What is machine learning?

**What are the applications of Machine Learning?**

Companies like Google, Facebook, Baidu, IBM, Microsoft uses ML extensively to push their respective ads to the relevant users.

Here are a few applications that you should know:

1. Banking & Financial services: ML can be used to predict the customers who are likely to default from paying loans or credit card bills. This is of paramount importance, as machine learning would help the banks to identify the customers who can be granted loans and credit cards.
2. Healthcare: It is used to diagnose deadly diseases (e.g. cancer) based on the symptoms of patients and tallying them with the past data of similar kind of patients.
3. Retail: It is used to identify products which sell more frequently (fast moving) and the slow moving products which help the retailers to decide what kind of products to introduce or remove from the shelf. In addition, machine-learning algorithms can be used to find which two / three or more products sell together. This is done to design customer loyalty initiatives, which in turn helps the retailers to develop and maintain loyal customers.

**Types of Machine learning**

**Logistic Regression**

Logistic regression is in reality an ordinary regression using the logit as the response variable. The logit transformation allows for a linear relationship between the response variable and the coefficients.

The coefficient 1.694596 implies that a one unit change in gender results in a 1.694596 unit change in the log of the odds. Odds ratio can be computed by taking ecoefficient value.

Assumptions:-

* The observations are independent.
* The independent variables are not linear combinations of each other.

**Q1. How to interpret odds ratio?**

**Q2. What is the algorithm behind logistic regression similar to OLS in linear regression?**

logistic regression is based on Maximum Likelihood Estimation which says coefficients should be chosen in such a way that it maximizes the Probability of Y given X (likelihood). With ML, the computer uses different "iterations" in which it tries different solutions until it gets the maximum likelihood estimates.

Q3. What is goodness of fit and how it is different from predictive power of the model?

Goodness of fit is measured on the training error

Predictive power has to do with model accuracy metrics i.e. test error

So it can be the scenario that model with the best fit might perform worse on predictive accuracy on unseen/test data and so there needs to be a real choice out there. Goodness of fit is more important while thinking of model as a purely descriptive and providing summary of the data

Q4 what is the difference between a logistic function and sigmoid function?

|  |  |
| --- | --- |
|  | [Logistic Function](https://i.stack.imgur.com/E1DX2.png)  Fig 1. Logistic Function  [Sigmoid Function](https://i.stack.imgur.com/eeba6.png)  Fig 2. Sigmoid Function |

The sigmoid function is a special case of the Logistic function when L=1, k=1, x0=0.

Parameter k controls how steep the change from the minimum to the maximum value is. L is the maximum value the function can take.

<http://www.wolframalpha.com/input/?i=plot+2%2F%7B1%2Bexp%7B-2%7Bx-0%7D%7D%7D,+with+-10+%3C+x+%3C+10>

Logistic regression is another generalized linear model (GLM) procedure using the same basic formula, but instead of the continuous Y, it is regressing for the probability of a categorical outcome. In simplest form, this means that we are considering just one outcome variable and two states of that variable- either 0 or 1.

The equation for the probability of *Y*=1 looks like this:

*P*(*Y*=1)=11+*e*−(*b*0+∑(*biXi*))

Your independent variables *Xi*

can be continuous or binary. The regression coefficients *bi* can be exponentiated to give you the change in odds of *Y* per change in *Xi*, i.e., *Odds*= *P*(*Y*=1)/*P*(*Y*=0) =*P*(*Y*=1)/1−*P*(*Y*=1) and Δ*Odds*=*ebi*. Δ*Odds* is called the odds ratio, *Odds* (*Xi*+1)/*Odds*(*Xi*). In English, you can say that the odds of *Y*=1 increase by a factor of *ebi* per unit change in *Xi*

.

Example: If you wanted to see how body mass index predicts blood cholesterol (a continuous measure), you would use linear regression as described at the top of my answer. If you wanted to see how BMI predicts the odds of being a diabetic (a binary diagnosis), you would use logistic regression.

The logit is a link function / a transformation of a parameter. It is the logarithm of the odds.

If we call the parameter π, it is defined as follows:

logit(*π*)=log(*π/*1−*π*)

The [logistic](http://en.wikipedia.org/wiki/Logistic_function) function is the inverse of the logit. If we have a value, x, the logistic is:

logistic(*x*)=*ex/*1+*ex*

Q5. What is the difference between logit and probit model and what to use when?

