32. If Z=X+Y, X and Y are random variables, then Z and X are not independent? At first I thought the statement was true, but after thinking about its contraposition: if Z and X are independent, then Z can't be equal to X+Y where Y is another random variable. Set Y = Z - X and it is disproved.

4. MCMC Metropolis–Hastings sampling method, it is better to have but you don't need a symmetric step distribution

5. The idea of rejection sampling from Wiki, read it through and understood!! Great

6. Metropolis-hasting understanding: read the deduction section and understand!

46. Linear algebra:  
Q: What's the nature of a matrix multiplying a vector on the right?  
A: Imagine M\*x where M is the matrix and x is the vector. In this case, we want to find M\*x means, suppose the element in x is (x1, x2, ..., xn) and the vectors in M is (M1,...,Mn), what does the vector of x1\*M1+...xn\*Mn look like in the standard Euclidean basis. M is actually another basis where the first vector is the first base and second one is the second base....   
  
suppose a vector x lives in space M, then this vector's projection on standard Euclidean Space is M\*x   
  
Now it's clear that, if there exists a vector x and M\*x=k\*x where k is a constant, it means x (who lives in M)'s projection in standard Euclidean space is just a stretching version of x. Now x is called eigenvector and k is called eigenvalue. It is like the eigenvector is just the intersection of space M and the standard Euclidean space.  
  
check Math exchange favored question for more information and more understanding   
  
47. For any vector, always think of it as a linear combination of the basis vectors.  
  
48. Always keep in mind that an orthonormal matrix times a vector on the left like M \* x means just rotate the vector in space, and M \* x where x is the eigenvector of M means M can't rotate this specific vector x in space but only can change the scale of it.

52. Now baby you understand why the random variable's square can be deterministic! think about x: -1, 1, P: 0.5, 0.5

86. copula is just a joint CDF with uniform marginals --- the entire CDF exists in a unit cube.

88. If A = Q' \* Q where A is an k x k correlation matrix and Q is from any decomposition, and if a n x k matrix X comprises uncorrelated columns that have the same variances, the correlation matrix of Y = X \* Q equals A.

112. transformed beta distribution and it cannot fit a convex distribution when x in [0, 1]: f(x) ~prop~ (x / d) ^ (b - 1) \* (1+(x / d) ^ c)^(-(a + b) / c).

144. how to think about 2 matrix multiplication: left matrix is N row d column, right matrix is d row M column: left matrix contains d vectors, right matrix contains M vectors of weights. The result, which is a N x M matrix, its i th vector is the linear combination of the vectors in left matrix with the i th weight vector in the right matrix

163. PCA score is exactly the observation vector's coordinates in the new orthogonal coordinate system. If SVD gets you M=U\*D\*V', then U\*D IS the PCA scores, and each column of V is a unit base of the new coordinate system. Also, D is the square roots of the eigenvalues coming out of eigen-decomposition of the covariance matrix. See why centering before PCA is important: <https://stats.stackexchange.com/questions/22329/how-does-centering-the-data-get-rid-of-the-intercept-in-regression-and-pca/22331#22331>  
Remember, PCA essentially assumes some joint Gaussian distribution has the same covariance matrix of your data, and takes the ellipsoids' axes as the principle components.   
Given a test observation x and if you want to find its coordinates in the space spanned from columns of V, you do V y = x and y = V^{-1} x IS the new coordinates.

169. ICC from the classical formula (Fisher) is not equivalent to Pearson, but that from the random effect model is. let's say y=m + a(i) + e(i,j) from wikipedia, so the intraclass correlation can be derived as Cov(a(i)+e(i,u), a(i)+e(i,v))=(Var(a(i))+0+0+0)=Var(a(i)), so the correlation is Var(a(i))/(Var(a(i))+Var(e(i,j)))

170. a random vector IS a multivariate random variable

183. However, in practice it suffices to compute the SVD up to a certain precision

235. In Gaussian mixture (hard) clustering, a point is assigned to the kernel that gives the highest membership weight, namely, the i\_th kernel that yields the highest  a \* Gi(x). This has been examined.

238. The Gauss-Seidel method guarantees convergence for a symmetric and positive-definite matrix which is equivalent to the Gram matrix of a set of linearly independent vectors. Therefore, Gauss-Seidel guarantees convergence for training a healthy least square model.

239. In an optimization problem, a slack variable is a variable that is added to an inequality constraint to transform it into an equality.  
  
240. Modulo a power of two: n & (d - 1) where d is the denominator.  
  
241. In machine learning, when they say "distribution", just think of a population .. shit

251. Why it is not correct to use arbitrary distances: because k-means may stop converging with other distance functions.

253. The nice part of Kalman filters is that the product of two Gaussian PDFs is still a Gaussian PDF --- the posterior PDF, and this enables us to update everything in full parametric form.

259. Independence does not imply conditional independence! Learned from the Bayesian network:)

261. X'X is positive semidefinite if X has more columns than rows. A positive semidefinite matrix is singular!!!

263. Read Wikipedia's simulated annealing entry for the **actual insignificance about acceptance probabilities**!

303. Given random variables X, Y and their CDF F\_X and F\_Y, the rank correlation between X and Y is the Pearson correlation between F\_X(X) and F\_Y(Y).

