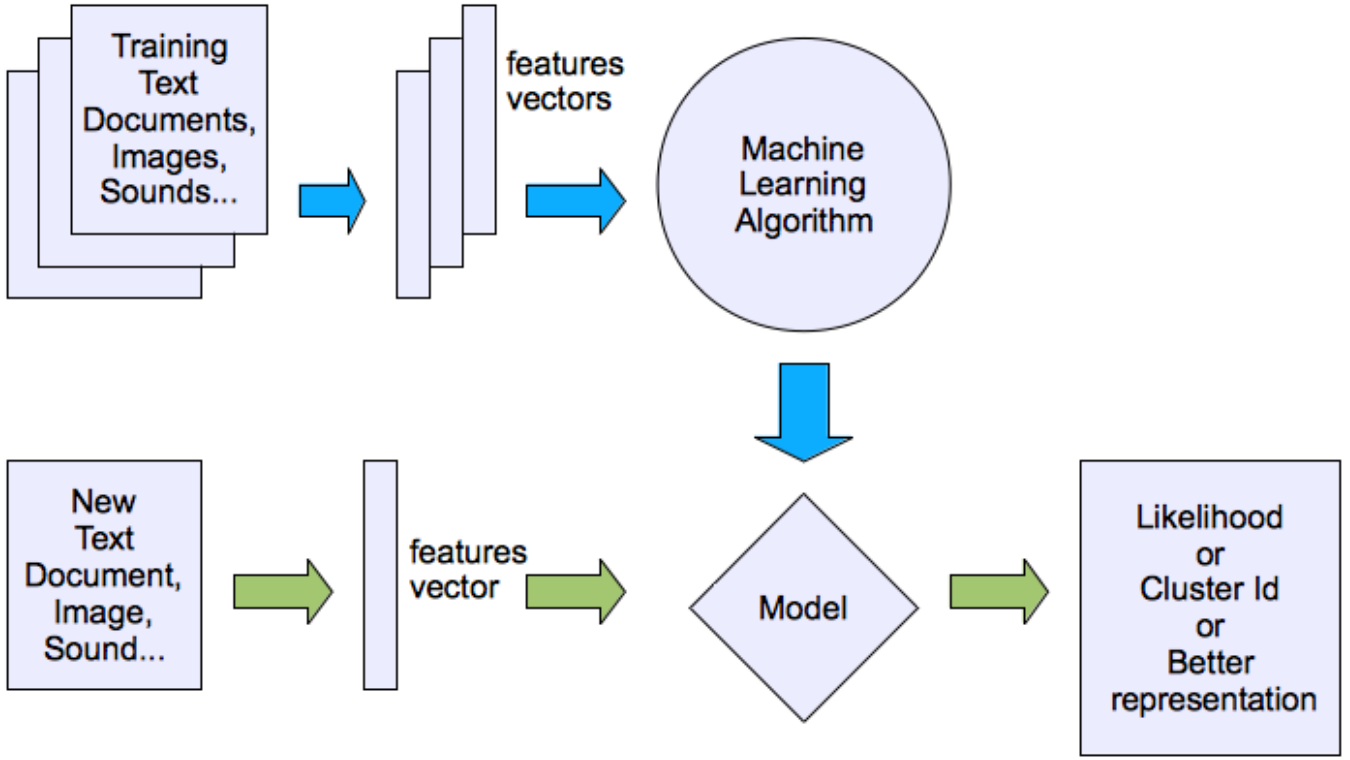
Ranking Applications

- User preference, e.g. Netflix “My List” -- movie queue ranking
- iTunes
- flight search (search in general)
- reranking N-best output lists

