**Machine Learning**

**Session 8**

1. Probability & Density Estimation;
   1. Three common probability distributions
      1. Binary variables: Bernoulli
      2. x is 1,0 (Heads or Tails). P(x;u) = u^{x}(1 – u) ^{(1 - x)}
      3. u (probability of Heads) from 0 to 1.
   2. Categorical variables: Multinomial
      1. x is 1-of-K encoding. P(x;u) = product sum of u\_i^{x\_i}
      2. ui (probability of outcome i) from 0 to 1. ui’s sum to 1.
   3. Continuous variables: Gaussian
      1. x is real vector. u is a real vector. S is a matrix.  
           
         p(x;u,S) = (1/Z) \* exp(-0.5\*(x-u)^{T}\*S^-1\*(x - u))
   4. Probability & Density Estimation
      1. Generative Perspective:
         1. Distributions tell us what data to expect according to specified parameters
         2. E.g., Biased coin (u=3/4). Expect H,H,H,T
         3. E.g., Mean & var of fish length Expect
      2. Density Estimation:
         1. Ask what probability distribution was responsible for specified data?
      3. There are simple exact solutions for the best estimates of binary, categorical, and Gaussian distributions given data
      4. Density Estimation: Ask what probability distribution was responsible for specified data?
         1. There are simple exact solutions for the best estimates of binary, categorical, and Gaussian distributions given data
   5. P(x;u) = u^{x}(1 – u) ^{(1 - x)} 🡺 u = (1/N) \* sum of (x\_i)
   6. P(x;u) = product sum of u\_i^{x\_i} 🡺 u\_k = (sum of(x\_ik)/double sum of(x\_ik)) = N\_k / N
   7. p(x;u,S) = (1/Z) \* exp(-0.5\*(x-u)^{T}\*S^-1\*(x - u)) 🡺 u = (1/N)\*(sum of(x\_i)), S = (1/N)\*(sum of((x-u)\*(x-u)^T)
2. Fitting a single Gaussian:
   1. Formally, we have a family of probability models for x tilde 𝑝 𝑥; theta , where theta denotes an unknown parameter vector.
   2. Let X = {𝑥1, 𝑥2, … , 𝑥\_n} we wish to fit a density 𝑝 𝑥; theta = 𝑁{𝑥; mu, sigma} , 𝑖. 𝑒. , theta = {mu, sigma}
3. Maximum Likelihood estimation:
   1. Assuming samples are independent, the data likelihood is:  
        
      p(x1, x2, …, xn; theta) = product sum of p(x\_n; theta) = L(theta)
   2. We minimise the negative loglikelihoood:  
        
      E = - ln sum of(L(theta)) = - sum of (ln p (x\_n given theta))
4. Case study: ML fitting of a single Gaussians:
   1. In one dimension: Let 𝑋 = 𝑥1, 𝑥2, … , 𝑥𝑁 we wish to fit a density 𝑝 𝑥; mu, sigma = 𝑁𝑥(mu, sigma)  
        
      𝐸 = − ln 𝐿(𝜃) = sum of( [n (𝑥𝑛 − mu)^2/2\*sigma^2] + [ln sigma] + 1/2 ln 2 pi.   
        
      Estimate mu ∶ Set derivative to zero   
       partial derivatives:  
        
      (del E / del mu) = - sum of((x\_n – hat(mu)) / sigma^2) = 0  
        
      hat(u) = (1/N) \* sum of (x\_n))

To solve for ML estimate of 𝜎 we use the fact that we already know hat(mu):  
  
(del E / del sigma) = sum of ((1/sigma) – ((x\_n – hat(u))^2 / sigma^3))  
  
Equating to zero gives:  
  
sigma^2 = (1/N) \* sum of((x\_n – hat(u))^2) sample variance  
  
Recall that the sample variance is a biased estimate for variance. However, such ML estimates are consistent and asymptotically optimal.