Logit and probit differ in how they define f (\*). The logit model uses something called the cumulative distribution function of the logistic distribution. The probit model uses something called the cumulative distribution function of the standard normal distribution to define f (\*). Both functions will take any number and rescale it to fall between 0 and 1.

Is logit better than probit, or vice versa? Both methods will yield similar (though not identical) inferences. Logit – also known as logistic regression – is more popular in health sciences like epidemiology partly because coefficients can be interpreted in terms of odds ratios. Probit models can be generalized to account for non-constant error variances in more advanced econometric settings (known as heteroskedastic probit models) and hence are used in some contexts by economists and political scientists. Logit has better interpretation than probit. Logistic regression can be interpreted as modeling log odds. Usually people start the modeling with logit

Q6. What is heteroscedasticity and does it impact logistic regression and if no , how logistic regression is able to overcome this?

Q7 Why normality assumptions of linear regression don’t have any relevance for logistic regression?

Q8 what is overdispersion?

Q9 what is wald statistic in logistic regression?

Q10 What is Hosmer–Lemeshow test?

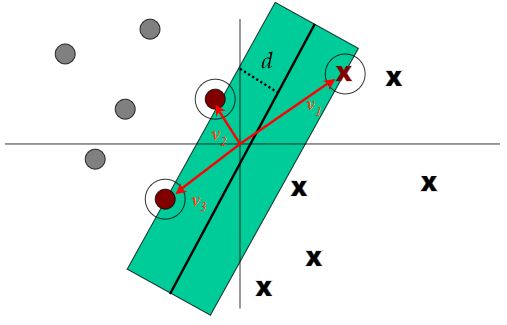
Q11 What is a AIC metric?

Q12 What is a deviance ?

<http://stats.stackexchange.com/questions/18750/hosmer-lemeshow-vs-aic-for-logistic-regression?rq=1>

**Support vector Machines**

Support vectors are the data points that lie closest to the decision surface (or hyperplane)



**Margin of Separation** (d):- the separation between the hyperplane and the closest data point for a given weight vector w and bias b.

Few keywords – margin, gaps, optimal margin classifier, Lagrange duality, kernels, SMO algorithm

Margins –

If a point is far from the separating hyperplane, then we are quite confident on its belongingness to a particular class/category. Main idea is to find a decision boundary that allows us to make correct and confident predictions.

Let us think of a linear classifier hw,b(x) = g(wTx + b) . Hence g(z) = 1 if z>=0 else g(z) = -1 where w is similar to [Θ1, Θ2, Θ3, …,Θn-1, Θn]T and b is similar to Θ0

Functional margin – is not a good measure of confidence. Functional margin although tells you the sign of the label but it doesn’t tell the magnitude with reference on how far it is from decision plane

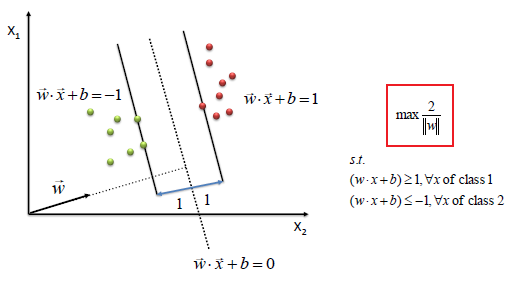
Functional margin (ϒ) of (w,b) wrt training set S of size m :-

ϒ(i)  = yi(wTxi + b)

& ϒ = min12..m ϒ(i)

Geometric margin is a better measure and so defined by and it is only a normalized version of functional margin with the help of weight vector. The geometric margin is telling you not only if the point is properly classified or not, but the magnitude of that distance in term of units of |w|

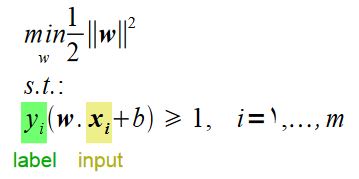
Another important discussion point is why it is profitable to find a wide margin instead of narrow margin.



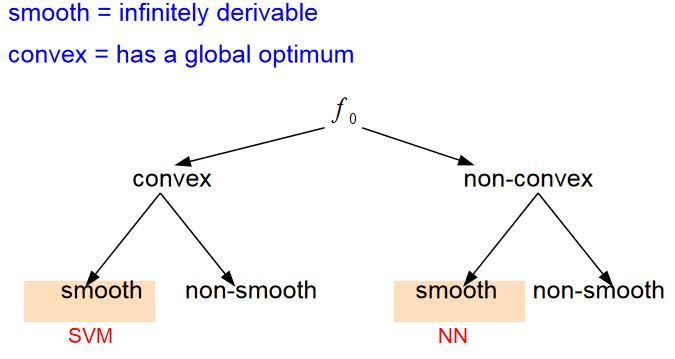
**Optimal Margin Classifier**

Here maximization 1/|w| problem has been converted to minimization ½ wTw problem.

where yi(wTxi + b) >=1.



