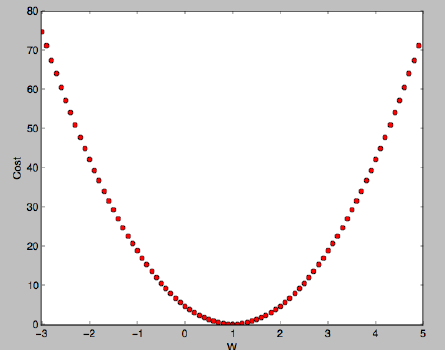
**Introduction**

* Gradient is derivative of function that has more than one input var known as slope of function measures change in all weights with regard to change in error.

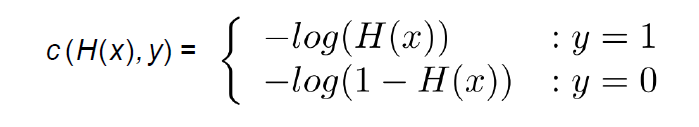
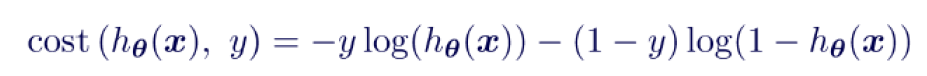
**Linear Regression**

* Supervised
* LR with one variable = univariate LR
* Cost to fit the line = MSE and hypothesis is Y=mx+c
* Cost function is like convex function
* To minimize the cost gradient descent algorithm can be used ..for a given cost function gradient descent algorithm finds m and c to minimize the cost
* Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. In machine learning , we use gradient descent to update the param of model
* For multivariate LR Y = w1x1 + w2x2 + w3x3 + b can be thought of as a matrix multiplication H = XW
* Data PreProcessing (to make sure features are on same scale and only one feature is not getting more attention) = Centering, Normalizing, Whitening
* Centering = Subtracting mean from every x value
* Normalizing = min-max [0,1] , z-score (Standardization)
* Lock away test set until your final preprocessing hypothesis is ready
* Precision = TP / TP+FP
* Recall = Sensitivity = TP / TP + FN
* Accuracy is imp when TP and TN are imp F1Score (higher the better) is used when FP and FN are imp
* Handle missing values : Replace with most frequent/mean/median/constant/zero, using KNN
* Evaluating model : MSE and Coefficient of determination (R2 = 1 – RSS/TSS) [0,1] 1 is best

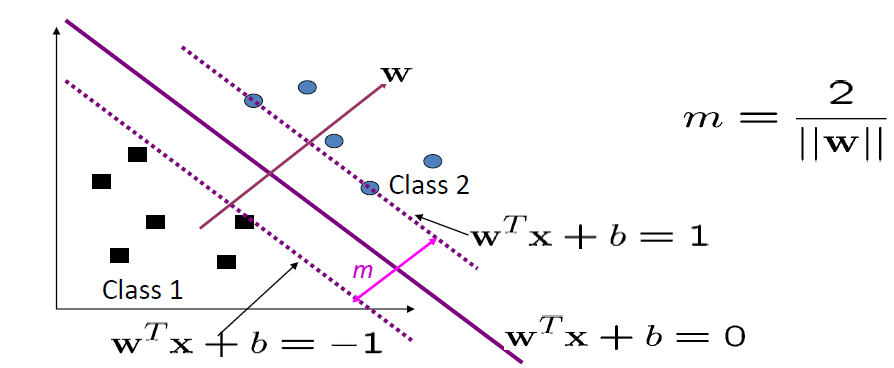
**Regularization**

* Bias are assumptions made by model. **High bias** oversimplifies the model (**underfitting**). High error on training and test set
* **High variance (overfitting)** does not generalize the data. Less error on training data high error on test data
* Regularization is solution for overfitting.
* Which means whenever model is trying to add new features into the model a penalty is added in the MSE.
* Lasso Regression (L1 Regularization) : Will add absolute theta value in error …theta=slope
* Ridge Regression (L2 Reg) : Will add summation of squares of theta in error
* Ridge regr is more useful when all the variables are not useful
* Feature Selection methods : Filter methods, Wrapper methods
* Wrapper methods : select subset of features, build ML model, evaluate model performance, Repeat.
* For selecting subset we can use **forward** feature selection (start with 0 keep adding), **backward** elimination (start with all keep eliminating), **exhaustive** (tries every combination), **bidirectional** (does frwd and bckwd to find one unique solution)