1. ML fitting a single Gaussians:
   1. In higher dimensions:  
        
      p(x given mu, sigma) = (1 / (2\*pi^{d/2} \* sigma^{0.5})) \*exp^{-0.5\*(x-u)^T\*sigma^-1\*(x-mu)}  
        
       we get similar results of:  
        
      ML estimate for mean, hat(mu):  
      hat(mu) = (1/N) \* sum of (x\_n))  
        
      and ML estimate for covariance matrix, hat(sigma):  
      hat(sigma) = (1/N) \* sum of((x\_n – hat(u)\* ((x\_n – hat(u))^T))
   2. Overfitting in ML density estimation
      1. Problem: In the case that the data dimensionality is higher than the data samples the covariance matrix is singular.
      2. Non-invertible, zero determinant.  
           
         hat(sigma) = (1/N) \* sum of((x\_n – hat(u)\* ((x\_n – hat(u))^T))
      3. p(x given mu, sigma) = (1 / (2\*pi^{d/2} \* sigma^{0.5})) \*exp^{-0.5\*(x-u)^T\*sigma^-1\*(x-mu)}  
           
         Put constraints in the form of the covariance matrix. E.g.  
           
         sigma^2 = (1/N) \* sum of((x\_n – hat(u)\* ((x\_n – hat(u))^T))  
           
         sigma\_j^2 = (1/N) \* sum of((x\_n (j)–hat(mu)(j))^2)
      4. Overfitting in ML density estimation:
         1. Regularise the covariance matrix by adding a diagonal matrix:  
              
            sigma\_0 = sigma\_0^2 \* *I*  
            hat(sigma) = (1/N) \* sum of((x\_n – hat(u)\* ((x\_n – hat(u))^T)) + sigma\_0^2 \* *I*  
              
            Leads to “thicker” Gaussian. Equivalent to adding zero mean Gaussian noise to the original dataset.
2. Gaussian Mixture Models:
   1. Tough data:
      1. K-means has problems
      2. Single Gaussian doesn’t fit it well
   2. GMM solution:
      1. Explain as: Weighted sum of K Gaussian densities
   3. GMM density estimation:
      1. Explain as: Weighted sum of K Gaussian densities
      2. Example problem and solution
3. Gaussian Mixture Models: Solution
   1. Optimization Criterion: Maximum Likelihood:  
        
