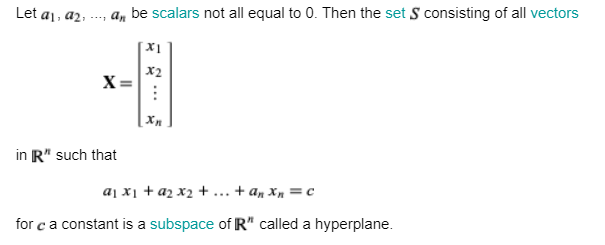
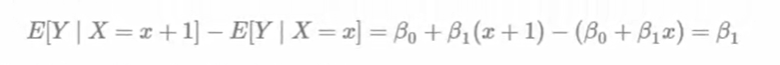
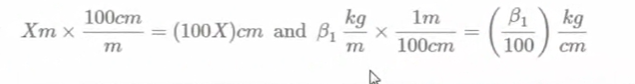
1. Curse of dimensionality:
   1. When a measure such as a Euclidean distance is defined using many coordinates, there is little difference in the distances between different pairs of samples.v One way to illustrate the "vastness" of high-dimensional Euclidean space is to compare the proportion of an inscribed hypersphere with radius r and dimension d, to that of a hypercube with edges of length.
   2. Another effect of high dimensionality on distance functions concerns k-nearest neighbor (k-NN) graphs constructed from a data set using a distance function. As the dimension increases, the indegree distribution of the k-NN digraph becomes skewed with a peak on the right because of the emergence of a disproportionate number of hubs, that is, data-points that appear in many more k-NN lists of other data-points than the average. This phenomenon can have a considerable impact on various techniques for classification (including the k-NN classifier), semi-supervised learning, and clustering,[16] and it also affects information retrieval.
   3. the typical formalizations of the curse of dimensionality affect i.i.d. data, having data that is separated in each attribute becomes easier even in high dimensions, and argued that the signal-to-noise ratio matters: data becomes easier with each attribute that adds signal, and harder with attributes that only add noise (irrelevant error) to the data. In particular for unsupervised data analysis this effect is known as swamping.
   4. Using cosine distance instead of Euclidian distance
2. Hyperplane:

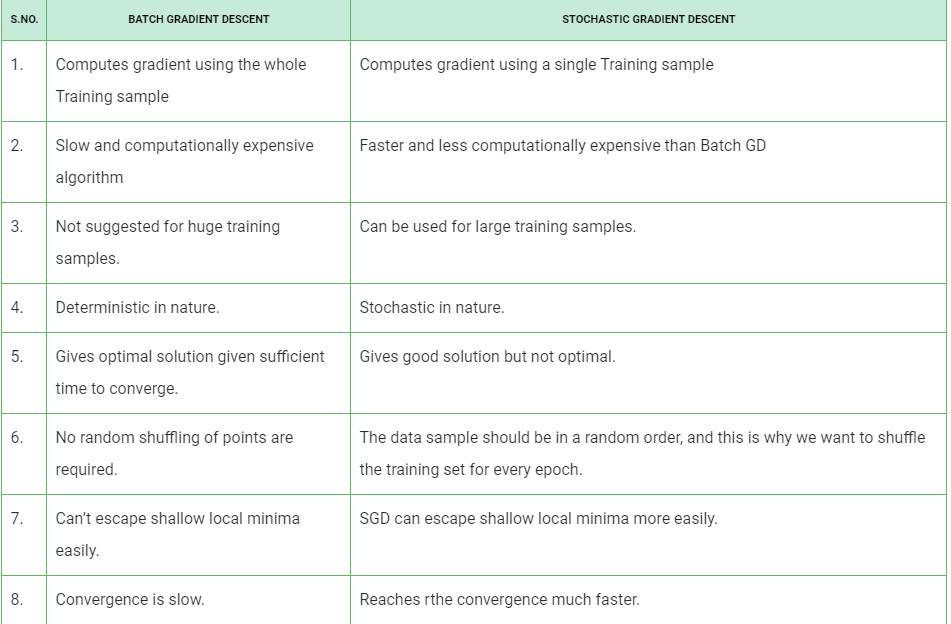
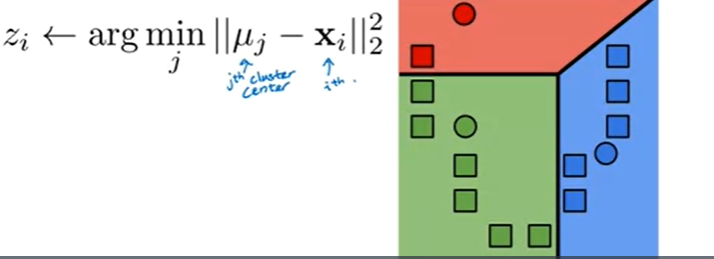