Above is an optimization problem with convex quadratic objective and linear constraints and Its solution gives us the optimal margin classifier

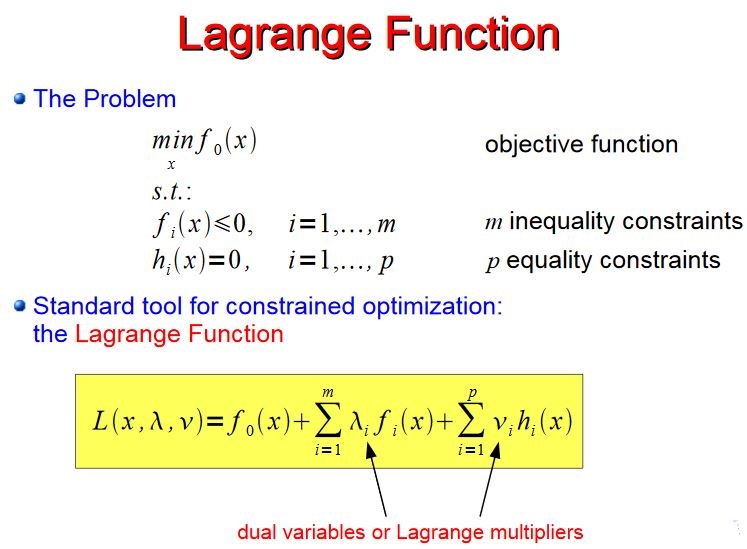


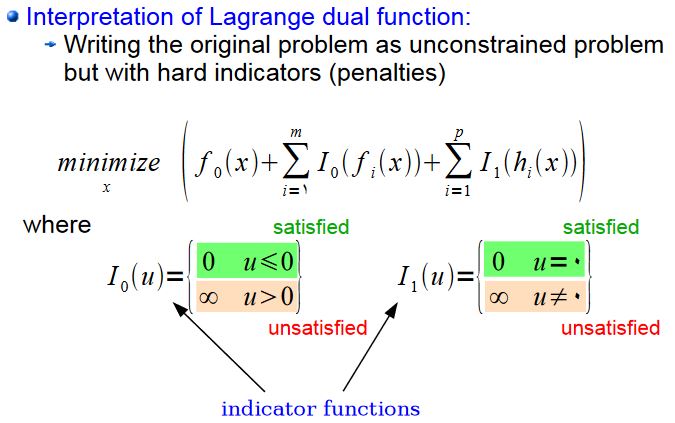
So, we need to ensure if the function is convex and smooth and only then a global solution will exist and SVM will be applicable for results with higher confidence.

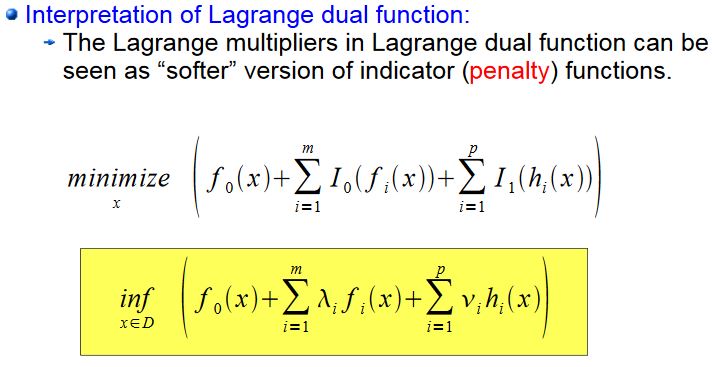
Until now we have been defined the problem in primal form.

Now we shall discuss what is dual problem and why we convert primal to dual. Why it is important to convert it into dual problem is that **it allows us to use kernels which helps us solving the problem efficiently in very high dimensional spaces**.

What is Lagrange function?





