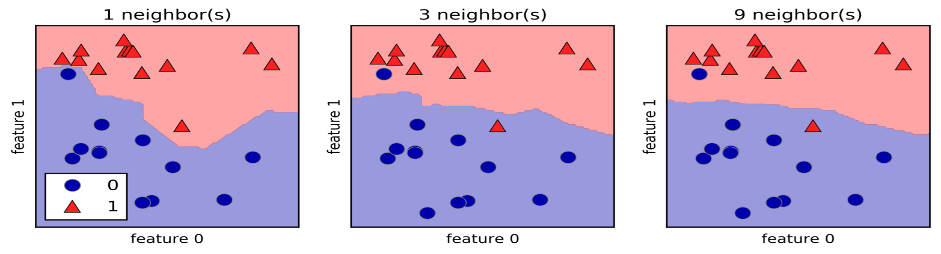
**Data Leakage** = When the data you're using to train contains information about what you're trying to predict.Obvious examples: Including the label to be predicted as a feature, Including test data with training data

A screenshot of a cell phone

Description generated with very high confidence**A close up of a necklace

Description generated with very high confidence**Linear regression makes a prediction by simply computing a weighted sum of the input features, plus a constant called bias term. **Minimize cost function like RMSE , MSE or SSE**

**GRADIENT DESCENT:** Tweak iteratively until the cost function is at its lowest.

**Use small learning rate so it can move nicely.** Or else the with big learning steps it will be all over the place and may never reach the global or local minimum. 0.1 is supposed to be good.

**A close up of a map

Description generated with very high confidenceMAKE DATA SCALED with standard scaler so that cost function can be minimized** nicely. THERE ARE 3 kinds of gradient descent based on the amount of data they use

**BATCH GRADIENT DESCENT:** Uses all the training sets and uses partial derivatives to calculate delta

**STOCHASTIC:** Uses random instances but finally reached its goals

**Mini BATCH** : computes the gradient on small random sets of instances called mini batches

**POLYNOMIAL:** To capture interactions between the original features by adding them as features to the linear model. To make a classification problem easier, we can apply other non-linear transformations to create new features.(Technically, these are called non-linear basis functions).Beware of polynomial feature expansion with high order as this can lead to complex models that overfit

**Problem of large coefficient:** When it becomes too large, the algorithm starts modelling intricate relations to estimate the output and ends up overfitting to the particular training data.

In order to overcome the issue we use model regularization by minimizing the coefficients.

**RIDGE REGRESSION**: uses alpha parameter to fine tune. At 0 they will be same as LM. At infinity they will all be 0. But somewhere lies a value that minimizes the MSE. Ridge prevents overfitting. Great to tackle X correlation

**LASSO REGRESSION:** Minimizes the value of coefficients and sometimes deletes them entirely. Could be a great tool for feature extraction. Going to be a great model for large number of X variables.

**LOGISTIC REGRESSION:** you have decision boundary and around that you have probability lines.

**SOFTMAX:** can do multiple classes but target is to have higher probability for target class but low for other.

**CROSS ENTROPY:** measures the average number of bits needed to identify an event drawn from the set of events.  Remember weather example. We knew sunny is target so we made it 0 and added 1 to all other categories. A screenshot of a cell phone

Description generated with very high confidence

**SVM.** Uses a vector with padding on both sides. Data must be scaled for this. Like a street going thru classes. Hard margin for outside of padding and soft margin if training sets are allowed in padding. Control d and you control