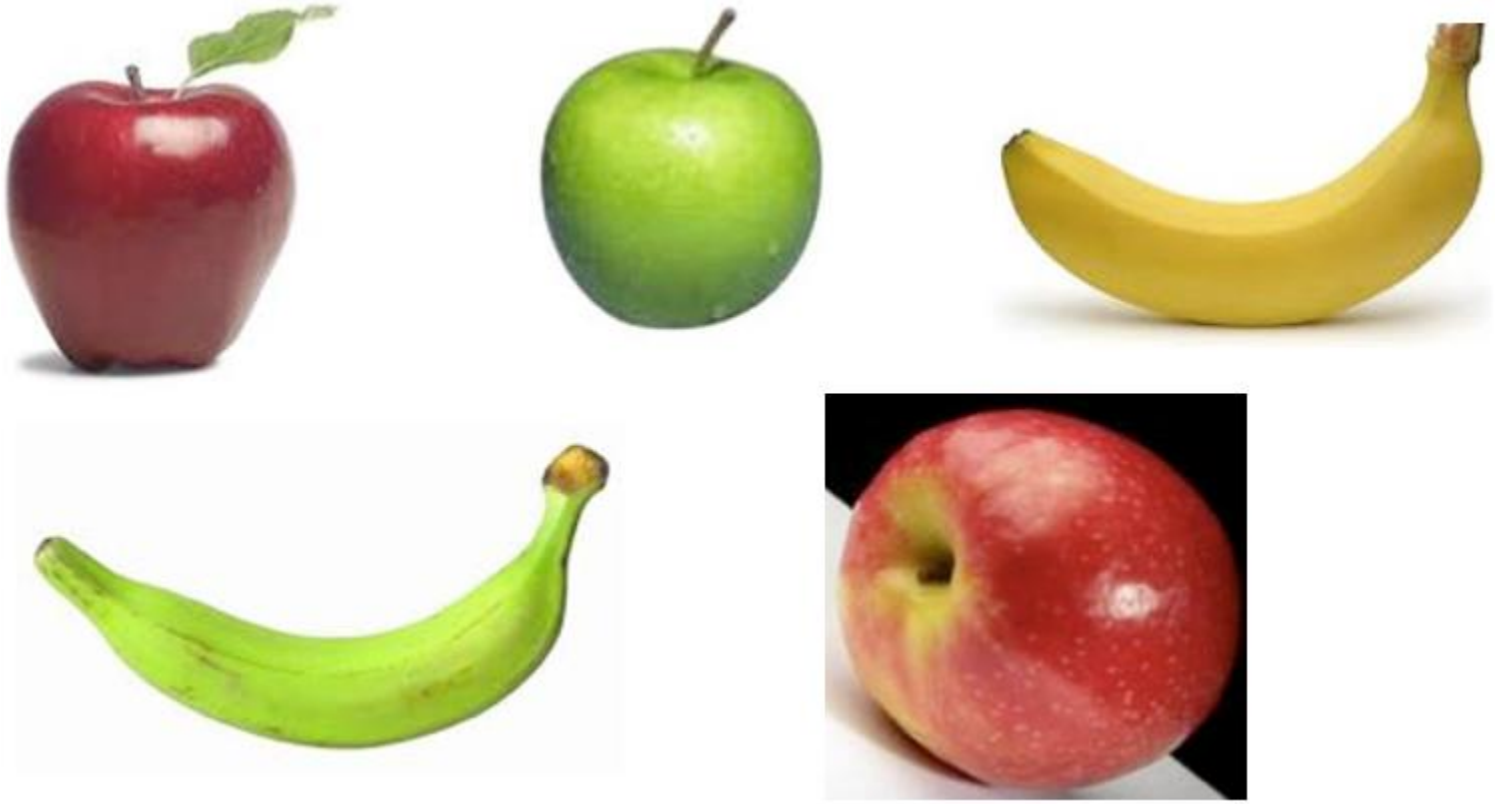
Unsupervised Learning (USL) – a form of ML

- When input data does not have labels, any algorithm that learns from the data belongs to USL.
- We are not "supervising" the learning process by offering labels.
- The system has to figure everything out on its own, with no help from us.
- USL is used for clustering, noise reduction, and dimension reduction.

