Mixtures of Gaussians and EM
Unsupervised Learning

• Suppose the data (i.e., x’s) belongs to different classes, but we don’t have the labels (i.e., we don’t have the y’s)
• Won’t to characterize the different x’s somehow (e.g., “x\(^{(i)}\) belongs to cluster B,” there are 3 different clusters of data)
• Or to compute features that could be useful for classification (e.g., (1, 0, 0) if the x belongs to Cluster A, (0, 1, 0) if the x belongs to Cluster B, (0, 0, 1) if the x belongs to Cluster C)
  • If we can figure out how to compute those features using a large unlabelled dataset, we could then use them to perform supervised learning on a small labelled dataset
  • Like using the AlexNet features to classify faces
A Generative View

To generate a datapoint:

• Pick Cluster A with probability $P_A$, Cluster B with probability $P_B$, ...

• If we picked Cluster cl, sample random coordinates from $N(\mu_{cl}, \Sigma_{cl})$
A Generative View

• If the data is well-described as several “clouds” of points, we can generate a datapoint that looks like it was sampled from the training set by picking a cloud and then picking a coordinate from the cloud.

• “Clouds” can be conveniently described as multivariate Gaussians
Mixture of Gaussians

• \( P(x|\pi, \mu, \Sigma) = \sum_{cl} P(x|\mu, \Sigma, cl)P(cl|\pi) \) by the law of total probability
  • Weighted sum of the likelihoods for all the clusters, weighted by the probabilities of the clusters

• \( P(x|\pi, \mu, \Sigma) = \sum_{cl} P(x|\mu, \Sigma, cl)P(cl|\pi) = \sum_{cl} P(x|\mu_{cl}, \Sigma_{cl})\pi_{cl} \)
Learning a Mixture of Gaussians

• Let $z^{(i)}$ be the cluster to which point $i$ is assigned.

• If we knew all the $z^{(i)}$, we could learn the Gaussians one-by-one. But we don’t. Instead, we can try to estimate

$$w_{cl}^{(i)} = p(z^{(i)} = cl|x^{(i)}, \pi, \mu, \Sigma) = \frac{P(x^{(i)}|\mu_{cl}, \Sigma_{cl})\pi_{cl}}{P(x^{(i)}|\pi, \mu, \Sigma)} \propto P(x^{(i)}|\mu_{cl}, \Sigma_{cl})\pi_{cl}$$

• But we don’t know $\mu, \Sigma, \pi$ either! But if we estimate the z’s, it’s easy to estimate $\mu, \Sigma, \pi$. 
Learning a Mixture of Gaussians

• E-step:
• Want to estimate the cluster assignments $z^{(i)}$.

$$
\phi_{cl}^{(i)} = P(x^{(i)}|\mu_{cl}, \Sigma_{cl})\pi_{cl}
$$

$$
w_{cl}^{(i)} = p(z^{(i)} = cl|x^{(i)}, \pi, \mu, \Sigma) = \frac{P(x^{(i)}|\mu_{cl}, \Sigma_{cl})\pi_{cl}}{P(x^{(i)}|\pi, \mu, \Sigma)} \propto \phi_{cl}^{(i)}
$$

$$
w_{cl}^{(i)} = \frac{\phi_{cl}^{(i)}}{\sum_{cl} \phi_{cl}^{(i)}}
$$
Learning a Mixture of Gaussians

- M-step: Assume probabilistic cluster assignments were done

\[
\pi_{cl} = \frac{1}{m} \sum_i w_{cl}^{(i)} \\
\mu_{cl} = \frac{\sum_i w_{cl}^{(i)} x^{(i)}}{\sum_i w_{cl}^{(i)}} \\
\Sigma_{cl} = \frac{\sum_i w_{cl}^{(i)} (x^{(i)} - \mu_{cl})(x^{(i)} - \mu_{cl})^T}{\sum_i w_{cl}^{(i)}}
\]
Learning a Mixture of Gaussians

• Start with an initial guess of $\pi, \mu, \Sigma$

• Repeat:
  • Perform E-step to estimate the (probabilistic) cluster assignments of each point
    \[ w_{cl}^{(i)} = p(z^{(i)} = cl | x^{(i)}, \pi, \mu, \Sigma) \]
  • Assume cluster assignments, and re-estimate $\pi, \mu, \Sigma$ based on them
Learning a Mixture of Gaussians

• Very easy to get stuck in local optima

• Example:
  • A Gaussian whose variance is very small, and whose mean is very close to one point x
  • The E-step will only assign x to that Gaussian since the variance of the Gaussian is very small so the likelihood for any other point is small
  • The M-step will make the mean exactly equal to x, and make the variance even smaller

• Solution: start with Gaussians with large variances
Learning a Mixture of Gaussians

• How do we select the number of clusters?
• Try different numbers of clusters, select the number of clusters that maximizes the probability density of the validation set
  • Imagine fitting a very small-variance Gaussian to every point in the training set: this would give a very small probability density to the validation set
K-means

• K-means is an algorithm for finding centres of clusters
• Simpler than Mixture of Gaussians, but the same idea
K-means

• Assignment step: assign each datapoint to the closest cluster
• Refitting step: Move each cluster center to the average of the points assigned to the cluster
Why K-means converges

• Whenever an assignment is changed, the sum squared distances of datapoints from their assigned cluster centers is reduced.

• Whenever a cluster center is moved the sum squared distances of the datapoints from their currently assigned cluster centers is reduced.

• If the assignments do not change in the assignment step, we have converged.
K-means: local optima

• You could get back local optima with k-means
• Try multiple starting points
  • How to evaluate how good the result is?
Speeding up Learning: MoG

• Run k-means first, initialize the means of the Gaussians to be the means obtained using k-means