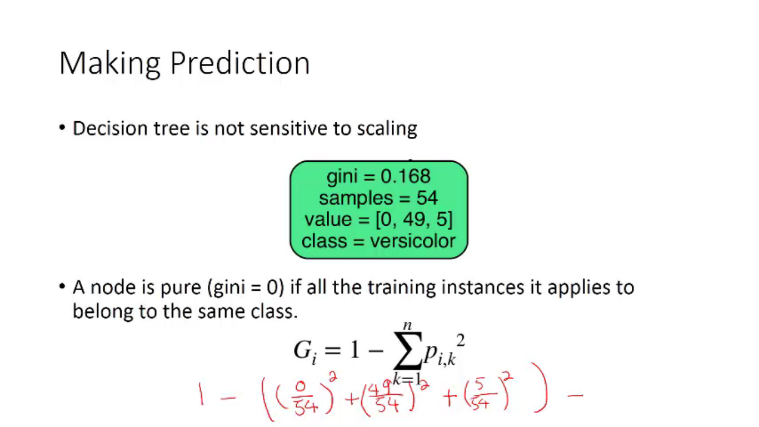
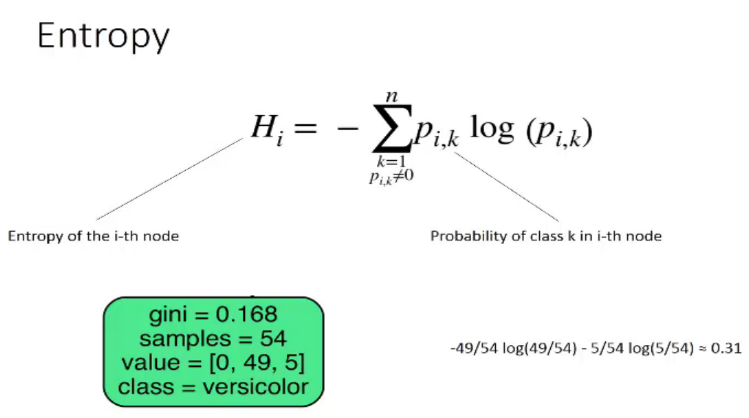
A close up of a piece of paper

Description generated with very high confidenceSo reduce **C** if its overfitting and **give it greater flexibility** and increase C if its underfitting and make its concise

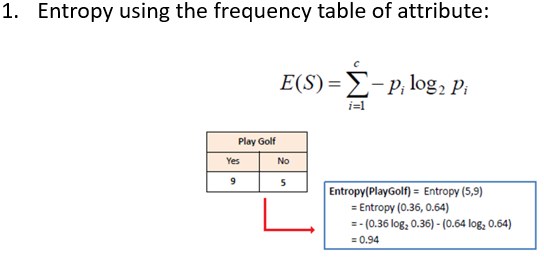
A close up of a logo

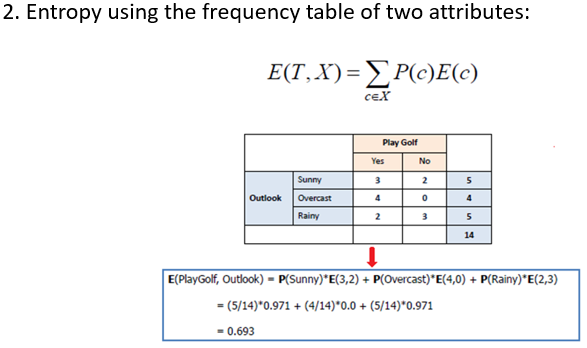
Description generated with high confidenceif model is **overfitting reduce Gamma value**. If its underfitting then increase the value of Gamma

**DECISION TREE**: They are not sensitive to scaling. **GINI** is scale of impurity so 0 means its pure. All column had only one value.



Gini computes faster. But Entropy makes more balanced trees.