      L(X;theta) = double sum of (pi\_k\*N)(x\_n;mu\_k,sigma\_k)
   2. Solution?
      1. If we knew which points belong to which clusters, we know how to fit Gaussians (Density Estimation: Gaussian)
      2. If we knew which points belong to which cluster, we know the relative size of each (Density Estimation: Multinomial)
      3. If we knew the Gaussians, we could find out which points belonged to which clusters (Bayes Theorem)
      4. EM iterative Algorithm Solution (at iteration i)
      5. E: Given parameters 𝜃^{𝑖−1}, infer how likely each point to each cluster. (calculate auxiliary variables 𝑞\_𝑛, 𝑘)
      6. M: Given soft assignments, update Gaussians and cluster prior. (calculate 𝜃^{𝑖})
4. EM for Mixture of Gaussians (MoG):
   1. Find parameters 𝜃 = { pi\_𝑘, mu(k), sigma(k) }\_{k=1 ... k}
   2. Data structures: N is
      1. #dinemsions, D is #samples, K is #Gaussians
      2. X (data matrix) NxD reals
      3. u = [mu\_k], (Gaussian centres) KxD reals
      4. SS = [sigma\_k], (Gaussian covariances KxDxD reals
      5. P = [pi\_k], (mixture coefficients) K reals
      6. q (Soft assignments matrix) NxK reals
   3. Algorithm:
      1. Initialize theta
      2. Iterate
         1. “E-step”: Softly assign points to Gaussians (fix u, optimize for q)
         2. “M-step”: Find new means/covariances and mixture weights of each Gaussian (fix q, optimize for P,u,SS)
5. EM Iterative update:
   1. Expectation − Maximization (beginning with initial values hat(theta)^{(0)}):
      1. E-step: calculate: q\_{n, k} = p(c given x\_n and hat(theta)^(i)) for all given *k* and *n*.
      2. M-step: maximise the weighted log of likelihood:
         1. hat(theta)^{(0)^{i+1} = maximise the argument sum of (q\_{n, k} ln p(x\_n given c\_k and theta\_k)) for all k
   2. A converged solution is a stationary point (e.g. local maximum) of the likelihood.
   3. Algorithm:
      1. “E-step”: Softly assign points to Gaussians (fix u, optimize for q)
      2. For 1< i < N for 1<k
         1. for 1<k<K
            1. q[n,k] = gaussian(X[n,:], u[k,:], SS[k,:,:])P[k];
         2. end
         3. q[n,:] = q[n,:] / sum(q[n,:]); % Normalise q along rows
      3. end
6. EM algorithm for MoGs:
   1. EM algorithm to estimate MoG parameters.
   2. Let the ith estimate for pi\_𝑘^{(𝑖)} = 𝑝(𝑐𝑘):
      1. E−step : Here we calculate q\_{𝑛, 𝑘} = 𝑝(𝑐\_𝑘 given 𝑥\_𝑛, theta^(𝑖)) by using
         1. p(𝑐\_𝑘 given 𝑥\_𝑛, theta^(𝑖)) = ((𝑐\_𝑘 given 𝑥\_𝑛, theta^(𝑖))\*p(c\_k)) / (x\_n given theta^(𝑖))
         2. = p(x\_n given c\_k, theta^(𝑖)) \* p(c\_k) / sum of (p(𝑐\_𝑘 given 𝑥\_𝑛, theta^(𝑖))) \* p(c\_j)
         3. = (𝑐\_𝑘 given 𝑥\_𝑛, theta^(𝑖)) \* pi^i\_k / sum of (𝑐\_𝑘 given 𝑥\_𝑛, theta^(𝑖)) \* pi^i\_k
         4. Where, (𝑐\_𝑘 given 𝑥\_𝑛, theta^(𝑖)) = (1/sigma\_k^{d/2}) \* exp(-0.5\*(x\_n – mu\_k)^T \* sigma^-1\_k \* (x\_n – mu\_k))
   3. EM iterative update (E-step):
      1. Expectation-Maximization: beginning with initial values hat(theta)^{(0)}
         1. E-step: calculate: q\_{n, k} = p(c\_k given x\_n, hat(theta)^{(0)} for all *k* and *n*
         2. M-step: maximise the weighted log likelihood:
         3. Hat(theta)^{I + 1} = maximise argument of sum of (q\_{n, k} \* ln p (x\_n given c\_k, theta\_k)) for all *k*.
      2. Algorithm: – M-step”: Find new means/covariances and mixture weights of each Gaussian (fix q, optimize for P,u,SS)
      3. for 1<k<K
         1. u[k,:] = sum(q[:,K]\*X) / sum(q[:,K]);
         2. SS[k,:,:] = sum(q[:,K]\*X\*X^T) / sum(q[:,K]);
         3. P[k]= sum(q[:,k]) / N
      4. End
   4. M−step :
      1. Mu\_k^{i+1} = (sum of (q\_{n, k}) \* x\_n) / (sum of (q\_{n, k}) – weighted mean
      2. Sigma^{i+1}\_k = (sum of(q\_{n, k} \* (x\_n – mu\_k^{i+1}) \* (x\_n – mu\_k^{i+1})^T) / (sum of (q\_{n, k})) – weighted covariance
      3. And we can estimate the priors 𝑝(𝑐𝑘) for each class 𝑐𝑘 using the following update:
         1. Pi\_k^{(i)} = (1/N) \* sum of q\_{n, k}
         2. All the variables can be updated as weighted averages!
   5. EM iterative update (M-step):
      1. Expectation-Maximization: beginning with initial values hat(theta)^{(0)}
         1. E-step: calculate: q\_{n, k} = p(c\_k given x\_n, hat(theta)^{(0)} for all *k* and *n*
         2. M-step: maximise the weighted log likelihood:
         3. Hat(theta)^{I + 1} = maximise argument of sum of (q\_{n, k} \* ln p (x\_n given c\_k, theta\_k)) for all *k*.
      2. Mu\_k^{i+1} = (sum of (q\_{n, k}) \* x\_n) / (sum of (q\_{n, k}) – weighted mean
7. GMM: Limitations
   1. Converges to local optima only
      1. Can also do multiple restarts and pick the best
   2. Picking K is still an issue
   3. Cost O(NKD^2)
      1. Data requirements >> O(D) due to covariance matrix
         1. (Estimating the shape information of each)
8. GMM versus K-Means:
   1. K-means: Algorithm
      1. For i=1:N
         1. ci=Index i of the cluster centroid closest to xi
      2. For k=1:K
         1. uk = average of points assigned to k
   2. Assumptions
      1. All clusters same size
      2. All clusters spherical
      3. Clusters are sharply peaked
      4. Hard assign points to clusters
      5. Optimize:
         1. Sum Squared Distance of points to clusters
   3. GMM: Algorithm
      1. For i = 1 : N
         1. Probability p(k given xi ) of belonging to each cluster k
      2. For k=1:K
         1. Fit the Gaussian N(uk ,Sk) given probabilities p(k given xi)
         2. Fit the cluster prior p(k) given probabilities p(k given x).
   4. Assumptions
      1. Cluster k of size pi\_k
      2. Clusters of shape S
      3. Clusters have spread S
      4. Soft assign points to clusters
      5. Optimize:
         1. Log-likelihood of data
9. GMM: A clustering perspective:
   1. K-means works if:
      1. Clusters are spherical
      2. Clusters are well separated
      3. Clusters are of similar spread
      4. Clusters have same number of points
   2. Motivate:
      1. Mixture of Gaussians algorithm
10. Choosing Number of Clusters For K-means and GMM:
    1. Elbow Method
       1. Plot E\_{KM/GMM} as a function of k, and choose the elbow point.  
          Fig:  
          A xy graph with 4 cluster of points and J(K) as the y-axis and K and the x-axis. Elbow point is where there is a highest decrease in J(K).
    2. Present results to end users see what they prefer
       1. Broader or more specialized groups
    3. Cross-validation
       1. For K = 1…Large
          1. Learn GMM on a train set.
          2. Evaluate Quality(K) = quality on validation set.
       2. Pick K with the highest validation set quality
11. Choosing Number of Clusters For K-means and GMM
    1. BIC/AIC Criterion
       1. (ML people: An approximation to the integration required in the Bayesian model selection)
       2. Adds a penalty to the cost that penalises more complex models.
          1. P: Number of parameters in model. N: Number of data points.
       3. Evaluate modified cost EK BIC for many values of K.
       4. Pick the K with best cost:
          1. E^K\_{BIC} = E^K – (p/2) log *N*
12. Applications:
    1. Anomaly Detection: Motivation
       1. Sometimes you are interested in finding unusual items
          1. “Anomalies”, “Outliers”
       2. …As a pre-processing step
          1. E.g., algorithms that use Sum-Squared objectives are not robust to outliers. Use outlier detection first to find and discard such rows
       3. …As an end goal.
       4. Example: Manufacturing Quality Control: Aircraft Engines   
          Fig:  
          xy graph ==> x-axis = heat, y-axis = vibration. Choose lowest point in the cluster and if a point is further away from there, it is considered an anomaly.
       5. Algorithm:
          1. Read in normal training data, {x}
          2. Compute the Gaussian (u,S) that best explains the data {x}
          3. Given a new example x and estimated u,S, compute p(x)
             1. If p(x) < T, then Anomaly