=====================================

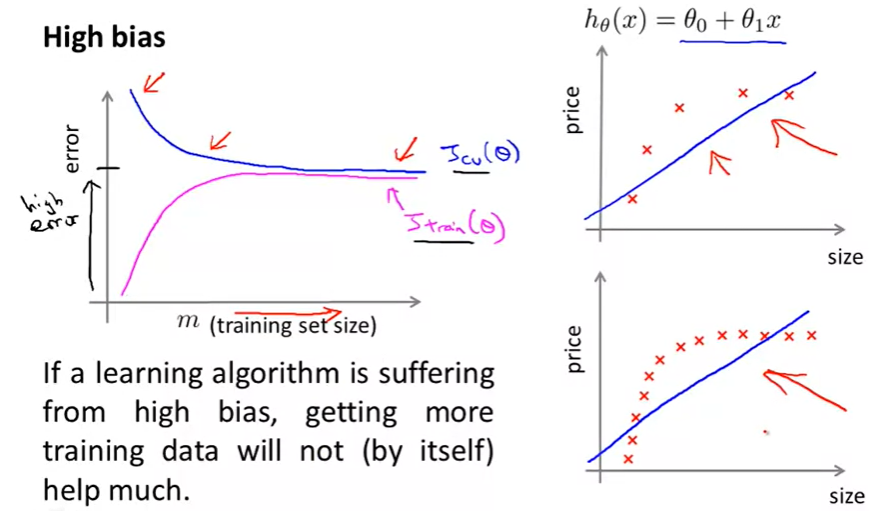
1. SDG stochastic gradient descent means learning from one example. It does not mean updating a random portion of the parameters.

2. Restriction bias vs preference bias.

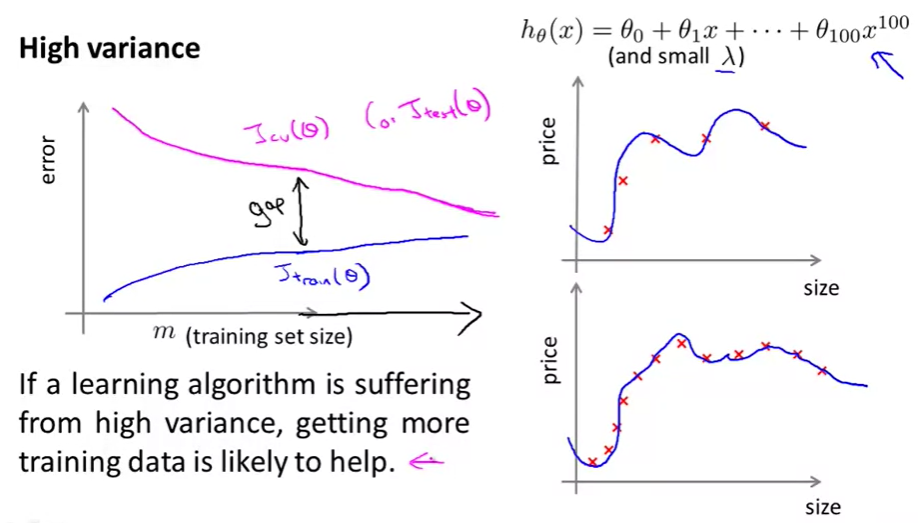
3. Check overfitting: the scatter plot of errors from the training set and the errors in the validation set where x-axis is the model complexity (number of parameters) and y-axis is the error.

4. k-fold cross validation: the average of k prediction errors measures the model.

5. Use a learning curve to identify if your model suffers from the problem of high bias or high variance.



High bias: given more training data, training error increases and then becomes flat, and the error is high. Meanwhile, cross-validation error decreases at first and then becomes flat.



The gap between cross-validation and training errors is large. Getting more training data can be helpful.

If the gap between training and cross-validation learning curves against training data size becomes steady near the end, it means you probably have realized the full potential of your model framework. It is a good thing if the errors are low, but a bad thing if the errors are high.

6. A categorical variable of k levels demands k - 1 dummy 0-1 variables.

7. When To Use Logistic Regression vs Support Vector Machine:

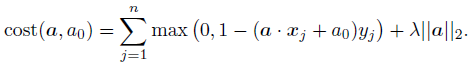
P = number of features,  
N = number of training examples  
(i). If P is large (1–10,000) and N is small (10–1000) : use logistic regression or SVM with a linear kernel.  
(ii). If P is small (1–1000) and N is intermediate (10–10,000) : use SVM with (Gaussian, polynomial etc) kernel

(iii). If P is small (1–1000), N is large (50,000–1,000,000+): first, manually add more features and then use logistic regression or SVM with a linear kernel

8. SVM regression is considered a nonparametric technique because it relies on kernel functions.

9. There are papers showing that random searching for hyperparameters is at least as good as grid-searching in training deep NN.

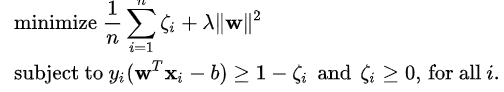
10. support vector machine svm:



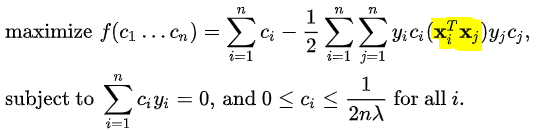
Why does greater lambda lead to less classification error? Because 2 / ||a|| is the margin between hyperplanes.

11. Subgradient: generalizes the gradient for non-differentiable function, e.g. piecewise linear function.

12. Kernel trick in svm: because the primal problem (convex)

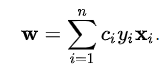


 has a dual in this form:



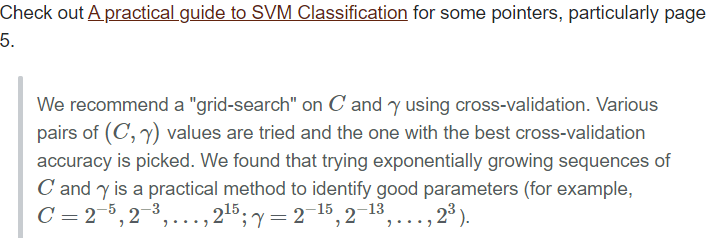
Solving the dual problem is equivalent to solving the primal thanks to the convexity. Here is where the kernel trick kicks in: the matrix in yellow can be replaced with any other suitable proximity matrix. Be aware of standardization of X before applying the kernel trick for certain kernels such as the polynomials.

One will find most c\_i s are zeros after solving it, and when we recover the weights W by



most elements in w will also be 0. Data points corresponding to nonzero w are then called **support vectors.**

13.