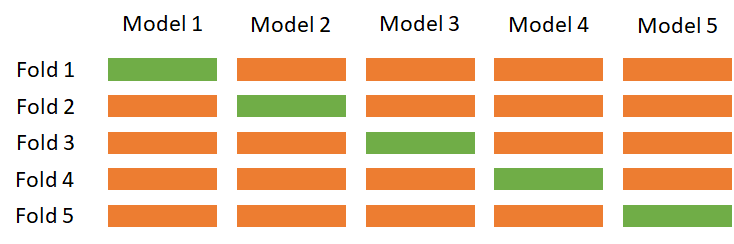




A screenshot of a cell phone

Description generated with very high confidence **We go for most info gain.** Meaning decrease in entropy after the split. You must find gain for each column ( or x variable) and see gain on each one of them. Split where you find the most gain. Once you choose then you repeat the exact same process and finally you will find a pure set. That is good and helps the decision tree. **T is for Target and X is the column we are checking for**

* **Model selection for classifiers: Cross validation** is more stable and thorough than a simple train/test split. We do that with different models. By default cv=3 meaning 3 folds. cross\_val\_score: returns accuracy values. In this we can control the structure of the train test data.

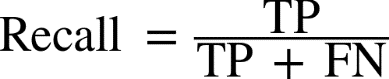
 STRATFIED K FOLD : scikit by default is stratified k fold meaning uniformly distributed so training is generalized well.

Leave one out is good too. Good for smaller datasets.

Grid Search \_ find your model on basic parameters and then find the best model on different sets of hypermeters.

Because we used it so much with test so we split in 3 areas. Train, test and validate finally on the validate.

* The best\_score\_ stores the mean cross-validation accuracy, with cross-validation performed on the training set.
* This is different from the score for evaluating the generalization of the model.



A picture containing furniture

Description generated with high confidenceA picture containing furniture

Description generated with high confidence Understand problem of imbalanced data. Fraud prevention. Only occurs about 0.1% so we can oversample to solve the problem.

Decision function, decision scores and probability is good for predict\_prob to understand the probability and then use threshold to finalize the model.

A close up of a map

Description generated with very high confidenceA close up of a map

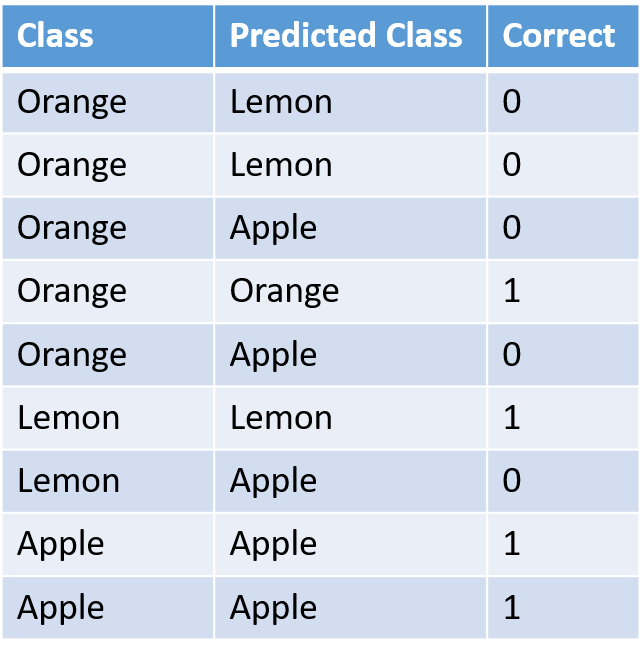
Description generated with very high confidence

The more curve at the top right of the screen the better. As it is precision on of 1 and recall of 1.

