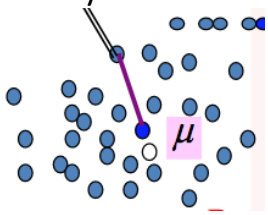
CSCI5150 Midterm Thesis

Zhang, Tuxin
1155101156

Machine learning is the field or subject that researches how to make computers learn. In other words, a machine learning algorithm is an algorithm that runs in the computer that trains themselves so that human don't need to explicitly describe the task human want to achieve.

To understand how machine learns, we first need to understand what is law(规律). Law has characteristics such as rhythmical, constancy, unadorned, impersonality. Everything in the world are constrained by law. But once we are familiar with law, we can actually learn some pattern or knowledges from it.



(For simplicity, we will assume our discussion are in 2-D space. But it can be extended to N dimensional space easily).

But how exactly can computer learn? A data point (a sample) is the key element of how machine is learning. A point (can be treated as a sample in realistic) is very useful. A point can represent a point, even a cluster (set of points). In vector form it can also represent a line, even a plane. Easily speaking a point can represent a structure. Start from the beginning, if a point representing a point, easy, it is what a point supposed to do. But in the cases that a point is representing a structure other than point, it becomes complicated. How to decide which is the "best" point to represent let's say a cluster (set of points)? How to find, this is called implementation, we need algorithm. Obviously, we want the point with the lowest error. The lower the error, the better the point is. The next question pops up is how to compute the error? What kind of error measurement method we want to use. Actually, learning is just to decide which theory (best theory, lowest error) to use, what is your learner(model), and finally implementation (code the algorithm). The way we are trying to find the optimal point to represent the cluster is also kind of learning.

Common error for points and points are square error and absolute error. Although we have discussed this in the assignment, but these two errors are quite different. And these two errors should be applied only in correct place. Square error is measured by so called Euclidean distance, basically means how far are these two points is there error. Absolute error focus more on the "coordinate error" which is the difference inside the coordinate system.

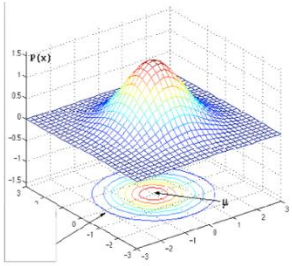
平方误差与高斯分布

$$e_t = x_t - \mu, \quad (x_t - \mu)^2 \Leftrightarrow p(x_t, \theta) \propto e^{-c(x_t-\mu)^2}, c > 0 \text{ Gaussian}$$

$$G(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}$$

where σ^2 is the variance, and μ is the mean value.

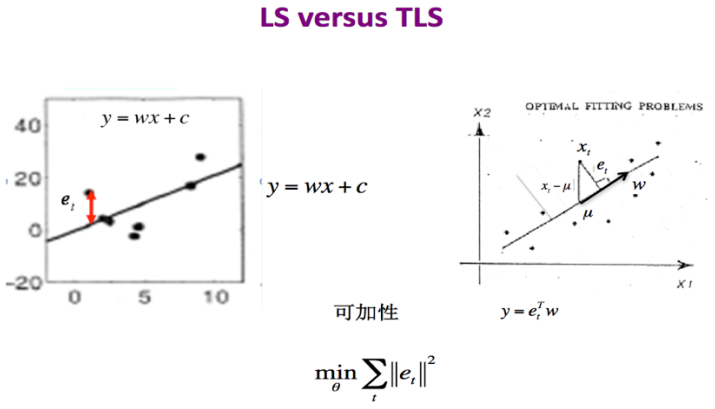
Assuming all points are independent and squared error are used, we learned that the mean point μ is the optimal (lowest error) point (can be derived from MLE). The error are added up together, importantly, not all error can be added together to find the minimum. The error can added up together if and only if the points are independent. We can also treat error in another way, how to find the better point, by the error, but we know the very large error appear less, small error appear a lot. Then we turned this into probability now. Probability can then be develop further. That it, Gaussian distribution, set of points can have many different distribution, the most famous one is Gaussian distribution. If a cluster's points are Gaussian distribution, the mean point μ we just computed and squared error method are related to Gaussian distribution. By central limit theorem we know that the points tend to close to mean with high probability, far from mean point with lower probability. From MLE we can obtain the formula to compute the variance. Together with all parameters we can obtain the completed Gaussian distribution formula and model. (MLE is a method of estimating the parameters of a statistical model such as Gaussian distribution, given the observation, in our case, observation are points. MLE attempt to find the parameters that maximize the likelihood function, minimum error can be viewed as largest likelihood, which means theory can be talk in error manner or likelihood manner).