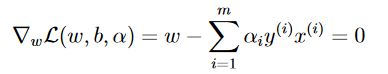


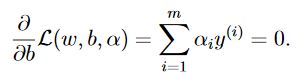
Lagrange dual is convex even if original problem is not

Karush-Kuhn-Tucker (KKT) conditions;-

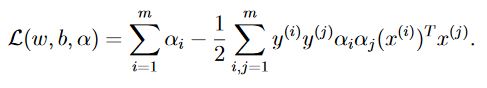
Karush–Kuhn–Tucker (KKT) conditions, also known as the Kuhn–Tucker conditions, are first-order necessary conditions for a solution in nonlinear programming to be optimal. Allowing inequality constraints, the KKT approach to nonlinear programming generalizes the method of Lagrange multipliers, which allows only equality constraints

Let’s take the derivative of L(w, λ, ν) w.r.t. w and b and set them to zero.

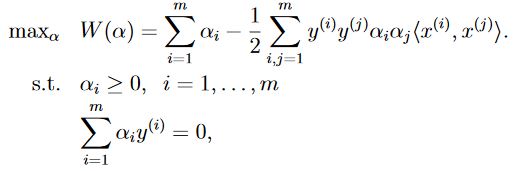




*After solving the above derivative equations*

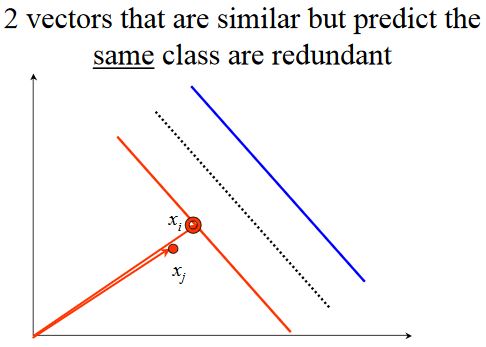
****

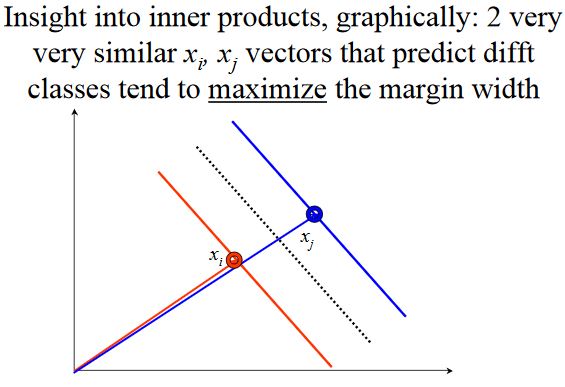
*Converting it into original problem format where L is the derivative wrt w and so putting it back with constraints αi ≥ 0*

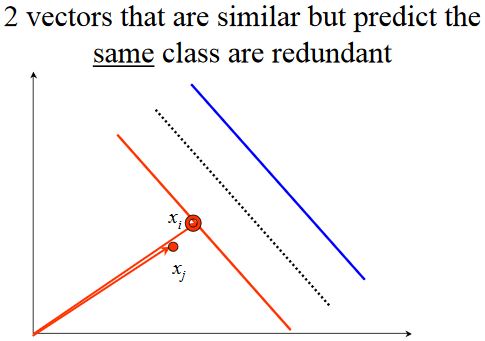
****

***Lagrange dual problem says instead of minimizing over weight w and bias b subjects to constraints involving a, we can maximize over a’s(dual variables).***

**Now the problem has become computationally achievable, as dual form just requires it to compute dot products of training points.**

******

******

****

**Hidden Markov model**

**RandomForest**

**GradientBoostingTrees (**random set of features or all used, fraction of samples for overfitting**)**

**Hyper-parameter (important ones) tuning in XGBoost**

1. Gamma: - it is a pseudo-regularization parameter in gradient boosting. It is dependent on training dataset and other parameters. Higher the Gamma, higher will be the regularization and at value 0, there is no regularization. Gamma values more than 20 are very high and should be used in conjunction with high max depth. Higher value of gamma means more positive loss reduction required to further split the trees. It becomes more conservative to adapt to complexity in the model. Gamma thus in a way help avoid overfitting by pruning the tree.
2. Max depth: - It is how many levels deep a tree is allowed to go. The larger the value more is the overfitting of the training set. Good values range from 6 to 20.
3. Min child weight: - This is how each group in the tree has to be. Larger values are more robust than smaller values. However, the higher max\_depth is, the higher this value should also be in order to avoid overfitting. In simple words if the minimum sum of instance weight is less than the min child weight, tree would stop further partitioning