There is another tool called ROC Receiving Operating Characteristic

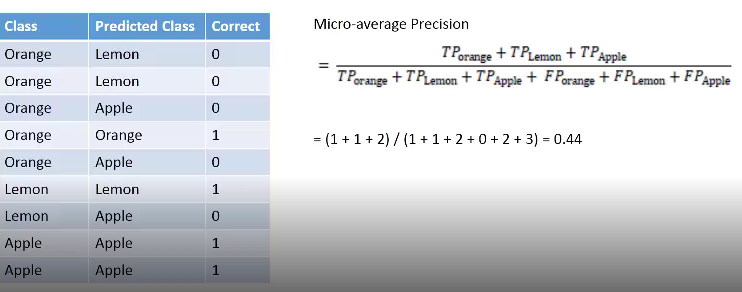
ROC is plot on FPR & Recall

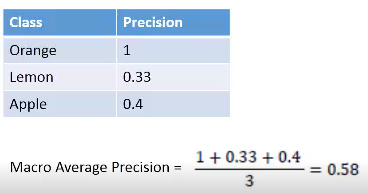
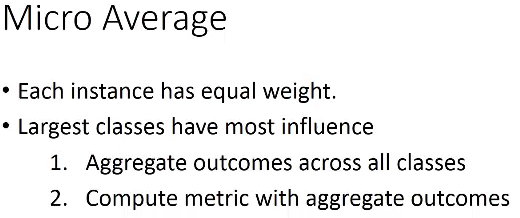
**FPR = FP /(FP +TN)**

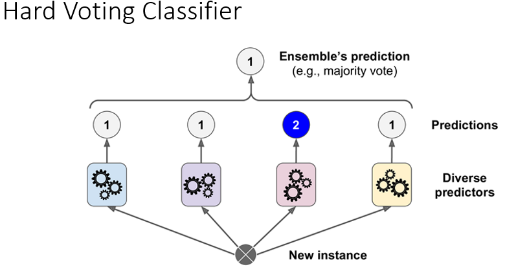
A close up of a map

Description generated with high confidencePython has We can compute the area under the ROC curve using the roc\_auc\_score function. We can refine TPR and decrease FPR with gamma. Look at the TPR & FPR at gamma = 0.01

For multiple class evaluation we have **macro (all classes together) and for just single class out we can use micro**

 In macro we first make a column which says correct prediction or not. This in return everything to 0 & 1 and now we can do metric

**Set up scoring argument in Grid Search**. This Will act like a model selection

**Ensemble. A group of predictors is called an *ensemble.***: train a group of Decision Tree classifiers, each on a different random subset of the training set. To make predictions, obtain the predictions of all individual trees, then predict the class that gets the most votes. ensemble of Decision Trees is called a Random Forest.

**Law of large numbers** when we use 1000 predictors so final predictions gets a lot better.

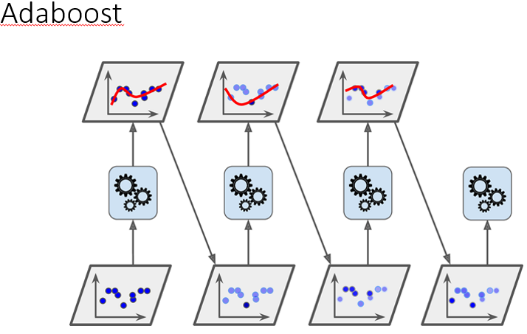
**Soft voting** Predict the class with the highest class probability, averaged over all the individual classifiers

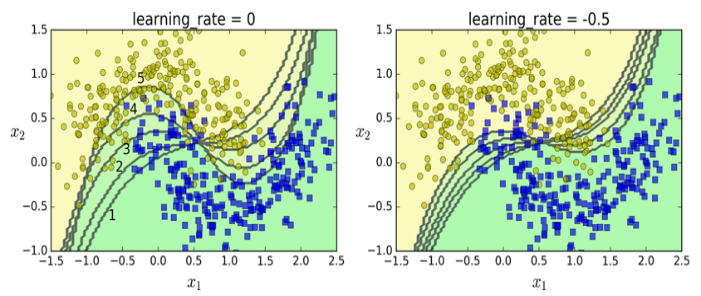
**BAGGING** uses same algorithm but uses different sampled datasets. When sampling is done with REPLACEMENT we call it bagging. Still bootstrapping but training sets are replaced. When sampling is performed without replacements then its called **PASTING**

***We prefer bagging over pasting as decision boundary is generalized well. In bagging classifier boostrap true means bagging and bootstrap false means pasting***