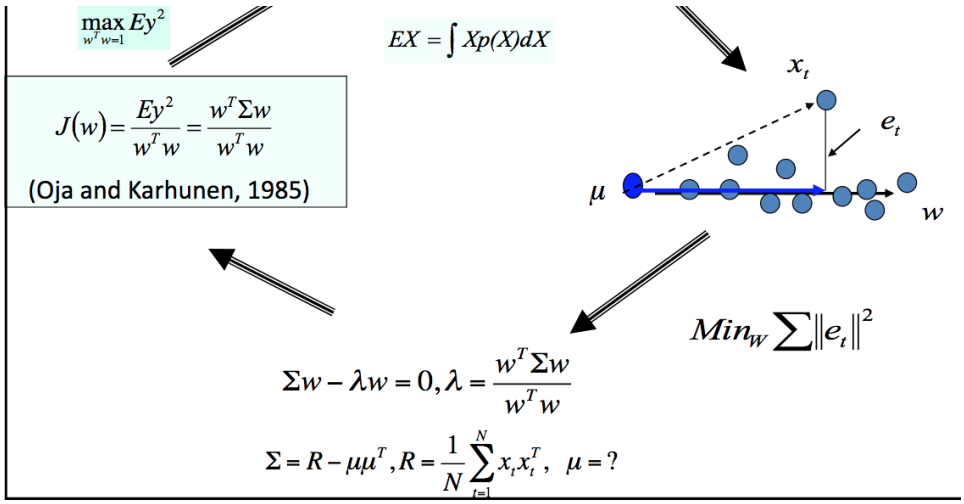
OK, what if use a point to represent a cluster of points is not enough, we want something more. We can further improve to use a line to represent a structure(etc. data points). First thing we need to understand is that how to measure error between points to line. There are two main kinds of error measurement method we have discussed in the assignment. The first one is called Least square error. Obviously, this measurement only measure the error by vertical residuals. However, the other one, Total least square error is measured by diagonal residuals. For instance, when line y=mx+c, point at (x_i,y_i). error = $e_t = y_i - mx_i - c$ (vertical resduals). when line ax+by+c, point at (x_i,y_i). error = $e_t = |ax_i+by_i+c|/(\sqrt{a^2+b^2})$ (Total least square used a.k.a "error in variables" linear regression). Two approaches to line fitting differ statistically in that Least square treat on variable as "no error" and we want to minimize the error of another

variable, while Total least square treats uncertainty in all variables. Total least square is similar to PCA, and is essentially fitting a multivariate Gaussian joint distribution p(x,y)p(x,y) to the data (in 2D case).

1. If your goal is to constrain the distribution of y_i given a precise value of x_i, then the least square curve is the solution.
2. If your goal is to constrain the "independent components" of the 2D (x,y)(x,y) data, then Total least square is better. For example the first principle component may have a common cause, in terms of the system dynamics.

