To do list:

1. Describe data science and ML projects
2. . Also more general questions around background, experience and motivation.
3. Why using random forest rather than other algorithms.
4. Skype, with ML questions about classic algorithms
5. deep learningetc. and more high-level questions about client interactions.
6. How do you explain the current rise of deep learning?
7. SVM,
8. Classification, Regression, Decision Trees.
9. code kmeans from scratch, code knn from scratch
10. - What is machine learning?  
    - Graph and matrix operation.  
    - Linear regression theory and derivation.
11. stochastic optimization, deep learning, linear programming, graph traversal, sparse matrix implementation, and a case study involving feature generation.
12. How are Scipy sparse matrices implemented

system/model estimation  general form for solving a linear least squares problem

1. What is PCA/Singular Value Decomposition and can you derive why it works?

I am currently a master student in biostatistics. Before this I got my bachelor’s degree in biological science. Based on my experience over the past two years , I think I am more into applying analytical skills to practical problems. That is why I would like to launch my career from a data science related position. I have taken multiple classes covering most common topics in statistics. I have also put extra energy to the classes where I can sharpen my programming skills and pick up machine learning as much as possible. Both R and python are extensively used in my school work and research project for the purpose of data management, data visualization, statistical inference, simulations and model building. Last summer, I interned in a biotech company doing clinical analysis and data mining. It was from that time that I started feeling that there is a lot of space for me to do creative work in data science. We need to keep abreast of the most up to date methods which is a great learning opportunity. For me/I feel, learning is more motivated and efficient in practice. Also, I am very happy to present and to translate technical results to audience with different backgrounds. So I look forward to building up my industry experience and expanding my knowledge in this area.

**Why quora:**

Firstly, quora is a platform for sharing knowledge and experience which I like a lot. It provides a window for people to go beyond their daily life and gain knowledge of the world outside. Quora covers a wide spectrum of topics from academics, life styles to interpersonal relationships. The great answers to questions are collected from people with rich experience or people can articulate complex ideas in an attractive way. It basically gathers the wisdom of everyone, organize it, and make it helpful to a larger population.

Secondly, I am actively looking for positions as data scientist or statistician. Since quora is a data-driven company, I believe there would be plenty of room for creativity. Many data scientist positions are solely targeted for commercial goals like how to maximize the revenue from ads or how to increase the click rate or conversion rate. But I feel the data in quora has richer flavor than the standard user indexes. A lot more interesting and meaningful work can be done with such data.

Also, I’ve noticed that quora’s data scientists come from a variety of backgrounds. Compared to other companies that require a ton of programming skills and experience, I feel that such diversity allows for the fusion of different insights and makes the work idea-oriented, which is cool/fun.

Questions for them:

The order of related questions

Long-term research figuring out user behavior product development/improvement

Not short-sighted

A/B test : more like the traditional the website/phone-based experiments primarily serving for optimizing advertising/straightforward and measureable goals

For the investigation of users behavior/ long term research/ tricky less straightforward to come up with appropriate metrics.

I have read through several data science work examples posted on your blog.

How well does the metrics you come up with reflect the questions you are trying to figure out

High volume of data allows for explorations

Features I like:

Simple clean layout of the webpage. Not busy, easy to read.

The bio attached after user’s ID. It somewhat gives me a quick impression of the reliability and the limitation of answers.

To improve:

I’ve seen a lot of great answers elaborating on all details they are related to. It is definitely a good thing but maybe most people are not that patient therefore some great answers probably don’t get enough attention and recognization. I think it might help it quora can provide summarized version next to the original answers to give users a quick overview. People who are really interested may choose to read the full text.

Another thing to pay attention is the authenticity of answers. Some answers are made-up but have won many upvotes. Using a combination of manual inspection and AI tools may help with this.

**Strengths:**

Work independently and I am a quick learner. I am willing to learn whatever is helpful for my job.

Besides, I think I am relatively optimistic person and don’t get easily frustrated by the difficulties in work.

I don’t procrastinate. Ive never had things piled up before the deadline. I usually plan early and finish early.