Unsupervised learning



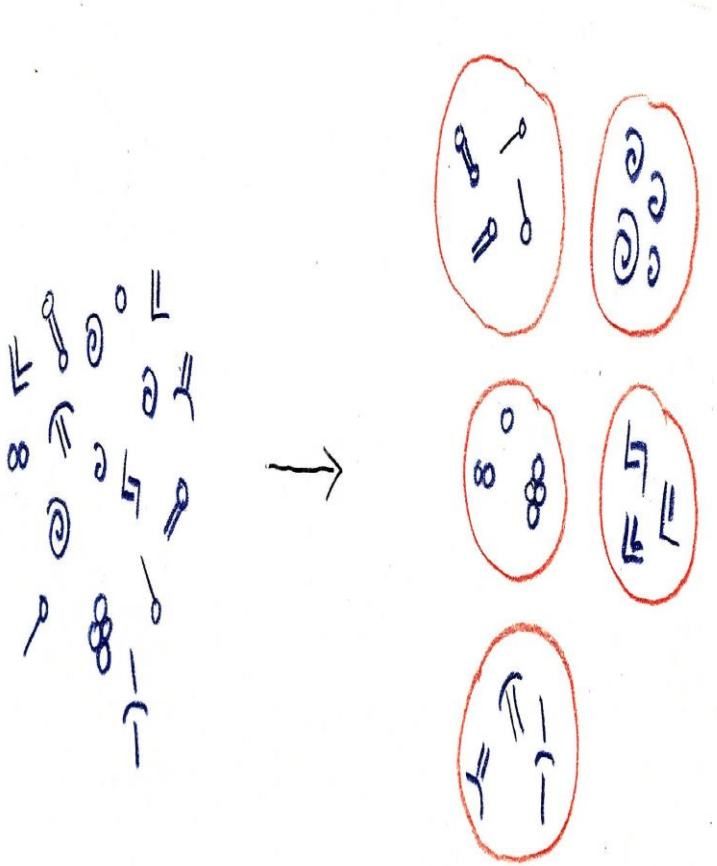
Unsupervised learning



Unsupervised learning: given data, i.e. examples, but no labels

USL forClustering - Via Pottery Example

- Suppose we're digging out foundation for a new house.
- Surprise! we find the ground is filled with old clay pots and vases.
- Call an archaeologist, who says its a jumbled collection of ancient pottery, from many different places and different times.
- Archaeologist doesn't recognize any of the markings and decorations, so she can't declare for sure where each one came from.
- Some marks look like variations on same theme, while others look like different symbols.



USL forClustering - Via Pottery Example

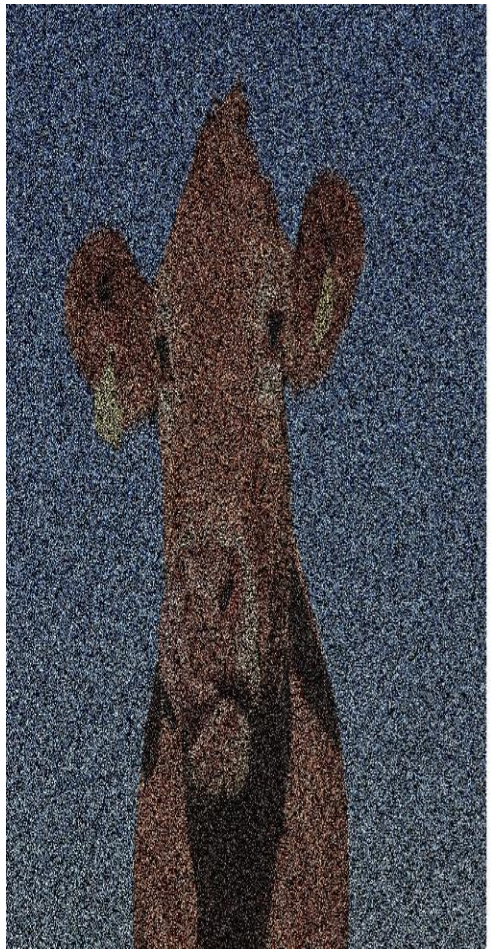
- She takes rubbings of the markings, and then tries to sort them into groups.
- But there are far too many of them for her to manage.
- She turns to a machine learning algorithm.

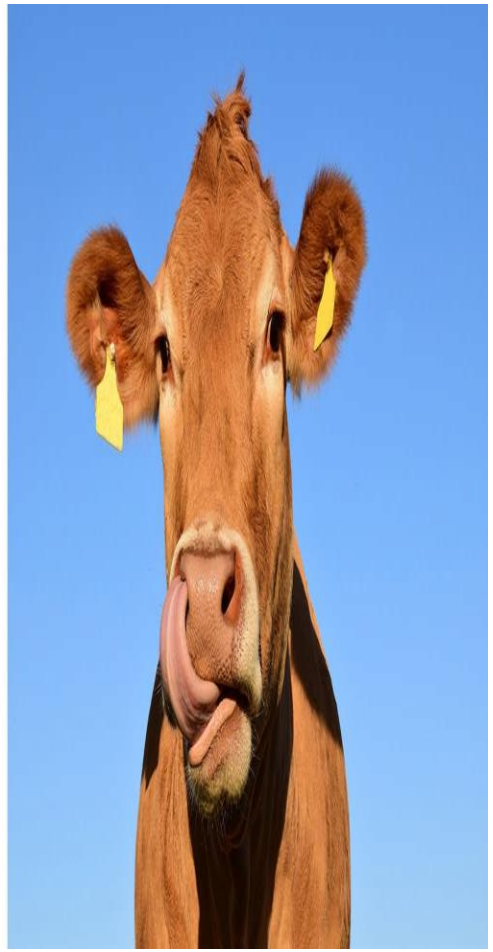
Why ML?