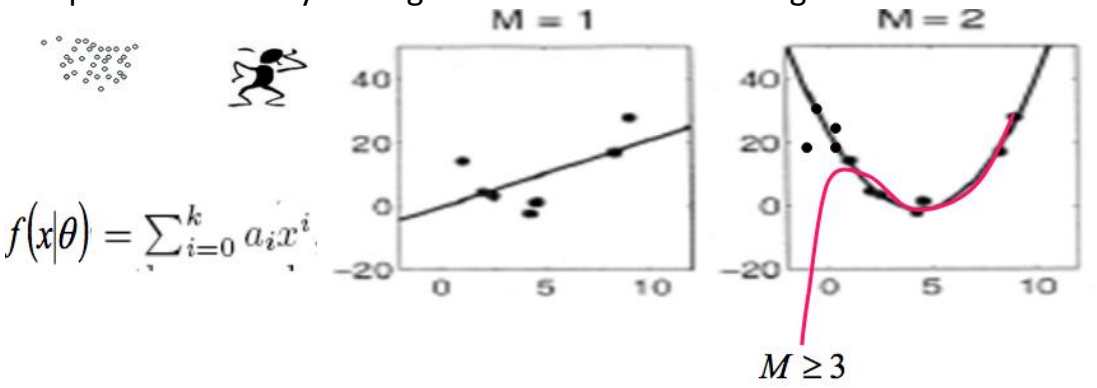


As mention above, if we want to use a line to represent a cluster of points. PCA uses Total least square error. The image below shows meaning of PCA in professor's slides.



To think deeper, Principle Component analysis's goal is actually trying to find the principle component, which we talked about is actually the vector(or line) that can represent the cluster of points with minimum error $Min_W \sum \|e_t\|^2$ (total least square). As professor explained, it corresponding to the eigenvector with the largest eigenvalue. There can be many eigenvectors but only the "principle component" is the best line to represent the points cluster (Minimum error). Minor component analysis (MCA) is the exact opposite of PCA. The minor

component is exactly the eigenvector with smallest eigenvalue in PCA.

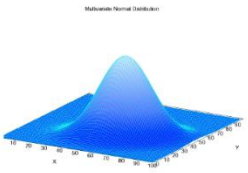


Also, the line can be non-straight. This is called linear regression. This is also the problem of fitting a line to data points and tries to minimize the error (not always). It is like we have many high dimensions points and we tried to fit a line to minimize the error. Each "dimension" in a point is actually representing a feature (variable), of each of feature is independent we can solve it just like solving linear equations. But if features are related, it become a little bit complicated. Not only the solving method needed to be changed, but also the error measurement method. I mentioned above that "not always minimize the error". The reason is that is we are fitting the line to the data points too well, it might cause overfitting. It's like we are fitting the data too well, if we want to do prediction using this line, it will work poorly, the prediction will be far away from the true value since this line is like perfectly for training data, not for future data. In this situation, it's actually remembering all the data, this is no longer called learning, it's remembering. Since PCA and linear regression are both learning model, we prefer to let it "learn" instead of "remember".

$$G(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} \mid \boldsymbol{\Sigma} \mid^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix, and $\boldsymbol{\mu}$ is the mean vector.

d is the dimension.



$e_t = x_t - \mu$

$\min_\theta \sum_t \| e_t^T \Sigma^{-1} e_t \|^2 \quad \longleftrightarrow \quad \max_\theta \sum_t \ln G(e_t \mid 0, \Sigma)$

Mahalanobis Distance

$\mu = \frac{1}{N}\sum_{t=1}^{N} x_t \qquad \Sigma = \frac{1}{N}\sum_t e_t e_t^T$

Furthermore, if we want to use circle/ellipse to represent data points instead of lines, it will complicate our algorithm a step forward. In case of circle, if we want to measure the error between a point and a circle, first we need to find the tangent point of the circle with respect to that data point, and then calculate the error. Also represent a circle is not as easy as represent a point, line or plane. Equation of circle contains two "square". Make us not that easy to represent it