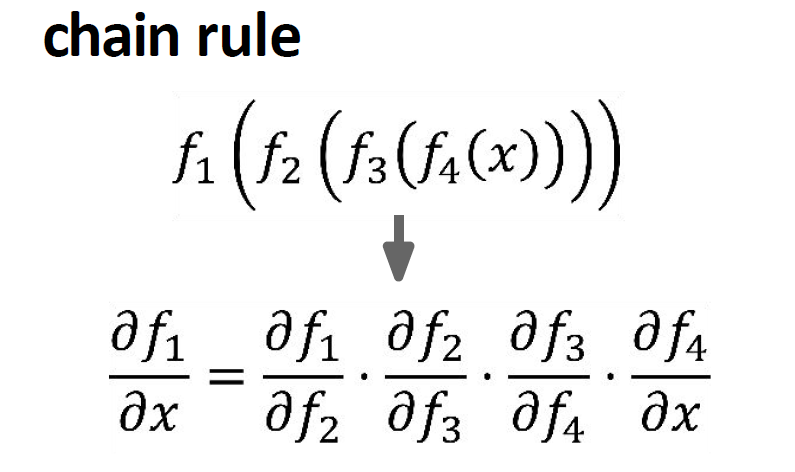
**Logistic Regression**

* Dependent variable is categorical
* Independent variable can be continuous or binary
* Linear regression can be used when relationship between I and D is linear, logistic is used irrespective of type of relationship
* It calculates a probability a record belongs to a particular class.
* Cost function 
* Minimize  - gradient descent problem
* Softmax is used for multi-classification in logistic regression (there is one right answer-multi class)
* Sigmoid for binary (more than 1 right – multi label)

**SVM**

* Classification problem, not necessary Linear separable data
* Tries to maximize the margin between two classes
* Low regularization in SVM prevents overfitting.
* Support vectors are those datapoints that margin pushes up against
* 
* Kernel trick means transforming data into another dimension that has clear dividing margin between classes
* Polynomial or RBF kernel is a good initial try

**Neural Network**

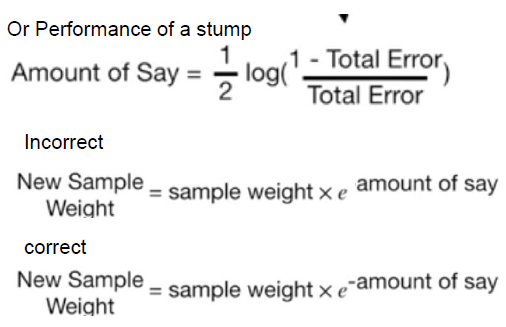
* Activation functions : Sigmoid, Tanh, RELU, Leaky RELU, ELU, Maxout
* Weight matrix with dimension = sj \* sj+1 and has sj+1 biases
* Theta is network parameter
* Backpropagation chain rule 
* Regularization in NN = Dropout = Dropping random neurons to prevent overfitting
* Steps is training NN

1. Do forward prop, calculate sum and activation of each neuron by iteratively doing matrix vector multiplication and taking component wise transfer function of all neurons in every layer.
2. Calculate error signal of final layer
3. Back propagate the error
4. Calculate derivative of cost function wrt weights, activation of each neuron is required to do this
5. Calculate deri of cost function wrt biases (can be skipped)
6. Update weights acc to gradient descent rule

**Decision Trees**

* Greedy, top-down, recursive partitioning
* Classifies items by posing a series of questions about features associated with items
* Pruning = when we remove subnodes of a node
* DT represent a disjunction of conjunction of constraints on attribute values
* Purer classes must be chosen as node (Maximize gain)
* Entropy = measure of homogeneity of set of examples, it measures purity
* Entropy = 0 when output is certain, and = log(n) when any outcome is equally possible
* Stopping condition : every attribute is included, training egs of a leaf node all have same target attr value
* A branch with entropy = 0 is leaf node
* Gini index is computationally efficient
* Gini index works with categorical target var, performs only binary splits, high gini high homogeneity, CART uses gini.
* Prevent Overfitting : Grow full tree and prune, stop growing tree when split does not yield an improvement
* Bagging = Sample dataset with replacement, parallel computing
* Boosting = Sample dataset with replacement over weighted data, serial

**Boosting**

* Algorithm to reduce bias and variance
* **AdaBoost :** misclassified samples receive higher weight(more attention), minimizes the upper bound of the training error
* Total error = summation of misclassified weights
* 
* Stump = weak learners
* Each model is built on subset of data
* **Gradient Boosting :** errors are minimized by gradient descent algo, using loss function, weak learner.
* **Algo:**

1. Compute pseudo-residual
2. Fit base learner on pseudo residuals
3. Compute multilayer something by solving optimization problem
4. Update model

Asan basha mei:

1. Calculate avg of target var and calc residual
2. DT to predict residual
3. Predict target var using residual = Avg. Price + lambda (residual from DT)
4. Compute new residual = Actual-Predicted
5. Repeat 2-4

* **XGBoost :** Gradient boosting with parallelization, scalable, efficient, small dataset may perform good with RF. Has built in regularization.
* **Pros :** parallel tree building, DFA, CV, Regularization, handles missing data
* **Light GBM :** high accuracy and speed, lower memory usage, compatible with large datasets, Equal good as XG but less time.