Tree based Modeling

#TODO - <https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/>

Some advantages of decision trees are:

* Simple to understand and to interpret. Trees can be visualised.
* Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
* The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
* Able to handle both numerical and categorical data. Other techniques are usually specialised in analysing datasets that have only one type of variable. See [algorithms](http://scikit-learn.org/stable/modules/tree.html#tree-algorithms) for more information.
* Able to handle multi-output problems.
* Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
* Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
* Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The disadvantages of decision trees include:

* Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning (not currently supported), setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
* Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
* The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
* There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
* Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

Important points to note:-

* [CART](https://en.wikipedia.org/wiki/Predictive_analytics#Classification_and_regression_trees_.28CART.29) (Classification and Regression Trees) is very similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node.
* scikit-learn uses an optimised version of the CART algorithm.

**Classification criteria –** Gini, Cross entropy, Misclassification error

Must read - <https://sebastianraschka.com/faq/docs/decisiontree-error-vs-entropy.html>

<https://www.garysieling.com/blog/sklearn-gini-vs-entropy-criteria>

<http://www.bogotobogo.com/python/scikit-learn/scikt_machine_learning_Decision_Tree_Learning_Informatioin_Gain_IG_Impurity_Entropy_Gini_Classification_Error.php>

<https://github.com/rasbt/python-machine-learning-book/blob/master/faq/decision-tree-binary.md>

**Regression criteria** – squared mean error

**Tips for practical use of decision trees:-**

* Decision trees tend to overfit on data with a large number of features. Getting the right ratio of samples to number of features is important, since a tree with few samples in high dimensional space is very likely to overfit.
* Consider performing dimensionality reduction ([PCA](http://scikit-learn.org/stable/modules/decomposition.html#pca), [ICA](http://scikit-learn.org/stable/modules/decomposition.html#ica), or [Feature selection](http://scikit-learn.org/stable/modules/feature_selection.html#feature-selection)) beforehand to give your tree a better chance of finding features that are discriminative.
* Visualise your tree as you are training by using the export function. Use max\_depth=3 as an initial tree depth to get a feel for how the tree is fitting to your data, and then increase the depth.
* Remember that the number of samples required to populate the tree doubles for each additional level the tree grows to. Use max\_depth to control the size of the tree to prevent overfitting.
* Use min\_samples\_split or min\_samples\_leaf to control the number of samples at a leaf node. A very small number will usually mean the tree will overfit, whereas a large number will prevent the tree from learning the data. Try min\_samples\_leaf=5 as an initial value. If the sample size varies greatly, a float number can be used as percentage in these two parameters. The main difference between the two is that min\_samples\_leaf guarantees a minimum number of samples in a leaf, while min\_samples\_split can create arbitrary small leaves, though min\_samples\_split is more common in the literature.
* Balance your dataset before training to prevent the tree from being biased toward the classes that are dominant. Class balancing can be done by sampling an equal number of samples from each class, or preferably by normalizing the sum of the sample weights (sample\_weight) for each class to the same value. Also note that weight-based pre-pruning criteria, such as min\_weight\_fraction\_leaf, will then be less biased toward dominant classes than criteria that are not aware of the sample weights, like min\_samples\_leaf.
* If the samples are weighted, it will be easier to optimize the tree structure using weight-based pre-pruning criterion such as min\_weight\_fraction\_leaf, which ensure that leaf nodes contain at least a fraction of the overall sum of the sample weights.
* All decision trees use np.float32 arrays internally. If training data is not in this format, a copy of the dataset will be made.
* If the input matrix X is very sparse, it is recommended to convert to sparse csc\_matrix before calling fit and sparse csr\_matrix before calling predict. Training time can be orders of magnitude faster for a sparse matrix input compared to a dense matrix when features have zero values in most of the samples.

**Ensembling in Decision trees:-**

**Boosting**

[Gradient Tree Boosting](https://en.wikipedia.org/wiki/Gradient_boosting) or Gradient Boosted Regression Trees (GBRT) is a generalization of boosting to arbitrary differentiable loss functions

The advantages of GBRT are:

* Natural handling of data of mixed type (= heterogeneous features)
* Predictive power
* Robustness to outliers in output space (via robust loss functions)

The disadvantages of GBRT are:

* Scalability, due to the sequential nature of boosting it can hardly be parallelized.