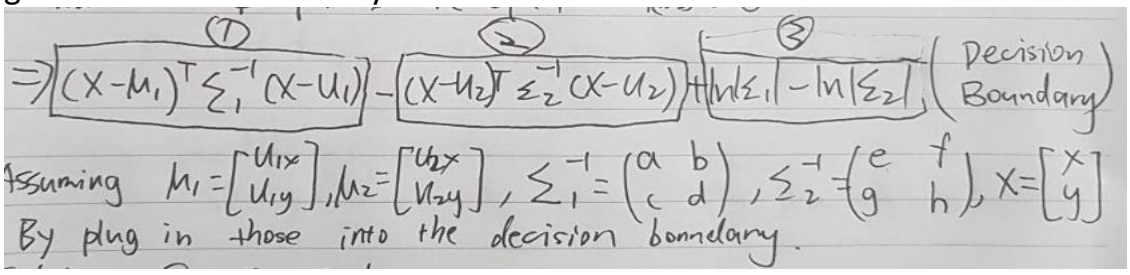
and calculate the error between it and the data point. Moreover, ellipse is even more complicated them circle. Suppose the error is the distance, it is quite complicate to find the distance between a point to an ellipse, and we still need to be minimizing it. And the equation of ellipse is more complicated then circle. This is why everything become more complicated when we need to represent data points using circle or ellipse.

With some little modifications, we can use a plane to represent data points. Basically, a plane is represented by a normal vector or two vectors, not very much different from line representing points. And different error measurement method is applied when different representation of a plane is used. If the plane is in one normal vector form, simple p.v project the vector on to the normal vector to see the distance (error). We can do similar projection even with two vectors representing the plane.

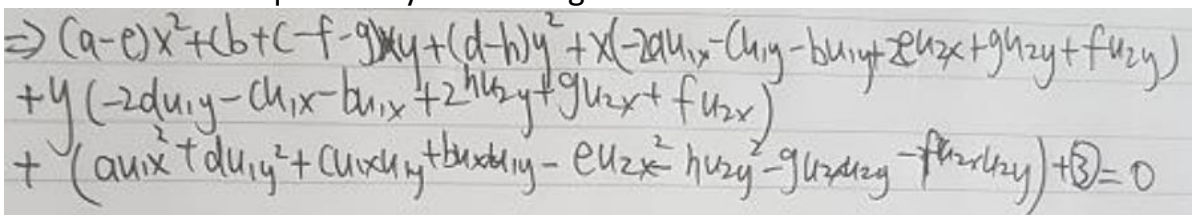
The Gaussian distribution I talked above is univariate. In practice multivariate Gaussian distribution is a generalization of the 1-D (univariate) distribution to higher dimensions. It is very similar to univariate, except x, mean, co-variance become vectors and matrix. Obviously, if a set of points are distributed under Gaussian distribution, surely, we can use Gaussian probability to represent a set of data points. Furthermore, we can make classification between different Gaussian distribution. Here is the discriminant function using the parameters in multivariate Gaussian distribution.

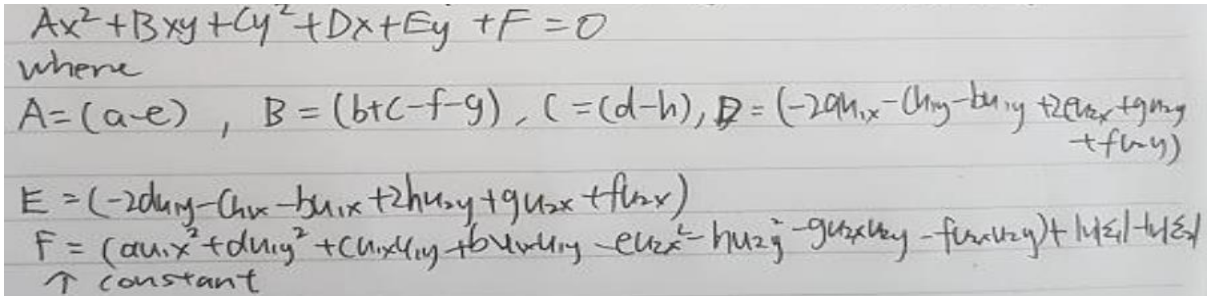
$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(w_i)$$

By setting $g_i = g_j$ (Two discriminant functions of two distribution). We can easily get the decision boundary.



Solve the above equation by the settings.



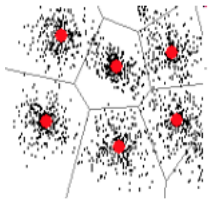


Finally, compute the determinant and discriminant.