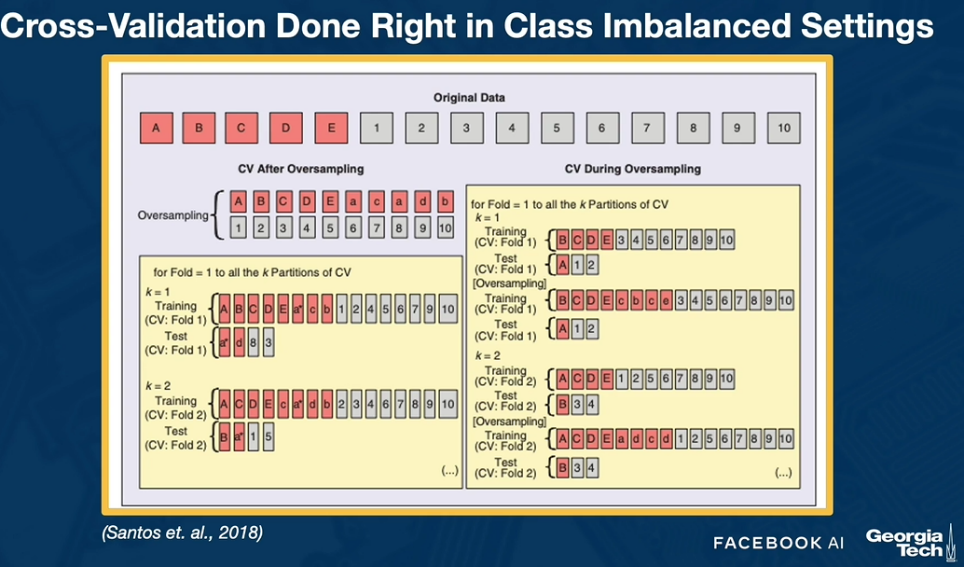
14. SVM with kernel tricks consumes huge memory. Remember X'X is explicitly stored.

15. Class-imbalance: SMOTE: Synthetic Minority Over-sampling Technique



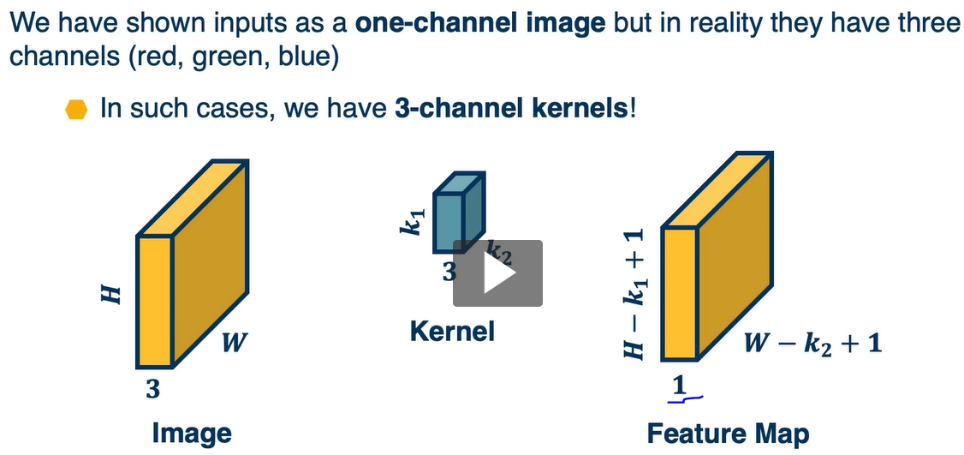
Select a data point from the K nearest neighbors of the data point of interest. On the hyperline between those two points, synthetize a new data point at random.

16.



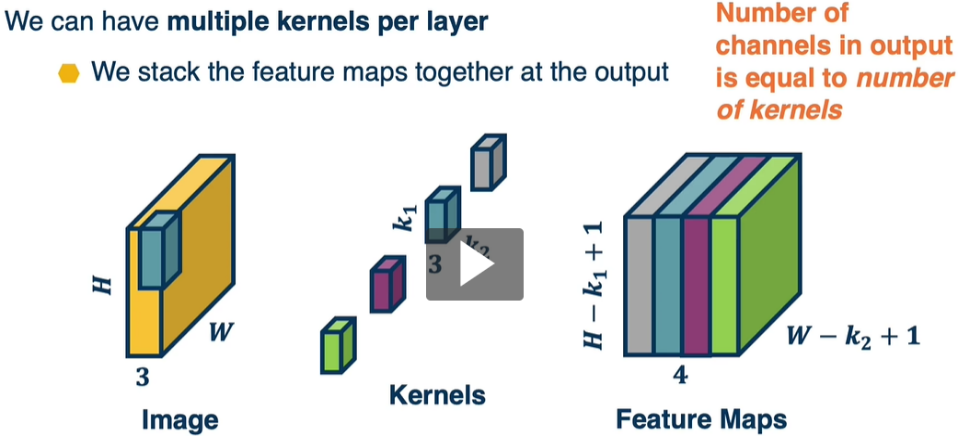
Remember oversampling should only happen in training data, not in the validation/test data.

17. Multi-channel convolutional layer:



Notice the feature map is only of depth of 1.

18.



Number of channels (depth) in output = number of kernels.

19. Leaky ReLU might be of great use!

20. Convolutional layer is nothing but a type of network connected using shared weights (the Convolutional kernel / filter). Abstractly, Convolutional layer -->+ nonlinearity --> max pooling layer ---> + nonlinearity convolutional layer --->... eventually we get a condensed image or several images (if we use multiple kernels to perform feature engineering.). Then we apply densely-connected nets on these images and stream them to the output classes.

21. Why drop-out works and Adam optimizer mechanism: see the slide.

22. Training set is to determine model parameters. Validation set is to determine model framework and hyperparameters.

23. PCA score is exactly the observation vector's coordinates in the new orthogonal coordinate system. If SVD gets you M=U\*D\*V', then U\*D IS the PCA scores,and each column of V is a unit base of the new coordinate system. Remember, PCA essentially assumes some joint Gaussian distribution has the same covariance matrix of your data, and takes the Gaussian's ellipsoid' axes as the principal components.