- To automatically group the markings together in a sensible way.
- On the right of previous figure, we show her captured marks, and the groupings that could be found automatically by an algorithm.
- This is a clustering problem
- The ML algorithm is a clustering algorithm.
- There are many clustering algorithms to choose from.
- Because our inputs are unlabeled, this archaeologist is performing clustering, using an unsupervised learning algorithm.

USL for Noise Reduction – Noisy Image Example

- Figure shows a noisy image, how a de-noising algorithm cleans it up.
- Why is de-noising a form of unsupervised learning (USL)?
- As we don't have labels for our data (for example, in a noisy photo we just have pixels)
- USL algorithm estimates what part of sample is noise & removes it.
- By removing weird and missing values from the input, learning process happens more quickly and smoothly.





USL for Dimensionality Reduction

- Problem: Sometimes our samples have more features than they need.
- So, simplify data:
- Remove uninformative features,
- Combine redundant features, or
- For these tasks, there are USL algorithms that can do the job.
- USL finds a way to reduce the number of features of our data - called as dimension reduction.

USL for Dimensionality Reduction Example#1 Weather

- Data: Weather samples in the desert at the height of summer.
- Record daily wind speed, wind direction, and rainfall.
- Given the season and locale, the rainfall value will be 0 in every sample.
- If we use these samples in a machine learning system, the computer will need to process and interpret this useless, constant piece of information with every sample.
- At best this would slow down the analysis.
- At worst it could affect the system's accuracy, because the computer would devote some of its finite resources of time and memory to trying to learn from this unchanging feature.

USL for Dimensionality Reduction Example#1 Weather

- Sometimes features contain redundant data.
- A health clinic might take everyone's weight in kilograms when they walk into the door. Then when a nurse takes them to an examination room, she measures their weight again but this time in pounds.
- Same information repeated twice, but it might be hard to recognize that because the values are different.
- Like the useless rainfall measurements, this redundancy will not work to our benefit

Unsupervised Learning techniques

- Unsupervised learning categories and techniques
  - Clustering
    - K-means clustering
    - Spectral clustering
  - Density Estimation
    - Gaussian mixture model (GMM)
    - Graphical models
  - Dimensionality reduction
    - Principal component analysis (PCA)
    - Factor analysis

Unsupervised learning applications

- learn clusters/groups without any label
- customer segmentation (i.e. grouping)
- image compression
- bioinformatics: learn motifs
- Face detection
- Object detection and recognition
- Image segmentation
- Multimedia event detection
- Economical and commercial usage

Semi- Supervised Learning (Generators)

- This process of data generation is implemented by ML algorithms called generators.
- Train generators with large numbers of examples- so that they can produce new versions with lots of variation.
- We don't need labels to train generators, so its unsupervised learning techniques.
- But we do give generators some feedback as they're learning, so they know if they're making good enough fakes for us or not.
- A generator is in the middle ground. It doesn't have labels, but it is getting some feedback from us. We call this middle ground semi-supervised learning.

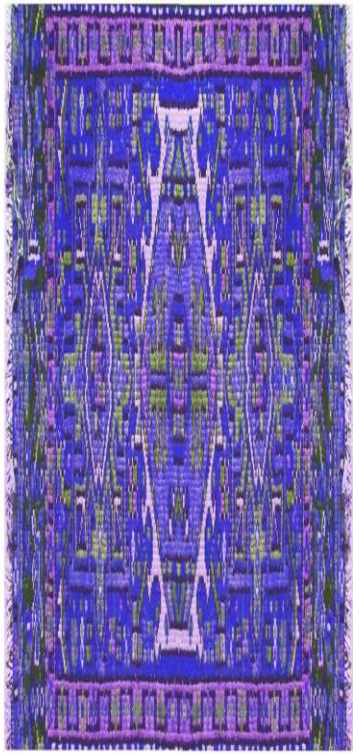
Semi-Supervised Learning (Generators) Example - Persian Carpets in Movie

- Problem: Want hundreds of carpets, all over the warehouse.
- Carpets on the floors, carpets on the walls, and carpets in great racks in the middle of the space.
- Each carpet to look real, but be different from all the others.
- Our budget is nowhere near big enough to buy, or even borrow, hundreds of unique carpets.
- So instead, we buy just a few carpets, and then we give them to our props department to make many fake carpets.
- Figure shows a Persian carpet that we'd like to generalize.