- if $B^2 - 4AC < 0$, the equation represents an ellipse;
  - if $A = C$ and $B = 0$, the equation represents a circle, which is a special case of an ellipse;
- if $B^2 - 4AC = 0$, the equation represents a parabola;
- if $B^2 - 4AC > 0$, the equation represents a hyperbola;
  - if $A + C = 0$, the equation represents a rectangular hyperbola.

Gives us the shape of the decision boundary.

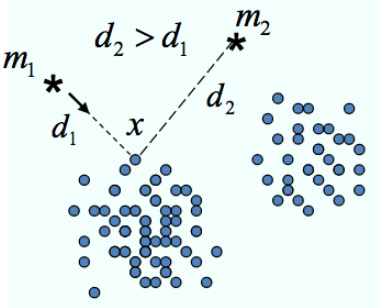
After done with above “one object representing a structure (points)”. What if there are not only one cluster of points? Or there are many sub-clusters inside one cluster? Use one object to represent it might cause large error, or we can simply represent it with multiple objects.



This is actually back to the beginning, using one point to represent a cluster (many points). We use many points to represent a big cluster. But inside it, each sub-cluster is represented by one single point. And the error measurement is just the summation of each sub-cluster. As we learned and discussed, using square error the optimal point of a cluster is the mean point. Since now we have k clusters, then this algorithm is called K-means. That it, in each sub-cluster, we use its mean point to represent it. But how can we find such sub-clusters and such mean points. This is the implementation of K-means. Since this algorithm is highly depends on initialization, we will assume that all initialization will be relatively fine. Then we can iterate through the algorithm to find the optimal solution.

```
1 : procedure K-MEANS(C, X, d, N, K)
2 :     while not convered do
3 :         S_k ← {x : k = argmin_i ||c_i − x||_2, x ∈ X}
4 :         Σ_k ← Σ_{x∈S_k} x
5 :         κ_k ← ||S_k||
6 :         c_k = Σ_k / κ_k
7 :     end while
8 :     return C
9 : end procedure
```

This is kind of similar to an algorithm called EM (Expectation Maximization Algorithm). We can think of one point belongs to some cluster j, then this j is the latent variable in EM algorithm. Then we will cluster this point to j if this point belongs to j with highest likelihood.



This is actually related to something called “competitive learning”, we can see that k points are actually compete for the data points. Every time a data point comes in, they will fight for the ownership for that point. What if the initializations are really bad, result in one point is always winning, eventually this specific point will represent k clusters and leave all others points trivial. Frequency sensitive learning is to prevent a point from always or high frequent winning. Also rival penalized competitive learning can be used when the initial number of K is bad, It will tries to “Kick” the bad point away so only the “useful” number of K point gets to represent a sub-cluster.

Furthermore, if we are not satisfied with representing K clusters with K points. We can do K-lines which only required a little modification on K-means:

1. Randomly initialized k lines j. (For j in 1…k)
2. For all points p, find the euclidean distance(vertical distance) from p the line j. Assign p to cluster j where p is closest among all k lines.
3. Computer local PCA inside each cluster, make the principle component the vector representing the line.
4. Re-assign the all points p base on the distance(p,p’) where p’ is the position of p after projection onto principle component.
5. See if the principle component converge(stable). If not, go back to step 2.