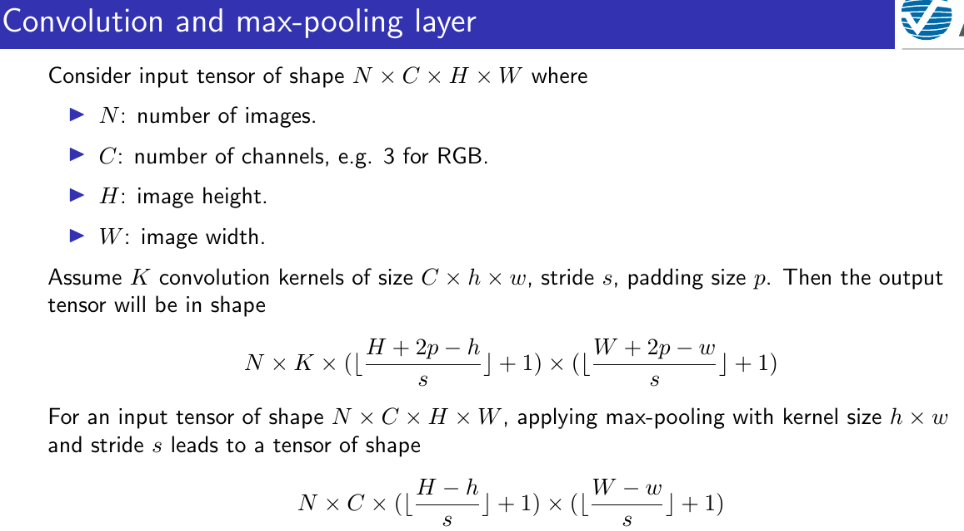
24. Remember, for ANNs, sometimes you should only shift each feature to 0 minimum, and then scale all elements by the global max. Scaling individual features could destroy informative internal structure. And it is always good to have input features be in [0, 1].

25. preference bias: out belief of what makes a good hypothesis. K-NN bias: locality, smoothness, and equally important.

26. Sparse Random Projections, fucking beautiful. <http://web.stanford.edu/~hastie/Papers/Ping/KDD06_rp.pdf>

Given data matrix A of dimension N x d, random projection produces B = AR where R is a random matrix of dimension d x k. Each row of R should have expected norm of 1.

27.



28. Some analyses show that global optimum could even have an overfitting problem.

29. One can understand why boosting (adaboost) will not lead to overfitting by considering the binary classification senario. As more boosting rounds are performed, the model accuracy may no longer go up, but the confidence of its predictions can keep increasing, because the boundary, drawn by the classifier, will keep being adjusted so that previous hard examples will move away from the boundary. The confidence getting higher can also be understood as the gap between the scores/probabilities output from the ensemble model getting larger. This gap coincides with margin in support vector machines. The margin operates as the regularization part of the objective to prevent overfitting. Greater margin weighs more in controlling the variance.

However, boosting tends to overfit if (i) data has pink (uniform) noise. Gaussian noise is white noise, (ii) the constituent weak learner tends to overfit data.

Weak learner only means it predicts the label better than random guessing (>0.5). There is no such a term as "strong learner". A learner with 0.9999 accuracy is still a "weak learner".

30. Restriction bias: the representational power of an algorithm; the hypotheses the algorithm will consider; what the model is able to represent.

31. Preference bias: what representation(s) a learning algorithm prefers, e.g. Decision trees prefer trees with fewer nodes.

32. Decision tree, restriction bias: the set of hypotheses that can be modeled by decision trees. Preference bias: fewer nodes and good splits near the top.

33. ANN, restriction bias: does not exist in the real domain. Any continuous function can be represented with 2 layers. An arbitrary function can be represented with 3 layers. Preference bias: less complex network, e,g, fewer and narrower layers; smaller weights. Networks with larger weights tend to overfit.

34. SVM, restriction bias: depend only on the proximity matrix between data points. The kernel characterizing distance between points must satisfy the Mercer condition. Preference bias: larger margin between the support vectors classified in different classes.

35. KNN: restriction bias: none as long as the distance between data points is computable. Preference bias: locality --- nearer points are similar; all features are equally weighted unless adjustments are made to the distance measure; the averaging of nearest neighbors assumes the function being modeled behaves smoothly.

36. Boosting: restriction bias and preference bias follow the underlying weak learners'.

37. Locally weighted regression is lazy learning.

38. It is known that hyperplanes in R-d have VC-dimension d+1.

39. Naive bayes: initialize nonzero probabilities --- smooth, inductive bias --- for labels that haven't shown up. This is to never trust the data too much, a kind of way of preventing overfitting. The inductive bias is that all things (combination of labels) are mildly possible. Naive Bayes is just the simplest Bayesian network: one root and then leaves nodes dependent on the root. Think about the spam email example.

40. Auto differentiation and computational graphs are more powerful than you think! In Pytorch, as long as you define a variable and start computing, its gradients are being tracked, and this can propagate almost every operation you may use, including min, max, ... absolute values...

41. How does PyTorch connect loss and optimizer?

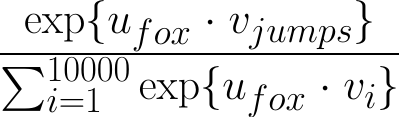
criterion = nn.CrossEntropyLoss()  
optimizer = optim.SGD(model.parameters(), lr = 0.001, momentum = 0.9)  
loss = criterion(model.forward(), target) # Compute loss.   
loss.backward() # Compute d(loss) / d(inputToLossFunction)  
optimizer.step() # Compute gradients of all parameters. Agent moves one step.  
The above code runs, but in the final 2 lines, shouldn't the optimizer need the gradients from loss to back-propagate? I did not see optimizers import any information from loss anywhere. How could this be possible?  
**Never mind. It should be loss = criterion(model(input), target). Consider loss as an object containing the network. loss.backward() performs back-propagation. optimizer.step() only updates the parameters.**

42. xgboost: for classification, given an example, if there are K trees, then the ensemble produces K scores in one or more classes. xgboost then applies softmax and decides the example belongs to the class of the highest probability, or, no, just by looking at the score is enough.