Semi-Supervised Learning (Generators) Example - Persian Carpets in Movie

- Figure shows a Persian carpet that we'd like to generalize.

- Figure shows some new fake carpets based on the starting image of previous figure made by ML Algorithms.



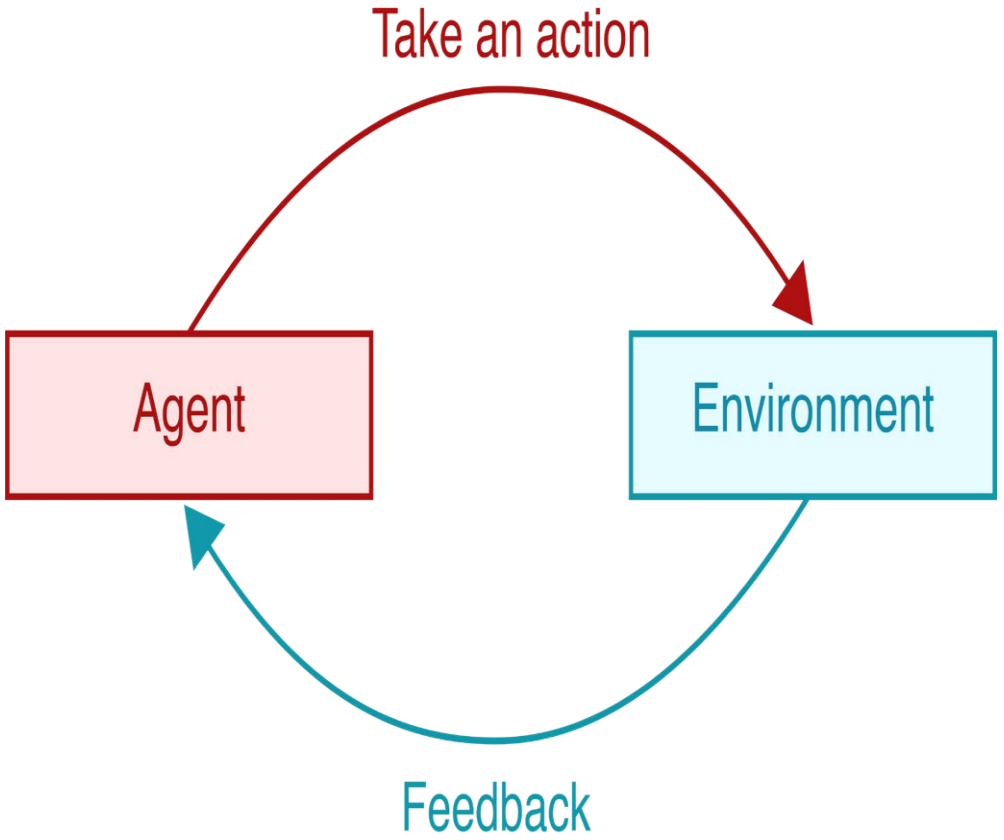
Reinforcement Learning

- Agent makes decisions and takes actions (the chef).
- Environment is everything else in the universe (the child).
- Environment gives feedback or a reward signal to agent after every action.
- Feedback tells how good or bad that action is.
- The reward signal is often just a single number, where larger positive numbers mean the action was considered better, while more negative numbers can be seen as punishments.
- The reward signal is not a label, nor a pointer to a specific kind of "correct answer."

Reinforcement Learning (Contd.)

In reinforcement learning,

- An agent (who acts)
- An environment (everything except agent).
- The agent acts, and the environment responds by sending feedback in the form of a reward signal.



How's RL different from SL?