**Loss function in Boosting**

* Regression
  + Least squares ('ls'): The natural choice for regression due to its superior computational properties. The initial model is given by the mean of the target values.
  + Least absolute deviation ('lad'): A robust loss function for regression. The initial model is given by the median of the target values.
  + Huber ('huber'): Another robust loss function that combines least squares and least absolute deviation; use alpha to control the sensitivity with regards to outliers (see [[F2001]](http://scikit-learn.org/stable/modules/ensemble.html#f2001) for more details).
  + Quantile ('quantile'): A loss function for quantile regression. Use 0 < alpha < 1 to specify the quantile. This loss function can be used to create prediction intervals (see [Prediction Intervals for Gradient Boosting Regression](http://scikit-learn.org/stable/auto_examples/ensemble/plot_gradient_boosting_quantile.html#sphx-glr-auto-examples-ensemble-plot-gradient-boosting-quantile-py)).
* Classification
  + Binomial deviance ('deviance'): The negative binomial log-likelihood loss function for binary classification (provides probability estimates). The initial model is given by the log odds-ratio.
  + Multinomial deviance ('deviance'): The negative multinomial log-likelihood loss function for multi-class classification with n\_classes mutually exclusive classes. It provides probability estimates. The initial model is given by the prior probability of each class. At each iteration n\_classes regression trees have to be constructed which makes GBRT rather inefficient for data sets with a large number of classes.
  + Exponential loss ('exponential'): The same loss function as [AdaBoostClassifier](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#sklearn.ensemble.AdaBoostClassifier). Less robust to mislabeled examples than 'deviance'; can only be used for binary classification.

**Deep Learning**

Deep learning is a family of advanced machine learning algorithms which used the concept of human brain neurons to model arbitrary functions to capture the real world non-linear complexities which traditional machine learning algorithms are not capable of. These algorithms require lot of data to perform.

Libraries like Theano and Tensorflow help in specific machine learning number-crunching operations like derivatives on huge matricies with large efficiency

**Deep Learning, NLP and intersection**:-

Deep Learning – is a subfield of machine learning. it is a family of advanced machine learning algorithms which used the concept of human brain neurons to model arbitrary functions to capture the real world non-linear complexities which traditional machine learning algorithms are not capable of. These algorithms require lot of data to perform.

It attempts to learn representations and output. It can learn unsupervised(from raw text) or supervised(with specific labels)

Deep learning in speech

Deep learning in vision

NLP – Natural Language Processing is a field at the intersection of computer science, artificial intelligence and linguistics.

Reason for Intersection of Deep Learning and NLP is that it will help computers to understand natural language like humans to perform useful tasks e.g. Question-Answering

Following steps can be performed in general:-

1. Speech (phonetic or phonological analysis) or text(OCR/tokenization)
2. Morphological Analysis
3. Syntactic Analysis
4. Semantic Interpretation
5. Discourse Processing

Applications of NLP – spell checking, keyword search, finding synonyms, extracting info from websites such as product prices, names, date, location etc., classification on reading level of text , sentiment analysis and identification on positive or negative. More complex tasks like machine translation, complex question answering or spoken dialogue systems.

NLP in industry –

1. Search (written and spoken)
2. Online Advertisements
3. Automated translation
4. Sentiment Analysis for marketing/trading/elections
5. Speech recognition
6. Automating customer support

Why NLP is complex?

NLP disambiguos examples

Deep Learning + NLP = Deep NLP

Earlier semantics were represented by Lambda Calculus but with deep learning it’s just a vector and calculations

Also traditional approaches involve using bag of words representation or hand designated negation features.

Question Answering –

A lot of feature engineering to capture real world e.g. regular expressions

**Word Vectors**

**Count based(traditional) vs Prediction based(unconventional)**

1. Document word concurrence matrix - lead to general topics
2. Window based concurrence matrix – captures both syntactic and semantic information

Dimensionality reduction on concurrence matrix X with the help of singular value decomposition

A word can be represented by a dense vector. How many dimensions to use is something an important decision based on the nature of the problem.