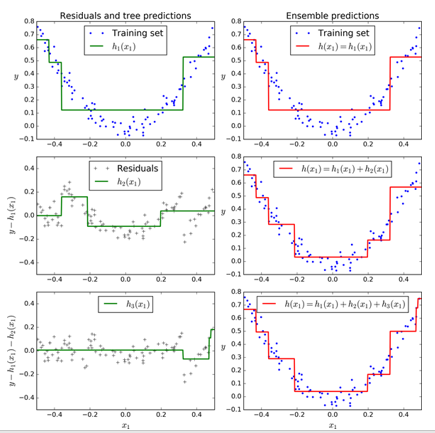
**OUT OF BAG evaluation** On average only 63% of the training instances are sampled in average. The remaining samples are called out-of-bag (oob) objects.

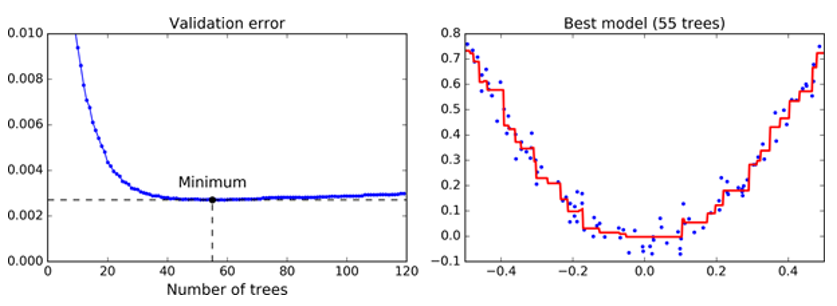
**Random subspace**: random sunset of feature set. **Random patch:** random subset of instance set…. So we select columns and rows.

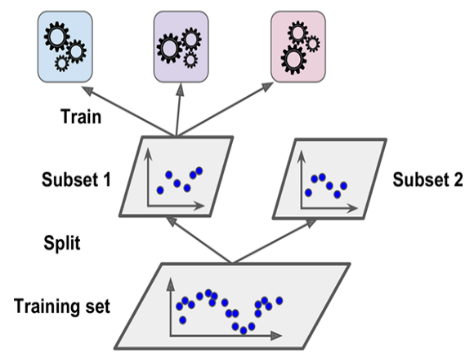
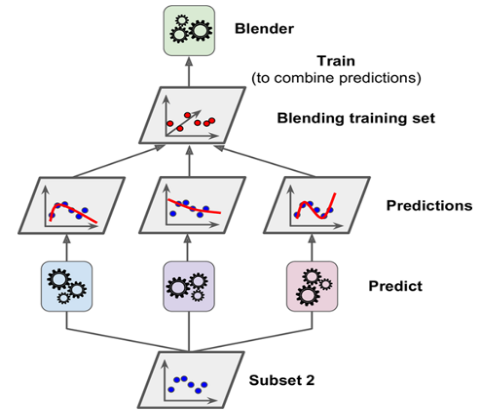
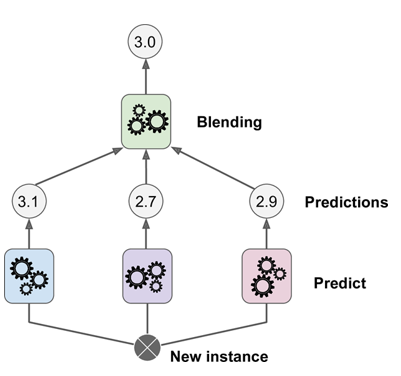
**Random forest**: An ensemble of decision trees, training via bagging (or pasting). **Feature Importance:** Measure the relative importance of each feature. How much the tree nodes that use that feature reduce impurity on average

We make learning rate -0.5 meaning that weights will be boosted half at every iterations for the ones that were classified wrong. In this sequential learning can overfit. Doesnot scale very well.

In gradient boosting we only do further iterations on the points that were misclassified. Compared to Adaptive boosting where we iterate over the whole thing.

with low learning rate you will need more estimators ( like 200). Goal is find best learning rate with number of estimators.