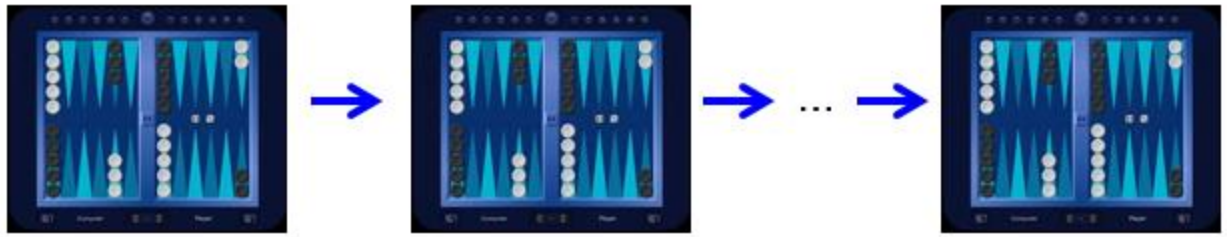
- The general plan of learning from mistakes is the same, but the mechanism is different.
- Supervised learning: system produces a result (typically a category or a predicted value), and then we compare it to the correct result, which we provide.
- Reinforcement learning: There is no correct result. The data has no label.
- There's just feedback that tells us how well we're doing.
- Feedback tells that our action was "good" or "bad."
- In contrast to supervised learning algorithms, the reward signal is not a label and no idea/pointer to "correct answer."

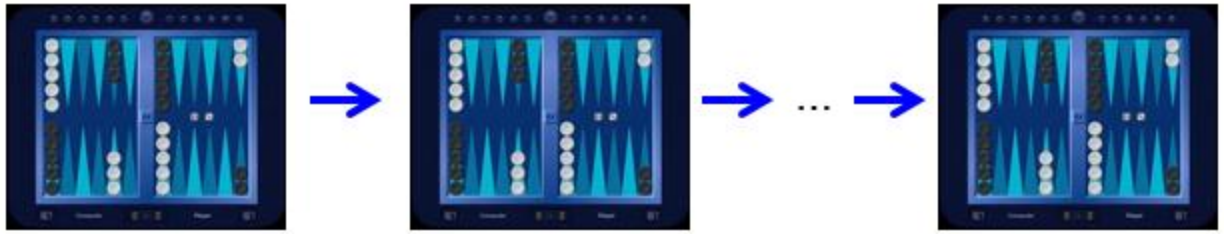
Reinforcement learning-Example

left, right, straight, left, left, left, straight	GOOD
left, straight, straight, left, right, straight, straight	BAD
left, right, straight, left, left, left, straight	18.5
left, straight, straight, left, right, straight, straight	-3

- **Given a *sequence* of examples/states and a *reward* after completing that sequence, learn to predict the action to take in for an individual example/state**

Reinforcement learning example





- **Given sequences of moves and whether or not the player won at the end, learn to make good moves**

Reinforcement Learning - Example

- Suppose you are taking care of a friend's three-year old daughter.
- You have no idea what the young girl likes to eat.
- First dinner: Make Pasta with butter. She likes it!
- Repeat this dinner for a week. She gets bored.
- Week 2: Add some cheese, and she likes it.
- Repeat this dinner for week 2. She gets bored.
- Week 3: Try pesto sauce. But girl refuses to take a bite.
- So pasta + marinara sauce , and she rejects that too.
- Frustrated,you make a baked potato with cream. She likes it!
- Weeks 3 & 4 : Try one recipe and one variation after another, trying to develop a menu that the child will enjoy.
- Only feedback: Little girl eats the meal, or she doesn't.
- Approach to learning is Reinforcement Learning !

Reinforcement Learning - Example

- Agent: Autonomous car
- Environment: Traffic/people on the street.
- Actions: Driving
- Feedback: Driving okay if following traffic rules and keep everybody safe.