**Weakness:**

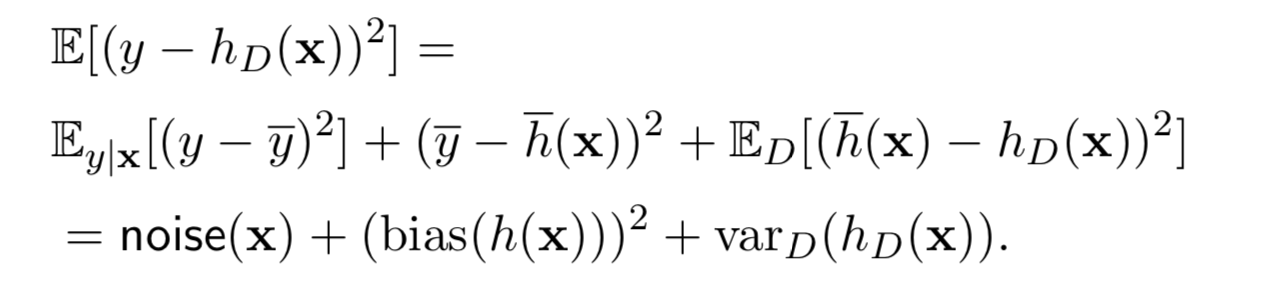
I am a new grad with very little work experience. This has been a biggest obstacle in my job hunting so far but it is the fact that I have to face. Experience accumulates with time and I think I need a little patience.

1. **Bias vs. Variance**: two sources of error that depends on the complexity of model and affect models’ ability to generalize beyond the training set.

Bias: caused by the deviation of model from the true structure in data. It is usually a problem of over-simplified models. Models that are not able to capture the relations between features and output and perform poorly on both the training set and the test set usually have the problem of underfitting, aka, bias

Variance: is an error from sensitivity to the choice of training data. If we build model several times on different training sets, the model estimates of complex models can vary a lot from each other. Those models capture too many details from the training set and therefore have low generalization. Overfitting.

The model with the optimal predictive capability is the one that leads to the best balance between bias and variance.



**how do you know if your model has high bias or high variance?**

Compare training error to the test error of a model:

High bias: training error and test error are quite close to each other and both performance are poor.

High variance: test error is much higher than the training error. ( training error below the tolerance, test error is above the tolenrance)

the **bias–variance tradeoff** is the problem of simultaneously minimizing two sources of [error](https://en.wikipedia.org/wiki/Errors_and_residuals_in_statistics) that prevent [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) algorithms from generalizing beyond their [training set](https://en.wikipedia.org/wiki/Training_set)

* The [*bias*](https://en.wikipedia.org/wiki/Bias_of_an_estimator) is an error from erroneous assumptions in the learning [algorithm](https://en.wikipedia.org/wiki/Algorithm). High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting). (add more data points for the use of more complex model)
* The [*variance*](https://en.wikipedia.org/wiki/Variance) is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random [noise](https://en.wikipedia.org/wiki/Noise_(signal_processing)) in the training data, rather than the intended outputs ([overfitting](https://en.wikipedia.org/wiki/Overfitting)). (ensemble method, decrease variance without adding bias)

2**. Bagging**: generate multiple bootstrapped training sets and build separate prediction model using each training set. The final prediction would be the average of predictions from all models. (more commonly used in decision tree) where each individual tree has high variance but averaging all trees reduces the variance.

Majority vote ( no overfitting if # of trees goes up) out of bag error

**Random forest:** a combination of bootstrapped training set and randomly selected feature sets for each split. Decorrelate bagged trees by preventing strong predictors dominating the trees.

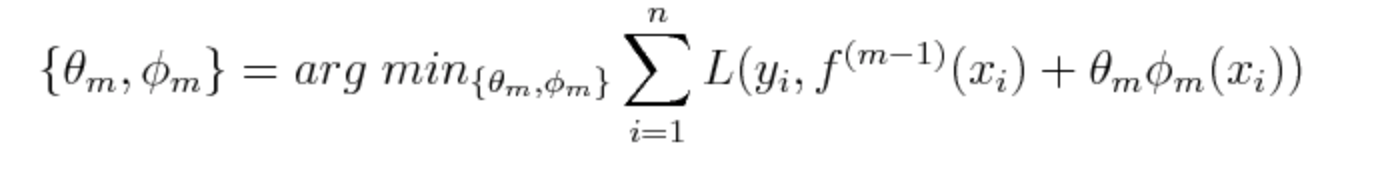
**Number of trees, number of features to be considered for each split, max depth of tree, split criterion(gini index or entropy), min number of samples at each leaf node.**