SVD doesn’t scale and computation time increases quadrtically

Word2vec is something as state of the art. Predict surrounding words in a window of length m of every word

import numpy as np

import random

*def softmax(x):*

*"""Compute the softmax function for each row of the input x. It is crucial that this function is optimized for speed because it will be used frequently in later code.You might find numpy functions np.exp, np.sum, np.reshape,np.max, and numpy broadcasting useful for this task. (numpy broadcasting documentation:*

*http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html*

*"""tmp\_x = np.max(x.T, axis=0)*

*tmp\_exp = np.exp(x.T-tmp\_x)*

*x = tmp\_exp / np.sum(tmp\_exp, axis=0)*

*x = x.T*

*return x*



**Motivation: Why Learn Word Embeddings?**

NLP systems traditionally treat words as discrete atomic symbols, and therefore 'cat' may be represented as Id537 and 'dog' as Id143. These encodings are arbitrary, and provide no useful information to the system regarding the relationships that may exist between the individual symbols. This means that the model can leverage very little of what it has learned about 'cats' when it is processing data about 'dogs' (such that they are both animals, four-legged, pets, etc.). Representing words as unique, discrete ids furthermore leads to data sparsity, and usually means that we may need more data in order to successfully train statistical models.

Vector space models (VSMs) represent (embed) words in a continuous vector space where semantically similar words are mapped to nearby points ('are embedded nearby each other'). VSMs have a long, rich history in NLP, but all methods depend in some way or another on the Distributional Hypothesis, which states that words that appear in the same contexts share semantic meaning. The different approaches that leverage this principle can be divided into two categories: count-based methods (e.g. Latent Semantic Analysis), and predictive methods (e.g. neural probabilistic language models).

In a nutshell: Count-based methods compute the statistics of how often some word co-occurs with its neighbor words in a large text corpus, and then map these count-statistics down to a small, dense vector for each word. Predictive models directly try to predict a word from its neighbors in terms of learned small, dense embedding vectors (considered parameters of the model).

Word2vec is a particularly computationally efficient predictive model for learning word embeddings from raw text. It comes in two flavors, the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model. Algorithmically, these models are similar, except that CBOW predicts target words (e.g. 'mat') from source context words ('the cat sits on the'), while the skip-gram does the inverse and predicts source context-words from the target words. This inversion might seem like an arbitrary choice, but statistically it has the effect that CBOW smoothes over a lot of the distributional information (by treating an entire context as one observation). For the most part, this turns out to be a useful thing for smaller datasets. However, skip-gram treats each context-target pair as a new observation, and this tends to do better when we have larger datasets.

**Skip- gram model**

E.g. **the quick brown fox jumped over the lazy dog**

We first form a dataset of words and the contexts in which they appear. We could define 'context' in any way that makes sense, and in fact people have looked at syntactic contexts (i.e. the syntactic dependents of the current target word, see e.g. Levy et al.), **words-to-the-left of the target, words-to-the-right of the target**, etc. Using a window size of 1, we then have the dataset

**([the, brown], quick), ([quick, fox], brown), ([brown, jumped], fox), ...**

of (context, target) pairs. Recall that skip-gram inverts contexts and targets, and tries to predict each context word from its target word, so the task becomes to predict 'the' and 'brown' from 'quick', 'quick' and 'fox' from 'brown', etc. Therefore, our dataset becomes

**(quick, the), (quick, brown), (brown, quick), (brown, fox), ...**

of (input, output) pairs. The objective function is defined over the entire dataset, but we typically optimize this with stochastic gradient descent (SGD) using one example at a time (or a 'minibatch' of batch\_size examples, where typically 16 <= **batch\_size** <= 512)

**CBOW model**

Algorithmically, these models are similar, except that CBOW predicts target words (e.g. 'mat') from source context words ('the cat sits on the'), while the skip-gram does the inverse and predicts source context-words from the target words

**Sampling**

It is a statistical procedure concerned with the selection of the individual observation to make statistical inferences about the population.

Types of sampling –

Random sampling – simple random sampling, equal probability systematic sampling, stratified simple random sampling, multistage stratified random sampling, cluster sampling, Multistage cluster sampling,

Non-random sampling – availability sampling, quota sampling, expert sampling,

Probability or non-probability sampling

Bootstrap sampling – take data set of size n and draw n samples with replacement to create new dataset and repeat 1000 times.

**Important Concepts:-**

**26-01-2017**

1. Mahalanobis distance – It is best suited to detect outliers in n-dimensions where n >1. Mahalanobis’ distance is a statistical measure of the extent to which cases are multivariate outliers, based on a chi-squared distribution. R's mahalanobis() function provides a simple means of detecting outliers in multidimensional data.