43. SLAM: simultaneous localization and mapping. Jay's answer is already very thorough, but just to answer my own question in a way that can be immediately understood by myself anytime, consider the matrix form of least square solution β=(X⊤X)−1X⊤y. The Omega matrix is exactly X⊤X and the Xi column vector is exactly X⊤y. The trick is, when each row of X contains an 1, a -1, and 0s for the rest, X⊤X can be directly constructed by the professor's method.

Although it seems to merely exploit a numeric incident, this trick is computationally significant. For large-scale linear regression, matrix multiplication X⊤X usually dominates the entire computing cost because inverting X⊤X is easy due to its positive-definiteness.  
Similarly, the weighted least square β=(X⊤WX)−1X⊤Wy corresponds to taking noises into the system.

44. word2vec good tutorial: <http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/>. The idea is to span each word in the dictionary, say which has 10000 words, into a high dimensional space, e.g. 300-dimensional, and use **two** 300-dimensional vectors (u, v) to characterize the word. Elements in these two vectors are model parameters to train. Given a training example, a word pair, (fox jumps), the learner computes the probability via softmax:



and tries to use gradient descent to maximize it, just like in a typical neural net.

45. Without negative sampling, the denominator of the above equation takes too long to compute. Negative sampling is to sample only a few, e.g. 20 v\_i that is not for "jumps". More details can be found in the tutorial.

46. A Gaussian variable has the largest entropy among all random variables of the same mean and variance.

47. ICA: Given X of dimension n x p, and given a target dimension k < p, fastICA decomposes X into the following matrices: K: p x k, W: k x k, A: k x p, S: n x k.

Let X be the centered version of itself (column means are 0, but fastICA will automatically shift X before decomposition),

then S = XKW.

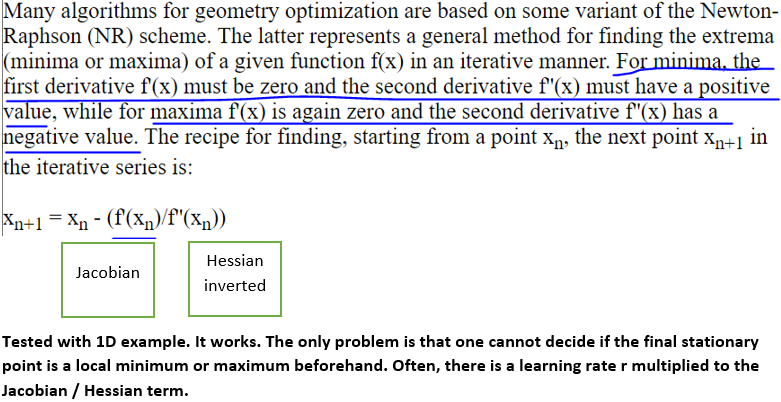
S's columns are the source signals and they have zero covariances. If k == p, then S == X.

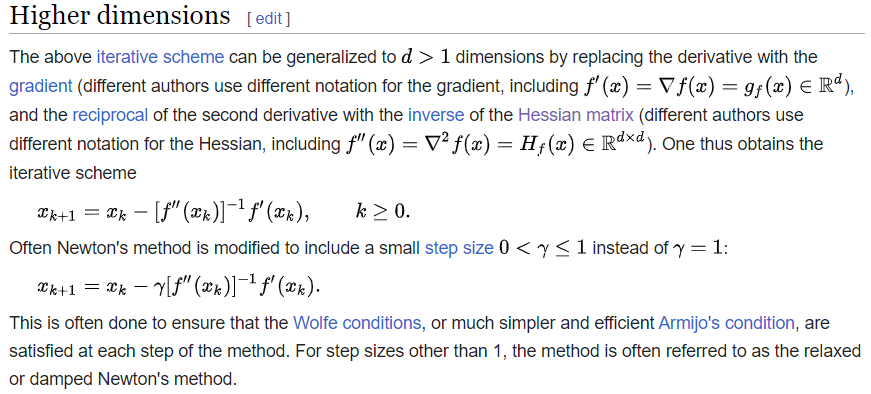
The fastICA algorithm takes the first k principal components of X and then creates S.

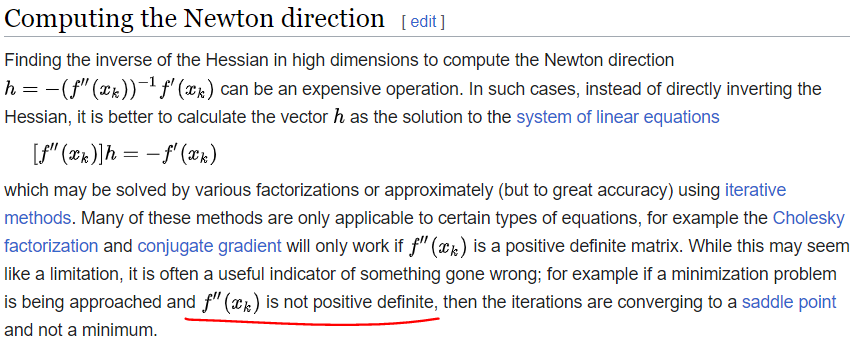
A's columns are not necessarily orthonormal. They are merely the mixing weights in order of X = SA. *After whitening the data (do PCA), ICA will “rotate the eigen axes” in order to minimize the Gaussianity of the projection on all axes (note that unlike PCA the axes do not have to remain orthogonal).*

48. If the signals extracted from a set of mixtures are independent, and have non-Gaussian histograms (kurtotic vastly different from 3) or have low complexity, then they must be source signals.