For hyperparameter tuning, we perform many iterations of the entire K-Fold CV process, each time using different model settings. We then compare all of the models, select the best one, train it on the full training set, and then evaluate on the testing set.

we only have a vague idea of the best hyperparameters and thus the best approach to narrow our search is to evaluate a wide range of values for each hyperparameter. Using Scikit-Learn’s RandomizedSearchCV method, we can define a grid of hyperparameter ranges, and randomly sample from the grid, performing K-Fold CV with each combination of values. [**RandomizedSearchCV**](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html#sklearn.model_selection.RandomizedSearchCV) implements a randomized search over parameters, where each setting is sampled from a distribution over possible parameter values.

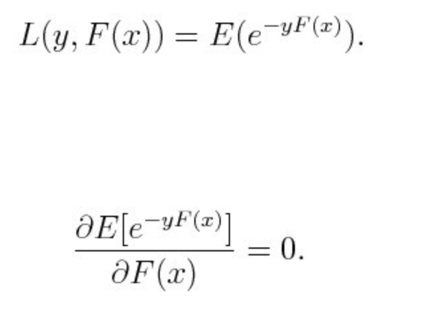
Exhaustive grid search

**Boosting:** build trees sequentially, no bootstrap sampling, the residuals from previous models are fit into later trees. (number of trees, shrinkage parameter/learning rate, number of split d)



 the ensemble problem is simplified greedily as a forward stage-wise additive model. We don’t optimize the ensemble in a global manner, but instead seek to improve the result based on the current estimate.

**AdaBoost**, it minimizes the exponential loss function under the constraint that **ϕm(x)** only outputs -1 or 1. heuristic approach without requiring expensive work in tuning hyperparameters for the learning rate schedule manually.



**Gradient boosting:**

 constructs a forward stage-wise additive model by implementing gradient descent in function space.

at the m\_th iteration, the direction of the steepest descent is given by the negative gradient of the loss function

ROC curve: True positive rate(TP/TP+FN) vs. false positive rate(FP/FP+TN) 1-specificity

AUC: overall performance of a classifier summarized over all possible thresholds

The ROC curve is a graphical representation of the contrast between true positive rates and the false positive rate at various thresholds.

Recall is also known as the true positive rate

Precision is also known as the positive predictive value,

Classification: perceptron loss

0/1 loss: uninformative, hinge loss/rectified linear activation

Convex, but does not have a closed-form solution.

Penalty scales with number of mistakes and how bad they are. Convex, but does not have a closed-form solution.

Issue with least square loss:

Take loss on correctly classified points.

Outlier correct points can change boundary.

Loss should target classification accuracy.

Logistic regression: the model specifies the linear discrimination boundary. Prediction is based on which side of the separator the predicted value is on. Model is trained via the perceptron algorithm which minimizes hinge loss/log likelihood . rectified linear activation (ReLU) /sigmoid activation

Multi-class Logistic Regression (LR) with a trained set of weights assigns x the class k for which it has the highest such score.

Naïve Baye: presence or absence of a specific feature of a class is not related to the presence or absence of any other feature

Tree: random forest, boosting

LDA, QDA(bayes): prior, normally distributed

KNN: non-parametric, flexible boundary. No interpretation on predictors

SVM

Dimension reduction :

access the lower-dimensional signal from the data.

minimizing the reconstruction loss of projecting data onto a d basis vectors, or as maximizing the variance in data that can be explained by d basis vectors.

Find the d largerst eigenvalues of normalized feature covariance, drop eigenvectors that are less important.

**Name some feature extraction techniques used for dimensionality reduction.**

* Independent Component Analysis
* Principal Component Analysis
* Kernel Based Principal Component Analysis

In *k*-fold cross-validation, the original sample is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, a single subsample is retained as the validation data for testing the model, and the remaining *k* − 1 subsamples are used as training data. The cross-validation process is then repeated *k* times, with each of the *k* subsamples used exactly once as the validation data. The *k* results can then be averaged to produce a single estimation.