1. Interpreting coefficient:
   1. Changing X by 1 will be equally to B1



* 1. Multiplication of X results is dividing the coefficient be a factor a



1. In a linear regression, the number of parameters is equal to the number of input variables plus one for the bias.
2. In words: for the k’th model, the k’th fold acts as a validation set.The estimated prediction error from CV ErrCV is the average of the test set prediction errors of each model.
3. Batch gradient descent computes the gradient using the whole dataset. This is great for convex, or relatively smooth error manifolds. In this case, we move somewhat directly towards an optimum solution, either local or global. Additionally, batch gradient descent, given an annealed learning rate, will eventually find the minimum located in it's basin of attraction.
4. Stochastic gradient descent (SGD) computes the gradient using a single sample. Most applications of SGD actually use a minibatch of several samples, for reasons that will be explained a bit later. SGD works well (Not well, I suppose, but better than batch gradient descent) for error manifolds that have lots of local maxima/minima. In this case, the somewhat noisier gradient calculated using the reduced number of samples tends to jerk the model out of local minima into a region that hopefully is more optimal.
5. 
6. K-mean:
   1. k-means algorithm assumes this simple definition of score that we described before where you just measured the distance between the observation, and the cluster center as a way of associating data points to specific clusters.
   2. And the first step of the algorithm is to initialize our cluster centers. So these circles are going to represent our cluster centers and we're going to notate the cluster centers with the Greek letter mu, so we have mu 1, mu 2 and mu 3. Here we're looking at an example with k = 3 clusters. So we have three clusters centers.
   3. And then having to find those cluster centers, the next step of the algorithm is to assign observations to the clusters based on the minimum distance to the cluster center. So in equations, that's written right here where this is our jth cluster center and xi is our observation. This is the cluster label for observation i. So remember, this is going to be our inferred cluster label for the ith observation, whereas in supervised learning we were assuming we had a given label yi. But what we're doing here with this function is this thing called arg min. So let's describe what arg min does. So arg min is going to return the index j, Of the cluster whose center is closest to observation Xi And this is contrast to min, Whereas if we think about minimizing a function, returns the value of the objective, which would be minimum value of this distance calculation we're doing here. So just to be clear by saying, arg min we're returning which index as we're searching over all cluster centers muj, mu1, mu2, mu3, with the ith observation fixed. And varying, This cluster center.
   4. 
   5. This little, Partitioning of this 2D space into these different colors is called a Voronoi tessellation. And what it represents is that, if any point falls into, for example, this blue region here, that means that the distance to this cluster center is
   6. smaller than it is to the distance to any of the other cluster centers. So this is the region in which any observation here will get assigned to this blue cluster. And any observation here will get assigned to the green cluster and finally, any observation here will get assigned to the red cluster.
   7. And then the next step in the k-means algorithm is we're going to update the positions of the cluster centers based on the points that were assigned to them. In particular we're going to compute the mean of the data points in that cluster or equivalently, we can think of this as the center mass of these data points.
   8. the last stage of k-means is to repeat the last two steps that we did. So we're going to keep iterating between assigning observations to cluster centers, then updating the cluster centers and going on and on and on until convergence,
7. Momentum:
   1. Momentum is where we add a temporal element into our equation for updating the parameters of a neural network – that is, an element of time. This time element increases the momentum of the ball by some amount. This amount is called gamma γ, which is usually initialized to 0.9. But we also multiply that by the previous update vt.
   2. For each time we roll the ball down the hill (for each epoch), the ball rolls faster towards the local minima in the next iteration. This makes us more likely to reach a better local minima