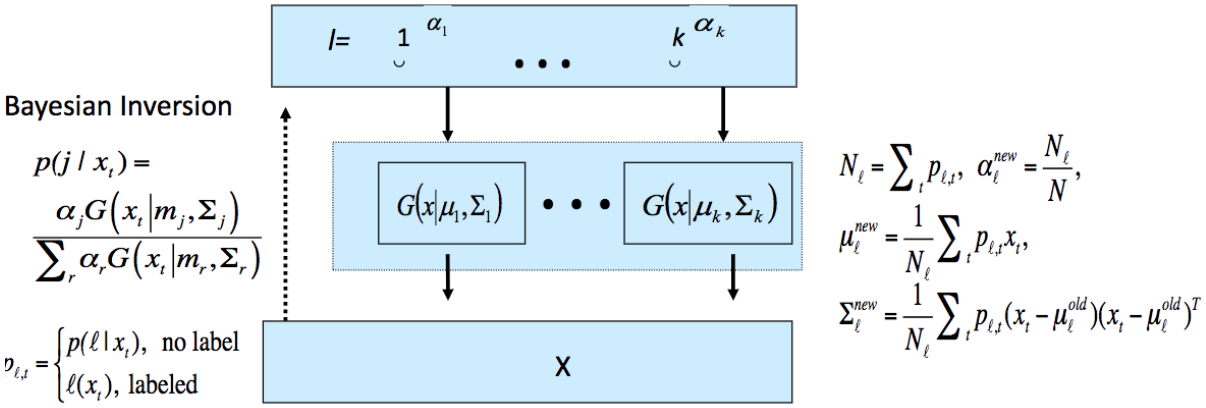
K-Gaussian is also an algorithm that uses K Gaussian distributions to represent K clusters. But I have already discussed that in the midterm questions, so I will skip it from here.

So far, all clustering algorithm we have discussed are all hard classification, winner take all, in the procedure of clustering, we will only assign only point to 1 cluster, nothing about probability involved here. I have learned Gaussian mixture model during the lecture. This is where the probability plays an important role during clustering. Unlike the hard classification, Gaussian mixture model uses “soft classification” (not winner take all). It is like each point has as the probability

that an observation comes from cluster(Gaussian) k (Gaussian with mean μ_k and covariance Σ_k). Each cluster is represented by an Gaussian distribution. We will then assign the points to K Gaussian clusters base on its.

$$q(x \mid \theta) = \sum_{r=1}^{k} \alpha_r G\left(x \middle| \mu_r, \Sigma_r\right)$$

We can estimate the parameters using MLE and EM, where P(r) = α_r is the "Latent variable".



The posterior P(j|x_t) is just given a point x_t, the probability that x_t belongs to Gaussian cluster j. It is equal to the mixing proportion α_k multiple by the Gaussian density function divided by the sum of for all α_k multiple by its Gaussian density. In total there should be K mixing proportion since there are K Gaussian cluster. The EM algorithm can be applied here to calculate the parameters.

1. Set the number of cluster K, for each cluster set there α_k, μ_k and covariance Σ_k
2. E step: Use the current α_k, μ_k and covariance Σ_k compute the current P(j|x_t). $p(j \mid x_t) = \frac{\alpha_j G\left(x_t \middle| m_j, \Sigma_j\right)}{\sum_r \alpha_r G\left(x_t \middle| m_r, \Sigma_r\right)}$
3. M step: Use the posterior P(j|x_t), compute the new α_k, μ_k and covariance Σ_k. $N_\ell = \sum_t p_{\ell,t},\ \alpha_\ell^{new} = \frac{N_\ell}{N},$ $\mu_\ell^{new} = \frac{1}{N_\ell}\sum_t p_{\ell,t} x_t,$ $\Sigma_\ell^{new} = \frac{1}{N_\ell}\sum_t p_{\ell,t}(x_t - \mu_\ell^{old})(x_t - \mu_\ell^{old})^T$
4. Check if the parameters or the maximum likelihood function are converge (stable). If not go back to step2.

We can say this algorithm is kind similar to K-means algorithm. They are all using the iterate technique. But they are also differed from classification (Soft and hard).

In conclusion, it seems like everything is connected. Starting from a single point to multiple points, a point representing multiple points, a line or a plane representing a cluster. Many points, lines, Gaussian even planes representing many points. Decision boundaries between different clusters, Gaussian distributions. Even different Gaussian distributions can be mixed together. Sometimes we want to fit the data perfectly to minimizing the error, sometimes we want to not completely minimize the error because it might cause overfitting. Iterate algorithm, EM algorithm for clustering works amazingly well. Overall, I have learned a lot of things.

Reference:
1.All PTT provided by Prof. Xu lei
2.https://www.quora.com/What-is-machine-learning-4
3.Assignments solution provided by TA
4.Wiki