else Ok

1. Gaussian Classifiers:
   1. Assume some training data X = {(x\_i, y\_i)}.
   2. Produce a model y = f(x\*) that can classify new data  
        
      p(y=j given x) = (p(x given y=j) \* p(j)) / p(x)
   3. Assign the new data point to class j if
      1. p(y=j given x) >= p(y=i given x), where i != j.
      2. p(x given y = j) \* p(j) >= p(x given y=i) \* p(i) where i != j.
   4. How to get p(x|y=i)? Maximum Likelihood Estimation:
      1. Fit a multivariate normal to {xi}\_i where y\_i=1
      2. Fit a multivariate normal to {xi}\_i where y\_i=2
   5. Change the likelihood according to the data type:
   6. For continuous data: p(x|w) is commonly Gaussian / Mixture of Gaussians. x is a d-dimensional vector:  
        
      u = (1/N) \* sum of (x\_n)  
      S = (1/N) \* sum of((x\_n - u) \* (x\_n - u)^T)
   7. For discrete data:
      1. Binary case: p(x given w) is Bernoulli, where x\_n is in the interval {0, 1}
      2. K-class case: p(x given w) is multinomial, where x\_n is in the interval {0, 1}
   8. Common for: spam classifiers, sentiment recognition, etc.
2. Summary
   1. Anomaly / outlier detection: Fit a probability density and detect outliers where probability is below a threshold.
   2. Bayesian classifiers: Fit a probability density to each class and classify according to the posterior of the class.
   3. Choose the distribution based on the data!