**Feature selection:**

*redundant* or *irrelevant*,

1. Filter methods: The selection of features is independent of any machine learning algorithms. features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable. do not remove multicollinearity , do not consider the relationships between variables.
2. Wrapper method: try to use a subset of features and train a model using them. Computationally expensive. Forward selection, backward elimination, stepwise selection, all possible subsets.
3. Embedded methods/regularization : lasso and ridge avoid overfitting, rule out features
4. Tree based models calculates feature importance for they need to keep the best performing features as close to the root of the tree. The feature importance in tree based models are calculated based on Gini Index, Entropy or Chi-Square value.

Regularization: Gaussian prior or Laplacean prior on coefficients.

**K means**: find the representation of each cluster by the point which is the average of data assigned to this cluster.

start with randomly specifying K vectors as the centroids of K clusters. Then keep alternating between assigning data points to their closest means and updating the centroids of clusters until convergence. Local optimal

Lloyd’s algorithm finds a local optimal solution. Finding the global optimal is NP-hard. A common strategy is to use random restarts.

K-means++: to randomly select some of the data to be the first cluster centers. This is done by iteratively adding cluster centers, sampling them in proportion to the squared distance of each example from its nearest cluster center. Thus K-Means++ tends to favor points that are distant from the existing centers and produce a more diverse set of centers.

**Hierarchical agglomerative clustering**: regard all data points as clusters, keep merging two closest clusters among all existing clusters until all points are merged into one big cluster.

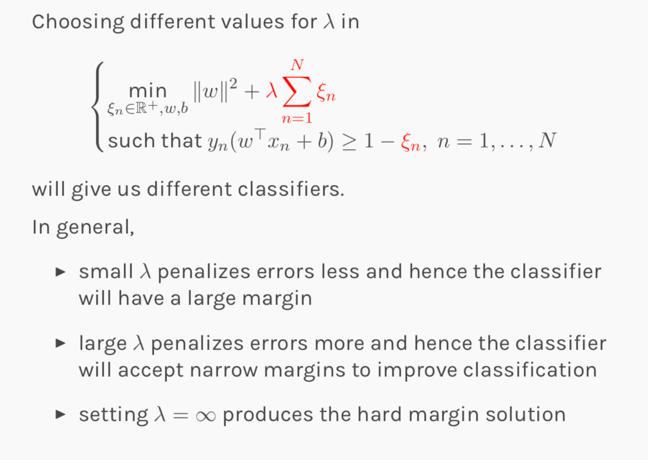
**SVM**:

find a decision boundary that maximizes the distance to both classes

hard margin: maximize the distance between support vectors while maintaining 0 misclassification rate

soft margin: allow for margin violation and misclassification by adding an error term in the optimization