49. Finally, an excellent excerpt on **Newton-Raphson (NR) optimization:**<https://www.cup.uni-muenchen.de/ch/compchem/geom/nr.html> . This shows the strength and weakness of second-order optimization over the first-order.







**From another perspective, Newton’s method is essentially approximating the function via second order Taylor expansion. In every iteration, the Newton’s method looks at the quadratic approximation and jumps to the minimum directly.**

**Quasi-Newton’s method exactly means using finite differencing to approximate derivatives.**

**Now, think about using 3 very close points on the objective function to fit a surrogate quadratic function. Then, use the optimum of this surrogate quadratic function as the update. This is more or less the essence of the second-order optimization. Notice that the quadratic function could turn out concave. In this case, gradient descent might come to the rescue. Or, how about cubic fitting? Overall, this idea is named as Line Search:** <http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/optim/tutori5b.html> . It is a building block of quasi-Newton’s method.

50. Self-multiplying Markov matrix converges to a stationary matrix where rows are identical. Let T be the original matrix and let M be a copy of it. The following iterations are theoretically equal

(i) while(not converged) M = M \* M

(ii) while(not converged) M = M \* T

and (i) should converge much faster. However, because in (i), each M loses a little bit of numeric precision, eventually (i) will produce a zero matrix. The remedy is to normalize the rows of M in each iteration in (i).

51. On the wikipedia diffusion maps page, in the section "Diffusion process", if you are a little lost again, consider alpha = 0, which later the page states: "with alpha = 0 it reduces to the classical graph Laplacian normalization".

52. **Inconvenient truth**!: the reason why your Gaussian mixture R package is still not numerically stable is because you did not understand **LogSumExp**! Don't ever say the Guassian mixture is numerically hellish for high dimensional problems! And also, there are ways to prevent collapsing covariance matrices, e.g. add 1e-6 to the diagonal entries in each update. Remember, **when in doubt, read Python sklearn or related library source codes!**

53. Validation accuracy greater than training possible reason: you used dropout.

54. and **lend** themselves very easily **to** parallelization

55. Here, about the attention mechanism: {u\_0, ..., u\_N} are repeatedly used to multiply q1, q2, ... qi.

56. introduce any image-specific **inductive biases.**

57. Just think about matrix multiplications in Transformers, e.g. self-attention step, as projecting input vectors into lowD space as compact characterizations.

58. The entire research field of deep learning is to formulate better objective function.

60. Good refresher on Transformers: <http://jalammar.github.io/illustrated-transformer/>

61. Seismic shift from RNN, CNN to Transformers.

62. If there is no sufficient data in high dimensional space, then these data are more likely located on axes, and so the Gaussian mixture covariances easily collapse.

63. **Dynamic programming** is essentially finding the recurrence relationship, and then going from bottom up without recursion: tabulation rather than memoization.

If the problem is about 1D sequences, there are basically 3 framework:

1. Longest common subsequence (substring): find recurrence between T[i, j], T[i, j-1],  T[i-1, j], thinking in just 1 direction.  
2. Longest palindrome string: find recurrence between T[i, j], T[i+1, j-1], T[i+1, j], T[i, j-1], thinking in 2 directions meeting in the middle.

3. Chain matrix multiplication / popping balloons: think about any element in the sequence and consider the 2 subproblems without that element.

64. The metagraph of the strongly connected components of a directed graph is a DAG --- directed acyclic graph.

65. A strongly connected component (SCC) is a meta-vertex consisting of all vertices that have paths to one another. A sink SCC cannot reach any other SCC. A source SCC cannot be reached by any other SCC.

66. A fundamental graph problem: given a general directed graph, find all SCCs in topological order. The general idea is as follows:

  66.1. Find a sink SCC, remove it from the graph, repeat.

    66.1.1. First we want to find a vertex, v, that is guaranteed to be in a sink SCC, then we visit all vertices that can be reached by v. Those vertices and v constitute the sink SCC of interest.

    66.1.2: We do not try to find a source SCC because exploring from any member vertex could end up in another SCC.

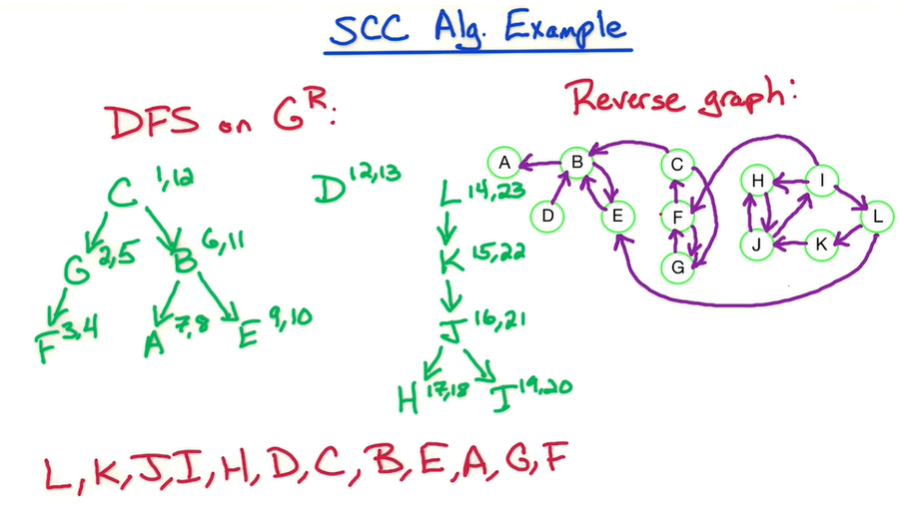
  66.2. From 66.11 we know the key to finding a sink SCC is to find a vertex guaranteed to be in a SCC.

  66.3. Recall that in DAG, the vertex with the lowest (highest) post-order number is a sink (source) vertex.

  66.4. In a general directed graph, the vertex with the lowest post-order number is not necessarily a sink vertex, but the one with the highest post-order number is definitely a source vertex.