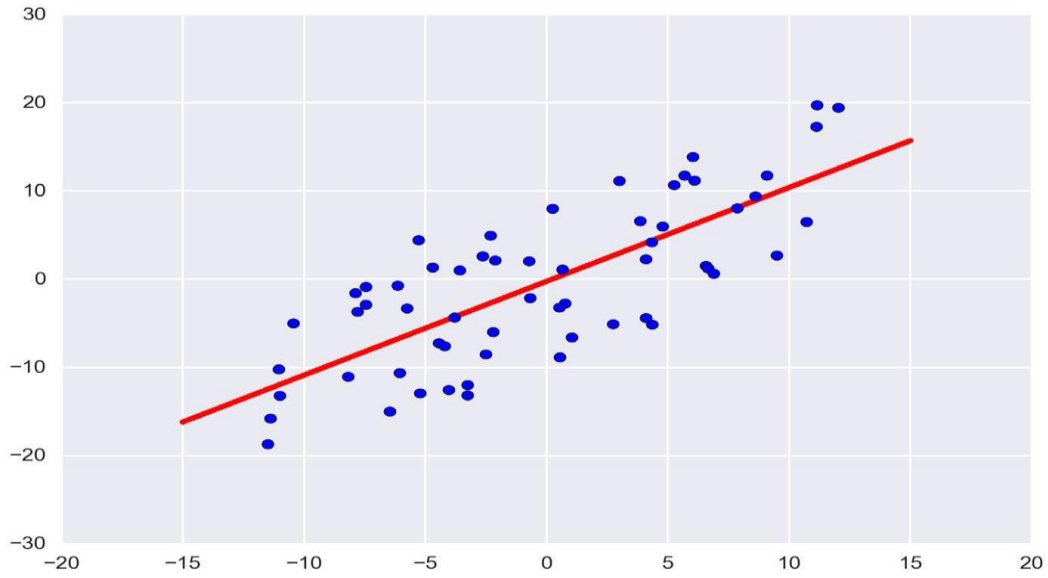
- Agent: DJ at a dance club
- Environment : Dancers
- Feedback: Like or dislike the music.

Deep learning

- More recently, the phrase deep learning was coined to refer to
  - Approaches to machine learning that use specialized layers of computation, stacked up one after the next.
- This makes a "deep" structure, like a stack of pancakes.
- Deep learning (DL) refers to the nature of the system we create, not to any particular algorithm.
- DL really refers to this particular style or approach to machine learning (ML).

Compare ML & DL - Fit the best line

- ML: Find the best straight line through a bunch of data points, see Figure.
- DL: Given a set of data points (in blue), we can imagine a straightforward algorithm that computes the best straight line (in red) through those points.



ML vs DL – Line fitting example

- ML: Represent a straight line with just a few parameters.
- Use some formula to compute parameter values, given the input data points.
- This is a familiar algorithm, uses analysis to find the best way to solve a problem.
- Strategy used by many ML algorithms.

- DL: Don't know how to directly calculate the right answer, so we build a system that can figure out how to do that itself.
- For DL, we create an algorithm that can work out its own answers, rather than implementing a known process that directly yields an answer
- DL learns slowly. Every time program sees a new piece of data, it improves its own parameters
- DL ultimately finds a set of good values.
- Much more open-ended than the one that fits a straight line.

Heart of DL success

- What's a main reason for the enormous success of deep learning algorithms?

- DL success is due to the programs that find their own answers (apart from availability of high performance computing power).

Evaluation Metrics

Table: A sample test set with model predictions.

ID	Target	Pred.	Outcome	ID	Target	Pred.	Outcome
1	spam	ham	FN	11	ham	ham	TN
2	spam	ham	FN	12	spam	ham	FN
3	ham	ham	TN	13	ham	ham	TN
4	spam	spam	TP	14	ham	ham	TN
5	ham	ham	TN	15	ham	ham	TN
6	spam	spam	TP	16	ham	ham	TN
7	ham	ham	TN	17	ham	spam	FP
8	spam	spam	TP	18	spam	spam	TP
9	spam	spam	TP	19	ham	ham	TN
10	spam	spam	TP	20	ham	spam	FP

Misclassification rate

$$\text{Misclassification rate} = \frac{\text{Number incorrect predictions}}{\text{Total Predictions}}$$

$$\text{Misclassification rate} = \frac{(2+3)}{(6+9+2+3)} = 0.25$$

Binary Prediction

For binary prediction problems there are 4 possible outcomes:

- **True Positive (TP)**
- **True Negative (TN)**
- **False Positive (FP)**
- **False Negative (FN)**

The structure of a confusion matrix

		Prediction	
		positive	negative
Target	positive	*TP*	*FN*
	negative	*FP*	*TN*

A confusion matrix for the set of predictions shown in Table

		Prediction	
		'spam'	*'ham'*
Target	*'spam'*	6	3
	'ham'	2	9

Misclassification accuracy

$$\text{classification accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

$$\text{classification accuracy} = \frac{(6 + 9)}{(6 + 9 + 2 + 3)} = 0.75$$

Classification accuracy

$$\text{misclassification accuracy} = \frac{(FP + FN)}{(TP + TN + FP + FN)}$$

$$\text{misclassification accuracy} = \frac{(2 + 3)}{(6 + 9 + 2 + 3)} = 0.25$$

Thank You