kernels: polynomial, radial basis, sigmoid



**Decision tree:**

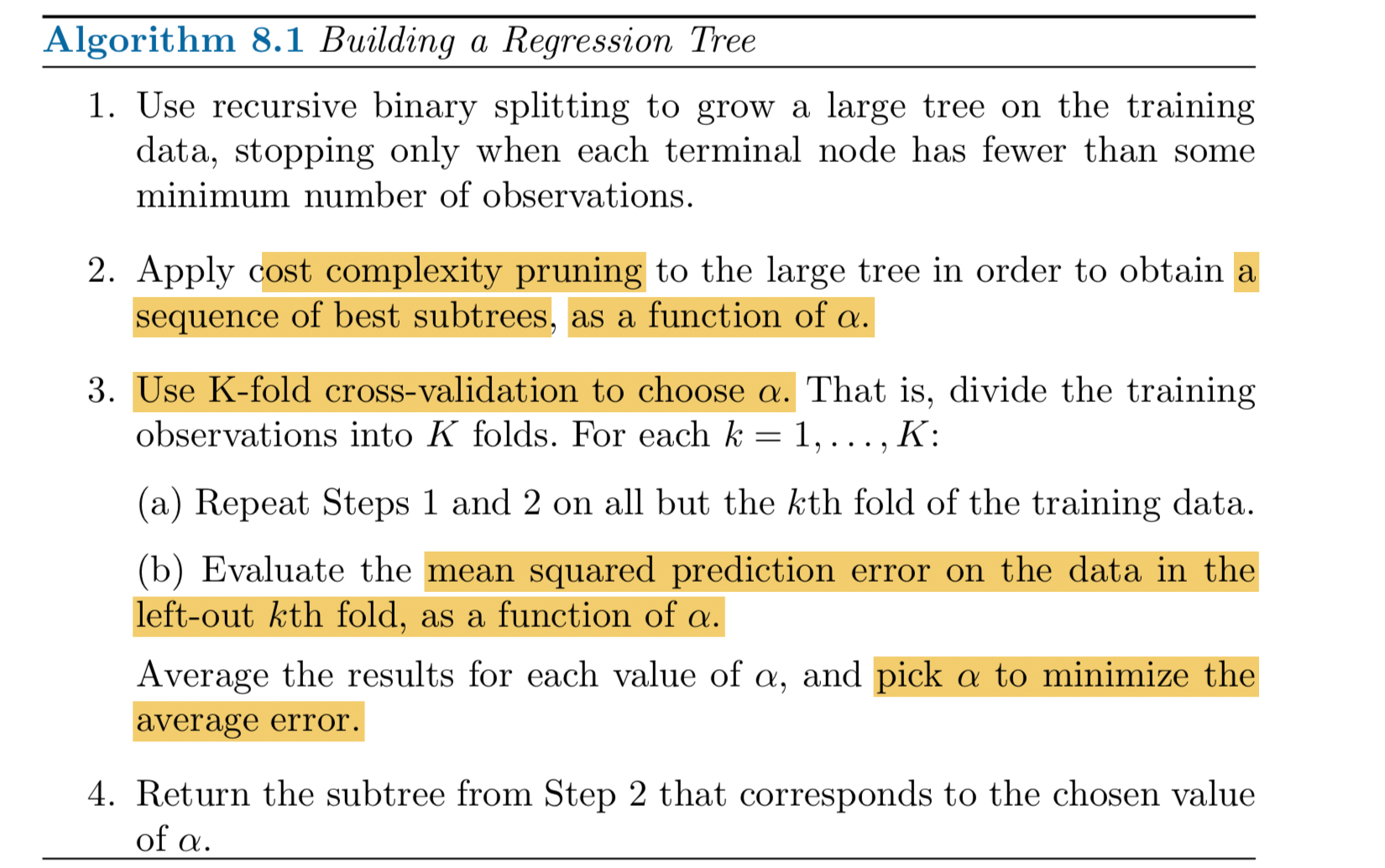
Recursive binary splitting: *all the features are considered and different split points are tried and tested using a cost function. The split with the best cost (or lowest cost) is selected.* cost functions try to find most homogeneous branches

Cost: classification error, gini index , entropy

Stopping condition: number of leaves, number of points in one region,

a *decision tree model* is one in which the final outcome of the model is based on a series of comparisons of the values of predictors against threshold values.

1. Start with an empty decision tree (undivided feature space)
2. Choose the ‘optimal’ predictor on which to split and choose the ‘optimal’ threshold value for splitting.
3. Recurse on each new node until *stopping condition* is met



Deep learning:

Deep learning refers to neural networks with multiple hidden layers that can learn increasingly abstract representations of the input data.

: how to use backpropagation and certain principles from neuroscience to accurately model large sets of unlabelled or semi-structured data. In that sense, deep learning represents an unsupervised learning algorithm that learns representations of data through the use of neural nets.

**Regularization:**

Model complexity: variable selection, remove collinearity, lasso, ridge, ensemble methods, tree pruning, PCA.

1. L1, L2 norm regularization on the hidden layers or the output layer
2. Dropout: randomly drop nodes on layers, ensemble technique
3. Data augmentation: shift or shear or rotate image data by transformations, increase sample sizes. Add more data

GMM:

A *mixture model* is a type of probabilistic model for unsupervised learning. The basic idea is the following: we assume that our data is generated by first sampling a category from some predefined set, and then sampling an observation within that category. That is, we sample a latent variable **z** according to a categorical distribution *p*(**z** = *Ck*; *θ*) = *θk*, and then sample an observation **x** from some distribution *p*(**x**|**z**).

Gaussian mixture models are widely used in data mining, pattern recognition,

view data as coming from a mixture over different components of a distribution. Each component will correspond to a class, or cluster. Use data D to estimate the parameters of this “mixture model.”