#TODO - <https://www.r-bloggers.com/outlier-detection-with-mahalanobis-distance/>

1. Goodness-of-fit - A goodness-of-fit test, in general, refers to measuring how well do the observed data correspond to the fitted (assumed) model

**Chi Square test, Kolmogorov–Smirnov test, Cramér–von Mises criterion**

#TODO - <https://www.r-bloggers.com/goodness-of-fit-test-in-r/>

1. Bootstrap sampling - If we don’t have enough data to train our algorithm then we can increase the size of our training set by randomly selecting items and duplicating them (with replacement).

#TODO - <http://www.statisticssolutions.com/sample-size-calculation-and-sample-size-justification/sampling/>

1. Adjusted Rand Score – Given the knowledge of the ground truth class assignments labels\_true and our clustering algorithm assignments of the same samples labels\_pred, the adjusted Rand index is a function that measures the similarity of the two assignments, ignoring permutations and with chance normalization. ARI requires knowledge of the ground truth classes while is almost never available in practice or requires manual assignment by human annotators

ARI = (RI - Expected\_RI) / (max(RI) - Expected\_RI)

#TODO - <http://scikit-learn.org/stable/modules/clustering.html#adjusted-rand-score>

1. For K-fold cross validation, what k should be selected?

#TODO - <http://stats.stackexchange.com/questions/61783/variance-and-bias-in-cross-validation-why-does-leave-one-out-cv-have-higher-var>

<https://www.quora.com/For-K-fold-cross-validation-what-k-should-be-selected>

1. How to deal with multi-collinearity situation in linear regression model? - To check multicollinearity, we can create a correlation matrix to identify & remove variables having correlation above 75% (deciding a threshold is subjective). In addition, we can use calculate VIF (variance inflation factor) to check the presence of multicollinearity. VIF value <= 4 suggests no multicollinearity whereas a value of >= 10 implies serious multicollinearity. Also, we can use tolerance as an indicator of multicollinearity.But, removing correlated variables might lead to loss of information. In order to retain those variables, we can use penalized regression models like ridge or lasso regression. Also, we can add some random noise in correlated variable so that the variables become different from each other. But, adding noise might affect the prediction accuracy, hence this approach should be carefully used.
2. Threshold in classification problems(Confidence Splitting criteria) - <http://nerds.airbnb.com/confidence-splitting-criterions/>
3. Accuracy paradox in an imbalanced dataset – Accuracy should not be used as the measure of model performance because accuracy might be only predicting major class correctly but we are interested only in the minor class. Therefore, we should use sensitivity i.e. true positive rate (also called as recall), specificity i.e. true negative rate, and precision i.e. positive predicted value). F measure to determine class wise performance of the classifier.

#TODO - <https://en.wikipedia.org/wiki/Sensitivity_and_specificity>

sensitivity is how many correctly predicted out of true positive labels and specificity how many true negative predicted out of true negative labels

#TODO - <https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/>

1. Ensemble Learning – use multiple algorithms for prediction and they combine output of multiple machine learning algorithms for getting more robust or generalized output which outperform all the individual models

#TODO - <https://www.analyticsvidhya.com/blog/2015/09/questions-ensemble-modeling/>

#TODO - <https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/>

#TODO - <http://mlwave.com/kaggle-ensembling-guide/>

1. R-squared and adj. R-squared – R-squared cannot determine whether the coefficient estimates and predictions are biased, which is why we must assess the residual plots. However, R-squared has additional problems that the adjusted R-squared and predicted R-squared are designed to address.Every time you add a predictor to a model, the R-squared increases or remains same.

#TODO - <https://discuss.analyticsvidhya.com/t/difference-between-r-square-and-adjusted-r-square/264/3>

**27-01-2017**

1. Entity disambiguation –

#TODO - <http://www.theatlantic.com/technology/archive/2011/03/does-anne-hathaway-news-drive-berkshire-hathaways-stock/72661/>

Machine learning interview questions

1. How CHAID and random forest are different from objective function perspective
2. How to detect solve multicollinearity graphically
3. What is the difference between r2 and adjusted r2?
4. Correlation and partial correlation?
5. Multicollinearity?
6. Formula for multicollinearity
7. Concordance and discordance in logistics regression
8. Formula for f value in regression
9. PCA formula
10. Arima la formula
11. Lazy vs eager learning in KNN
12. Why SGD is sensitive to feature scaling
13. Is SGD a sequential algorithms and how can it be parallelized
14. How SGD is different from gradient descent algorithms
15. Why is AdaBoost algorithm sensitive to noisy data and outliers? And how? Due to exponential loss function <https://stats.stackexchange.com/questions/20622/is-adaboost-less-or-more-prone-to-overfitting>