  66.5. Thus we first reverse all edges in the general directed graph G of interest, find a source vertex by reading the post-order numbers. The vertex would be a sink in G.

  66.6. How to find the post-order numbers of all vertices in a general directed graph? By DFS:



Start from any unvisited vertex and visit all reachable vertices, and **continue**.

67. Algorithm for finding the SCCs of a general directed graph G:

1. Obtain RG by reversing all edges in G.

2. Compute all vertices' post-order numbers in RG using DFS. List the vertices in descending order: the first vertex has the highest post-order number and thus a source in RG. Denote the list by L.

3. In G, visit all vertices reachable by L[0]. These vertices are the first SCC. Remove them from L. Repeat 3 until L becomes empty.

68. Priority queue is just a min-heap.

69. The FFT of sequence ( a0, a1, … a\_{N-1} ) is the evaluation of A(x) = a0 + a1 \* x + … + a\_{N-1} \* x ^ {N - 1} at N points: W = { w^0, w^1, …., w^{N-1} } . Here w = cos(2pi/N) + i\*sin(2pi/N)

A(x) = Aeven(x^2) + x \* Aodd(x^2): this is the result from the property of W.  
Aeven(y) = a0 + a2 \* y + … a\_{N-2} \* y ^ {N / 2 - 1}   
Aodd(y) = a1 + a3 \* y + … a\_{N-1} \* y ^ { N / 2 - 1}  
Now A(-x) =  Aeven(x^2) - x \* Aodd(x^2), meaning half of W can be calculated “effortless”.  
  
The inverse FFT, (IFFT) of sequence ( a0, a1, … a\_{N-1} ) is the evaluation of A(x) = a0 + a1 \* x + … + a\_{N-1} \* x ^ {N - 1} at N – 1 points: w^0, w^1, …., w^{N-1} . Here w = 1 / [cos(2pi/N) + i\*sin(2pi/N)], or [cos(2pi/N) + i\*sin(2pi/N)] ^ {N – 1}.  
x = IFFT( FFT(x) ) / len(x)  
Root of unity: w ^ T = 1, w ^ {k + T} = w ^ k, w ^ {k + T / 2} = -w ^ k .

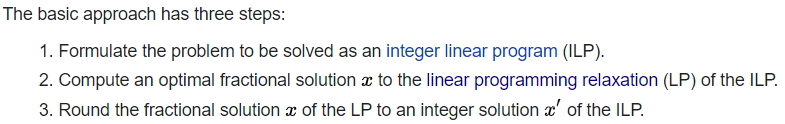
70. When you cannot understand a solution on Leetcode, just manually copy the code line by line, which is definitely helpful for you to understand.

71. (i) NP problems are search problems, that is, you can verify a solution in polynomial time. P is in NP except that finding the solution for a P problem only takes polynomial time.

(ii) NP-complete problems are the hardest problems in NP class. Knapsack problem is not NP-complete **by far** but NP-hard because the solution cannot be verified in polynomial time.

72. Monotonic stack / stack: when you want to take a fixed-size subset of an array such that: (i) the subset maintains the internal order in the superset, (ii) the subset contains the largest / smallest possible values.

73. A framework for solving NP-hard problems provided that they can be converted into ILPs:



The rounding step would take exponential runtime for an exhaustive search. So randomized rounding is typically employed. For example, 37.6 has 0.6 probability to be rounded to 38, 0.4 probability to be rounded to 37. Good evolutionary algorithms can be designed. Difficulty can arise when constraints are too harsh.

74. How to compute ceiling(x / y) when x and y are positive integers: (x + y - 1) / y.

75. Linear recurrence can always be parallelized! For example, x(n) = x(n - 1) + (n - 1) \* x(n - 2). One can derive x(n + 1) as a function of x(n - 1) and x(n - 2) and apply matrix multiplication:

76. In practice, in parallel computing, consider latency (time to establish communication) is 1000 times slower than communication (data transmission), and communication is 1000 times slower than computation (arithmetic operations on the single computer).

77. Using R, eigen decomposition of the covariance matrix produces the eigenvectors (in columns) and the eigenvalues. Multiplying the square roots of the eigenvalues to the eigenvectors are the axis vectors of the covariance contour.

78. Internally, XGBoost models represent all problems as a regression predictive modeling problem that only takes numerical values as input. If your data is in a different form, it must be prepared into the expected format.

79. When using xgboost's cross validation functionality, and if you want to supply your own folds, always supply both train and test folds! If you only supply the test folds, training and test errors will decrease to 0 !! It would be more convenient and also faster to just run 4/5-fold cross validation!

80. **When exactly can XGBoost get a better result than deep learning?** When the features are heterogeneous. When your data is made of heterogeneous columns such as age, weight, number of times the client called, average time of a call, etc then Xgboost is usually better than Deep Learning.

81. Gradient descent operates on the whole dataset. Stochastic gradient descent operates on a random sample of the dataset. Mini-batch stochastic gradient descent operates on a batch of samples of the dataset.

82. F1 score is good for a dataset that is highly imbalanced.

83. Gaussian process regression: consider a N x d training dataset with a label vector of dimension N x 1, and a testing dataset of dimension M x d with an unknown label vector of dimension M x 1. Columns are features. Gaussian process regression (i) treats the (N + M) x 1 labels as a random vector following an (N + M)-dimensional Gaussian joint, (ii) uses the (N + M) x d feature data to estimate an (N + M) x (N + M) covariance matrix, (iii) computes the M-dimensional Gaussian joint **conditioned on** the N x 1 known labels. The (N + M) x (N + M) covariance matrix can be computed empirically (except for d = 1), or by using a kernel function of any two rows. Means of the conditional distribution can be used as the point estimates for the M x 1 unknown labels, and confidence intervals can be constructed around the means. The idea is similar to Kalman filtering --- just a process of updating a Gaussian joint with evidence.