Given this, can predict the most likely cluster (or class) for each example. The latent variable is the group assignment for each point. The prediction of class assignment is the product of prior class probability and the class conditional distribution(normal). Since the labels are unseen,

(E step) Use the current parameters to predict the assignments zi, in particular, a distribution on the class for each example.

(M step) Update the parameters: maximize the “expected complete data log likelihood” using the predicted assignments.

Given current parameters, estimate “soft class assignments” ; maxi the expected complete data log likelihood;

Asymmetric random starts.

**Stochastic optimization**: loss function/or the quantity we need to optimize incorporate randomness. We alternate between updating parameters and taking the descent of loss function until convergency. For logistic regression. Start from a random start, iterate over data and modify weights/paramters until converge

A generative model learns the **joint** probability distribution p(x,y) and a discriminative model learns the **conditional**probability distribution p(y|x)

Topic Modelling.

As the name suggests, it is a process to automatically identify topics present in a text object and to derive hidden patterns exhibited by a text corpus.

unsupervised approach used for finding and observing the bunch of words (called “topics”) in large clusters of texts.

document clustering

[**fully connected neural network**](https://developers.google.com/machine-learning/glossary/#fully_connected_layer), which means that the neurons in one layer take inputs from *every* neuron in the previous layer

The ideal number of hidden layers and neurons depends on the problem and the data set. Like many aspects of machine learning, picking the ideal shape of the neural network requires some mixture of knowledge and experimentation. As a rule of thumb, increasing the number of hidden layers and neurons *typically* creates a more powerful model, which requires more data to train effectively.

The n\_classes parameter specifies the number of possible values that the neural network can predict.

The steps argument tells train to stop training after the specified number of iterations.

The number of steps to train is a **[hyperparameter](https://developers.google.com/machine-learning/glossary/" \l "hyperparameter)** you can tune.

Activation functions: relu, sigmod,tanh

Non-convex optimization: sensitive to initial state

Adding the layer and extra nodes produced more repeatable results. On each run, the resulting model looked roughly the same. Furthermore, the converged test loss showed less variance between runs.

Dropout regularization:

Randomly drop out neurons in a network for a single gradient step

backpropagation's failure cases

**Vanishing Gradients**

The gradients for the lower layers (closer to the input) can become very small. In deep networks, computing these gradients can involve taking the product of many small terms.

When the gradients vanish toward 0 for the lower layers, these layers train very slowly, or not at all.

The ReLU activation function can help prevent vanishing gradients.

**Exploding Gradients**

If the weights in a network are very large, then the gradients for the lower layers involve products of many large terms. In this case you can have exploding gradients: gradients that get too large to converge.

Batch normalization can help prevent exploding gradients, as can lowering the learning rate.

**Dead ReLU Units**

Once the weighted sum for a ReLU unit falls below 0, the ReLU unit can get stuck. It outputs 0 activation, contributing nothing to the network's output, and gradients can no longer flow through it during backpropagation. With a source of gradients cut off, the input to the ReLU may not ever change enough to bring the weighted sum back above 0.

Lowering the learning rate can help keep ReLU units from dying.

## **Dropout Regularization**

Yet another form of regularization, called **Dropout**, is useful for neural networks. It works by randomly "dropping out" unit activations in a network for a single gradient step. The more you drop out, the stronger the regularization:

* 0.0 = No dropout regularization.
* 1.0 = Drop out everything. The model learns nothing.
* Values between 0.0 and 1.0 = More useful.

Softmax: for multi-class NN, for single-type classification(each example only belongs to one class), probs of all classes add up to 1. Number of softmax nodes is equal to number of all classes.

One-vs.-the rest: multi-class, multi-labels classification, train a sequence of binary classifiers. Does not add up to 1.

a)      If you are sure that your data is outlier free and clean then go for SVM. It is the opposite -   if your data might contain outliers then Random forest would be the best choice

b)      Generally, SVM consumes more computational power than Random Forest, so if you are constrained with memory go for Random Forest [machine learning algorithm](https://www.dezyre.com/article/top-10-machine-learning-algorithms/202).