**KNN**

* Supervised
* Works only with numerical attributes
* Ways to choose k : K=sqrt(N) or cross validation or odd number
* Make sure features are in same scale hence normalize them
* Strengths : Simple and intuitive, good classification if samples are large enough
* Weakness : more time, choose k, need to calculate and compare distance from new example to all other examples

**KMeans**

* Unsupervised learning algo
* Partitonal (determine all the clusters at once), Centroid based
* Convergence criterion : no or minimum assignments of data points, no or minimum change of centroids, minimum decrease in SSE
* Strengths : easy to understand and implement, O(tkn) since both t and k are small it is considered as a linear algo
* Weakness : Mean is defined only for numerical data, for categorical data mode (most frequent) value is used. Sensitive to outliers, choose K
* To avoid outliers : Remove data points that are much further away from centroids, Perform random sampling
* It is sensitive to initial centroid chosen

**Recommender System**

* **Content** – Assume access to side information. New item is not problem, item profile is created
* **Collaborative** – Does not have access to side info, does not work on new items, **USER BEHAVIOUR,** similar users like similar movie
* **User Profiles** : Binary, percentage, weighted average
* Collaborative suffers from overspecialization, difficult to find appropriate features to represent items
* Two types : Neighborhood methods, Latent factor methods (create a latent space and see where the user lies)
* Matrix Factorization : A matrix can be decomposed into unique combination of matrices

**NLP**

* Sub-field of AI
* Analyzing and representing naturally occurring texts/speech at one or more levels
* Term document count matrices – number of occurrences of a term in a document (imp word occurring only once will be ignored)
* Bag Of Words = CountVectorizer – Create vectors with count of words for each document ….Con = Ordering of words is lost, we don’t know which word is more important
* TFIDF = How imp a word is to a document.
* TF = rep of words in sentence/words in sentence , IDF = log(no of sentences/no of sentences containing words)
* Word2Vec = NN model to learn word associations. Represents each word as vector. Embeddings are not hand crafted, they are learnt during NN training. Used for next word prediction. Cons = quality depends on input data, number of samples and size of vectors (long computational time)
* GloVe = Global vector for word representation : calculate pairwise relationship. Use a sliding window.

**PCA**

* Curse of dimensionality : Redundant, non corelated features are also considered
* The number of training examples required increases exponentially with dimensionality
* Feature Extraction method
* Find projection on the PC line
* Take the data, calculate covariance matrix, find eigen vectors of cov matrix, find eigen values of those eigen vector. Those eigen vectors orthogonal to each other are PC that capture most variance
* First Center the data then apply SVD
* Reduce memory/disk needed, speed up learning algo, visualization is easy
* PCA decorrelates the data
* Limitation : Does not consider class separability since it does not take into account class info, no guarantee that direction of max variance will contain good features

**CNN**

* Disadvantage of ANN in image classification : too much computation, sensitive to location of object in image, treats local pixels same as pixels far apart
* Pooling layer is used to reduce size
* CNN is not fully connected
* Conv layer has particular number of filters and size of filter
* Convolving = When filter goes through every pixel in input and generate a feature map
* Output size = (N-F) / stride + 1

**RNN (Recurrent Neural Networks)**

* Feed Forward NN cannot handle sequential data because it only focus on current state
* **Ht = fw(ht-1, xt) yt = Why Ht**
* RNN are good at processing sequence data for predictions but suffer from short term memory (faster and less resources because there are less tensor operations)
* RNN offer a lot of flexibility (1:M, M:1, M:M, 1:1(Vanilla NN))
* The repeating module in standard RNN contains single layer
* Short term memory and vanishing gradient are problems due to nature of back propagation
* FNN train in forward direction, makes prediction and then compares it with the truth to find the error value and backpropagate this error to calculate gradient
* Gradient is value used to adjust the weights. Greater the gradient greater the adjustment
* Vanishing gradient problem : When the error back propagates the upper layers are adjusted much and the next layers are adjusted a little, and gradient decreases exponentially as it goes down the layers. So the starting layers weight are barely adjusted and there is no learning in there.
* Because of the vanishing gradient RNN does not learn long range dependencies across time, hence RNN makes prediction with only what it has. Not being able to learn from earlier time steps causes network to have short term memory
* To mitigate short term memory in RNN - LSTM and GRU were invented (specialized RNN)
* LSTM are able to learn long term dependencies thru mechanism called gates

**LSTM**

* The repeating module in LSTM contains 4 interacting layers
* LSTM has three gates (regulated structure) to protect and control cell state. LSTM can add or remove info to cell state
* Forget gate : decides which info is useless. Use ht-1 and xt, outputs between 0 and 1. 1 means keep it, 0 means throw it
* Step 2 = sigmoid (input gate layer) + tanh, sigmoid layer decides what values we will update. Tanh layer creates vector of new candidate values. (Update old cell state by replacing)
* Final step = sigmoid + tanh. Sigmoid decides which parts of cell state we are going to output, put the cell state through tanh and multiply it by output of sigmoid gate.

**GRU (Gated Recurrent Unit)**

* GRU are faster to train and need fewer data to generalize
* GRU has 2 gates : Update gate, Reset gate
* Update gate : how much parse memory to keep
* Reset gate : how much parse memory to forget
* More popular now