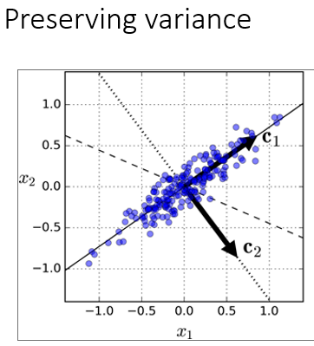


**Lets do Stacking** its an ensemble approach. Instead if using hard voting. We can train

**PCA :**  system will perform worse, you loose info. BUTTT training is faster and data visualization is great. **UNIT SQUARE EXAMPLE:** Take a unit square and divide it into 1000\*1000 blocks. Its probability to be near border is 4.0%.But if that is hyper cube ( 100 dimensions) then we have 99.9% probability that it will be near the border. **Sparse :** high dimension data is very sparse and will overfit so we must reduce the dimensionality.

*Principal Component Analysis* (PCA) is by far the most popular dimensionality reduction algorithm.

First it identifies the hyperplane that lies closest to the data, and then it projects the data onto it.It like to preserve the variance. It chooses its axes by minimizing the RMSE by looking at actual vs projected.

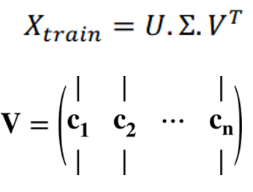


PCA identifies the axis that accounts for the largest amount of variance in the training set.

The unit vector that defines the ith axis is called the ith *principal component* (PC).

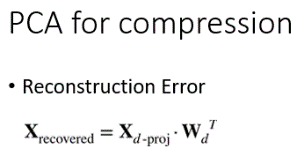
The direction of the principal components is not stable.

Singular value decomposition --- decompose training sets into dot product of 3 matrices.

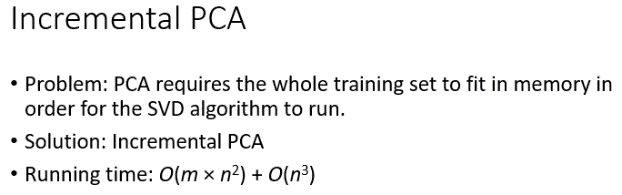
V is the most important matrix as it holds the principal component that we are looking for. PCA assumes that dataset is arranged surround origin. So centralize. Meaning subtract mean from every value. Or in scikit learn it happens naturally

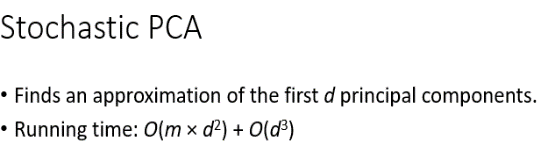
We use explained variance with elbow method to choose the best number of columns/features

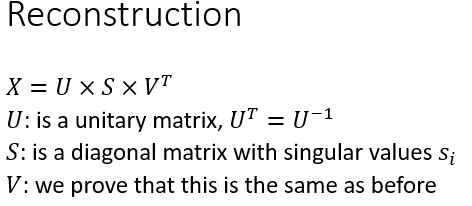
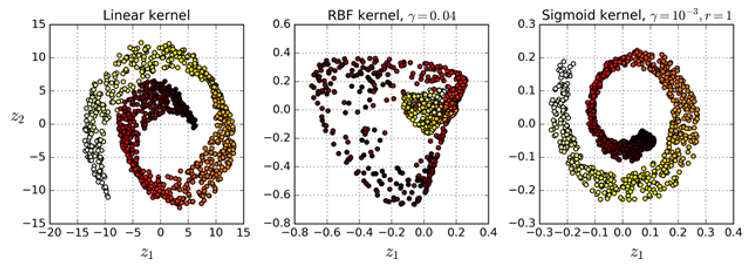
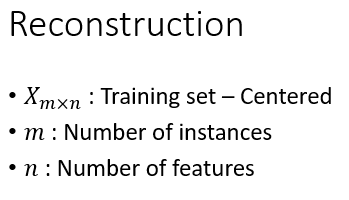
There is a reverse transformation from PCA back to original. PLEASE understand that we have lost some info and it won’t be created as before. But still very close. Use MNIST example



There is a loss and we call it reconstruction error. Mean squared distance between original & reconstructed data is called reconstruction error. X- the original X.