**c)** Random Forest gives you a very good idea of variable importance in your data, so if you want to have variable importance then choose Random Forest machine learning algorithm.

d)      Random Forest machine learning algorithms are preferred for multiclass problems.

e)     SVM is preferred in multi-dimensional problem set - like text classification

**The three main flavors of gradient descent are batch, stochastic, and mini-batch.**

**SGD:**

is a variation of the gradient descent algorithm that calculates the error and updates the model for each example in the training dataset.

SGD will keep overshooting.

when we slowly decrease the learning rate, SGD shows the same convergence behaviour as batch gradient descent, almost certainly converging to a local or the global minimum for non-convex and convex optimization respectively.

**Batch gradient descent** is a variation of the gradient descent algorithm that calculates the error for each example in the training dataset, but only updates the model after all training examples have been evaluated.

computes the gradient of the cost function w.r.t. to the parameters θθ for the entire training dataset:

As we need to calculate the gradients for the whole dataset to perform just *one* update, batch gradient descent can be very slow and is intractable for datasets that don't fit in memory. Batch gradient descent also doesn't allow us to update our model *online*, i.e. with new examples on-the-fly.

Batch gradient descent is guaranteed to converge to the global minimum for convex error surfaces and to a local minimum for non-convex surfaces.

One cycle through the entire training dataset is called a training epoch.

* Fewer updates to the model means this variant of gradient descent is more computationally efficient than stochastic gradient descent.

The more stable error gradient may result in premature convergence of the model to a less optimal set of parameters.

**Mini-batch gradient descent** is a variation of the gradient descent algorithm that splits the training dataset into small batches that are used to calculate model error and update model coefficients.

Implementations may choose to sum the gradient over the mini-batch or take the average of the gradient which further reduces the variance of the gradient.

Mini-batch gradient descent seeks to find a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. It is the most common implementation of gradient descent used in the field of deep learning.

* The batching allows both the efficiency of not having all training data in memory and algorithm implementations.

The model update frequency is higher than batch gradient descent which allows for a more robust convergence, avoiding local minima.

Batch size is a slider on the learning process.

* Small values give a learning process that converges quickly at the cost of noise in the training process.
* Large values give a learning process that converges slowly with accurate estimates of the error gradient.

Momentum

accelerate SGD in the relevant direction and dampens oscillations

by adding a fraction γ of the update vector of the past time step to the current update vector:

Stochastic gradient descent requires two parameters:

* **Learning Rate**: Used to limit the amount each coefficient is corrected each time it is updated.
* **Epochs**: The number of times to run through the training data while updating the coefficients.

**Tune deep neural network:**

* L2/1 regularisation
* Dropout
* Initialisation
* Batch normalisation
* Data augmentation: imaing data
* Ensemble methods: random initialization
* Early stopping: number of epochs, after the validation error stays stable
* Number of nodes, number of layers, activation function.

**Reverse a linked list**

1. Initialize three pointers prev as NULL, curr as head and next as NULL.
2. Iterate trough the linked list. In loop, do following.  
   do until current if NULL

*next = curr->next*

*curr->next = prev*

*prev=curr  
curr = next*

**How will you find middle element of a linked list in single pass?**

**Method 1:**  
Traverse the whole linked list and count the no. of nodes. Now traverse the list again till count/2 and return the node at count/2.

**Method 2:**  
Traverse linked list using two pointers. Move one pointer by one and other pointer by two. When the fast pointer reaches end slow pointer will reach middle of the linked list.

NLP:

In a typical [document classification](https://en.wikipedia.org/wiki/Document_classification) task, the input to the machine learning algorithm (both during learning and classification) is free text. From this, a [bag of words](https://en.wikipedia.org/wiki/Bag_of_words) (BOW) representation is constructed: the individual [tokens](https://en.wikipedia.org/wiki/Type%E2%80%93token_distinction) are extracted and counted, and each distinct token in the training set defines a [feature](https://en.wikipedia.org/wiki/Feature_(machine_learning)) (independent variable) of each of the documents in both the training and test sets.

Machine learning algorithms, however, are typically defined in terms of numerical vectors. Therefore, the bags of words for a set of documents is regarded as a [term-document matrix](https://en.wikipedia.org/wiki/Term-document_matrix) where each row is a single document, and each column is a single feature/word; the entry *i*, *j* in such a matrix captures the frequency (or weight) of the *j*'th term of the *vocabulary* in document *i*.