**The Gaussian process regression is just K-nearest-neighbors on steroids. Syntactically different yet semantically similar.**

84. Basic Bayesian optimization using Gaussian process for parameter tuning: (1) sample N points at random from the parameter space, (2) evaluate noises plus objective functions at these points, (3) sample K points from the parameter space, (4) compute the (N + K) x (N + K) covariance matrix using a kernel function, (5) compute the means at the K points conditioned on the N observed means, (6) from the K points keep the top L points that have the lowest conditional means, (7) let N <- N + L and go back to Step (2).

85. Convert sparse clustering problem to bipartite graph problem: an observation node is connected to a feature node if the observation has impact on that feature. First convert every observation so that they have 0-1 features. Next build a graph where observations are nodes, and nodes that share at least 1 or k features are connected. Next, run dfs/bfs to traverse the graph. This could reveal the disjoint subgraphs of the graph. Then run clustering on the subgraphs would be easier due to the smaller sizes.

Simpler term: one-mode projection

86: Graph: given an adjacency matrix A of 0 and 1, (A ^ k)[i, j] equals how many ways can one walk from node i to node j in k steps.

87. Pareto distribution vs. Exponential distribution. They are not the same. Pareto has a substantially fatter tail. The degree distribution in most real-life networks follows Pareto distribution.

88. Tree-based models are immune to multicollinearity. <https://datascience.stackexchange.com/questions/31402/multicollinearity-in-decision-tree>

89. Confirmed via reading source code: <https://machinelearningcompass.com/machine_learning_math/subgradient_descent>

In the example of the absolute value, subgradient at 0 is just any gradient between -1 and 1. In the actual training algorithm, the gradient is set to -1 for negative values and 1 for positive values. And there is a threshold, e.g. 1e-6 to zero the target coefficient if it falls under the threshold.

90. Boosting in regression is easy to understand. The key moment in boosting for classification is using one iteration of Newton's method to approximate the loss function for the new tree. The new loss function is independent of the loss of the existing ensemble, and is only dependent on the first and second order derivatives of the leaf nodes. The new tree is built in a greedy manner guided by the loss function: find the splitting point that maximizes the loss reduction.

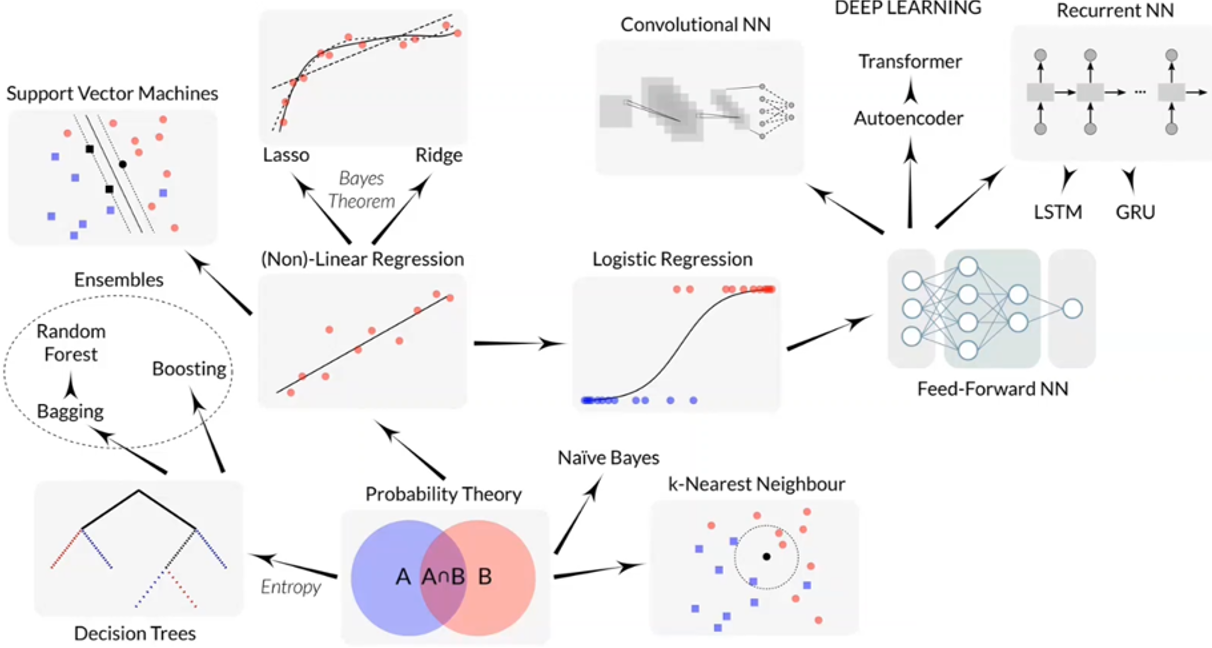
91. XGBoost weighting data can only be set in xgb.DMatrix, info = list(weight = ...)

92. The reason why L1 norm is often called "robust", is because it relates to the quantiles of the data distribution. Consider 1D data, the point that has the least sum of L1 distances to all the other points is the **median**

93. There are more ways of linearly transforming your data than just PCA! You can do discrete cosine transform: in this case, every row of the N x P matrix is a signal living on a P-point grid, and it is a mixed sum of P cosine waves in different frequencies. Weights on these cosine waves would be your new data.

94. For Mx where M is a N x P matrix and x is a p x 1 vector, think of the linear transformation as projecting x in P-dimensional space into N-dimensional space. Always remember that a fully-connected neural net before nonlinear activation is essentially matrix multiplication.

95.



96. The nature of transfer learning is better initialization. When trained on large datasets, the parameters, e.g., weights in convolutional kernels are in a great initialization, so continuing training the model for specialized tasks will produce better results than training from scratch, and will probably converge faster.

97. Q-learning at its simplest stores data in tables. This approach **falters** with increasing numbers of states/actions since the likelihood of the agent visiting a particular state and performing a particular action is increasingly small.

98. Cross entropy is essentially negative weighted log likelihood: to measure the loglikelihood of a PMF against an underlying continuous distribution. One can then apply maximum likelihood principle to it.