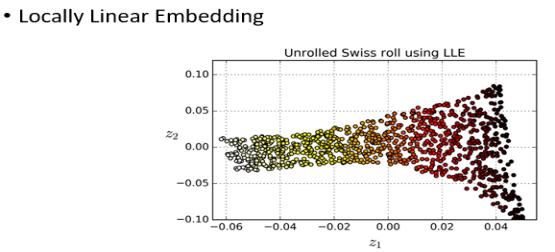
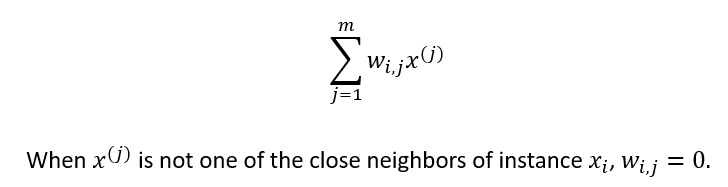


Stochastic picks random spots and reduces time significantly.



we saw swiss roll example. How RBF had better numbers but linear & Sigmoid restored the structure better

Best kernel is the one with lowest reconstruction error which is x – x(reconstructed)

**MANOFOLD LEARNING** This rolls out meaning flatness out the data on lower dimension. It does that thru KNN technique. In this one W is o for those Xi that are 

Not the nearest neighbor or part of those 10 N.

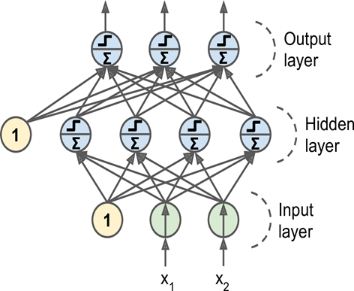
Second step is that Zi is the image of Xi in lower dimension. But nearest neghbors of Xi will remain its neighbors in lower dimension. This creates a powerful PCA. We don’t use LLE ( manifold) in large datasets.

A drawing of a face

Description generated with high confidence**Neurons** They receive two or more inputs and then fire an output.

***Hebb's rule:*** Cells that fire together, wire together.

Perceptrons donot give probabilities. They simply give hard threshold. Classification is built on linear equations only. Where w is weight



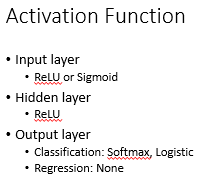
This is a dense setup. As every perceptron is connected with all the layers behind it and all the perceptron’s in front of it. Below is how Backward propagation works.

1. make a prediction (forward pass)
2. measure the error
3. go through each layer in reverse to measure the error contribution from each connection (reverse pass)
4. tweak the weights to reduce the error

We use Sigmoid and RELU functions to tweak. Their X AXIS is weights and y value is the value we will use. So tweaking is smoother and more continuous. RELU are getting more traction lately.

A close up of a clock

Description generated with high confidencethese Xi\*Wi +bias(its 1) and their sigma is sent thru RELU or Sigmoid. Then that’s a hard classification. More complex problem you can have more layers. There is a trial and error process but you have to know. Note that w gets tweaked which is contributing the highest error and that is how we refine our results. The input data ( number of features) will decide the # of input layers and form a funnel



For MNIST data set where we have to do multiclassification We used Softmax. So we produce a vector

Compile model could be 1. OPTIMIZER 2. Loss function 3. Evaluation method

* epochs: number of times the model is exposed to the training set. At each iteration, the optimizer tries to adjust the weights so that the objective function is minimized.
* batch\_size: number of training instances observed before the optimizer performs a weight update.