Typically, these vectors are extremely [sparse](https://en.wikipedia.org/wiki/Sparse_matrix). such dictionaries take up a large amount of storage space and grow in size as the training set grows.

The common approach is to construct, at learning time or prior to that, a *dictionary* representation of the vocabulary of the training set, and use that to map words to indices. [Hash tables](https://en.wikipedia.org/wiki/Hash_table) and [tries](https://en.wikipedia.org/wiki/Trie" \o "Trie)are common candidates for dictionary implementation. if the vocabulary is kept fixed and not increased with a growing training set, an adversary may try to invent new words or misspellings that are not in the stored vocabulary so as to circumvent a machine learned filter.

 applying a hash function *h* to the features (e.g., words), then using the hash values directly as feature indices

More sophisticated BOW representations use TF-IDF weights and/or n-grams instead of raw word counts

There are many choices concerning learning algorithms, but the most common suspects are Naïve Bayes, random forests, logistic regression and, increasingly, neural networks.

Feature hashing:

a) can deal with out-of-vocabulary words, b) doesn’t require us to retrain our models every time we encounter a new word or misspelling, and c) is as accurate as possible

map data of arbitrary sizes to data of a fixed size.

* Hash functions may output the same value for different inputs (collision).
* we’ll start by making a zero column vector with a huge number (say, 2²⁸) of elements for each of our training examples
* Next, we’ll choose a hash function *f*that eats strings and outputs values in the range [0, 2²⁸).
* we feed each word, one by one, through our hash function, and increment the value at the given index by one
* filter circumvention using out-of-vocabulary words.
* Notice that we just add 1 to the nth dimension of the vector each time our hash function returns that dimension for a word in the text.
* It is possible to use a 2nd hash function that returns +1 or -1 to decide if you add or substract from the vector. This will minimize collisions making a dimension of the vector taking a very large value.

Latent Dirichlet Allocation

assumes documents are produced from a mixture of topics. Those topics then generate words based on their probability distribution.

LDA is a matrix factorization technique.

LDA converts this Document-Term Matrix into two lower dimensional matrices – M1 and M2.

M1 is a document-topics matrix and M2 is a topic – terms matrix with dimensions (N,  K) and (K, M) respectively, where N is the number of documents, K is the number of topics and M is the vocabulary size.

Alpha and Beta Hyperparameters – alpha represents document-topic density and Beta represents topic-word density.

Number of Topics

Number of Topic Terms – Number of terms composed in a single topic. I

we will remove the punctuations, stopwords and normalize the corpus.

The results of topic models are completely dependent on the features (terms) present in the corpus. The corpus is represented as document term matrix, which in general is very sparse in nature.

Partial least square:

construct- ing predictive models when the factors are many and highly collinear.

emphasis is on predicting the responses and not necessarily on trying to understand the underlying relationship between the variables.

not usually appropriate for screening out factors that have a negligible effect on the response.

PLS has been applied to monitoring and controlling industrial processes; a large process can easily have hundreds of controllable variables and dozens of out- puts.

principal components regression and maximum re- dundancy analysis.

extracting latent vari- ables T and U from sampled factors and responses, respectively. The extracted factors T (also referred to as X-scores) are used to predict the Y-scores U , and then the predicted Y-scores are used to construct predictions for the responses.

* **Principal Components Regression (PCR):** The X-scores are chosen to explain as much of the factor variation as possible. This ap- proach yields informative directions in the factor space, but they may not be associated with the shape of the predicted surface.
* **Maximum Redundancy Analysis (MRA) (van den Wollenberg 1977):** The Y-scores are cho- sen to explain as much of the predicted Y varia- tion as possible. This approach seeks directions in the factor space that are associated with the most variation in the responses, but the predic- tions may not be very accurate.
* **Partial Least Squares:**
* seeking directions in the factor space that are associated with high vari- ation in the responses but biasing them toward directions that are accurately predicted.
* Ridge regression and neural nets are probably the strongest competitors for PLS in terms of flexibility and robustness of the predictive models, but neither of them explicitly incorporates dimension reduction--- that is, linearly extracting a relatively few latent factors that are most useful in modeling the response.

Discrete Singnal processing: Fourier, laplace, Z transform

Adam: Adaptive Moment Estimation