67750: Advanced Practical Machine Learning 2017/8

Quiz - Solution

TA: Daniel Gissin

1 Manifold Learning (25 Points)

1.1 LLE

In the last step of LLE, after estimating each data point using an affine transformation of its k nearest neighbors described by the weight matrix W, we decompose $M = (I - W)^T (I - W)$ into its eigenvectors.

1. Given the calculated W from step 2 of the algorithm, what was our original minimization objective $\Phi(Y)$, which we showed to be equivalent to minimizing $y^T M y$?
2. Show that $\mathbf{1}$ is an eigenvector of M. What is it's eigenvalue?
3. Which eigenvector of M should we take as the coordinates for the 1D case? Why?

Solution

1. In LLE we start by calculating the best affine approximation for each point, given its neighbors. This is expressed as a weighted sum of the neighbors. After calculating W, the matrix representing those weighted sums, we want the weighted sums in the intrinsic coordinates to also represent the actual coordinates well:

$$\Phi(Y) = \sum_{i=1}^{n} ||y_i - \sum_{i \neq j} W_{i,j} y_j||^2$$

2. We will use the fact that the rows of W sum to one:

$$M\mathbf{1} = (I - W)^T (I - W)\mathbf{1} = (I - W)^T (I\mathbf{1} - W\mathbf{1}) = (I - W)^T (\mathbf{1} - \mathbf{1}) = \mathbf{0}$$

3. We are minimizing $y^T M y$, so taking the bottom eigenvector will result in taking $\mathbf{1}$. This is not what we want and is basically the same as taking $\mathbf{0}$, since we are assigning all of the data points to the same single point. While this allows us to easilly represent each point as a weighted sum of its neighbors, it's not very productive... This is why we take the next lowest eigenvector.

1.2 Constrained Optimization

Let $A \in R^{m \times n}$. We wish to find the vector v on the unit sphere which, after multiplication by A, has the minimal squared L_2 norm.

1. Define the Lagrangian for the above constrained optimization problem.
2. Show that the solution v^* can be obtained by an eigendecomposition of a matrix, and show which matrix we need to decompose.
3. Which eigenvector do we need to take to minimize the squared norm? What will the minimal norm be?

Solution

Our constraint in this question is that the original vector is on the unit sphere (its norm equals 1). What we want to maximize is the squared norm of the vector Av.

1. A squared norm of a vector can be expressed as v^Tv, so we can simply write the Lagrangian as:
$$L(v,\lambda) = (Av)^T(Av) - \lambda(v^Tv-1) = v^TA^TAv - \lambda(v^Tv-1)$$
2. We solve the optimization problem by differentiating the Lagrangian w.r.t. all of the variables. We'll start by differentiating w.r.t. v (keep in mind, A^TA is symmetric):
$$\frac{\partial L}{\partial v} = 2v^TA^TA - 2\lambda v^T = 0 \to v^T(A^TA) = \lambda v^T \to A^TAv = \lambda v$$
Differentiating w.r.t. λ gives us our constraint, which removes the solution $v = \mathbf{0}$ from the possible solutions, and so v^* must be an eigenvector of A^TA.
3. Since we want to minimize the norm, we are looking for the eigenvector which minimizes v^TA^TAv. This is clearly obtained by the (or one of the) eigenvector(s) corresponding to the minimal eigenvalue:
$$u_i^TA^TAu_i = \lambda_i u_i^Tu_i = \lambda_i \to \arg\min_{u_i}(u_i^TA^TAu_i) = u_{min}$$
And we can see that the maximal norm is $\sqrt{\lambda_{min}}$...
This may also ring a bell as the right singular vector of A corresponding to the minimal singular value.

2 Unsupervised Image Denoising (25 Points)

2.1 The Gauss-Markov Theorem

1. State the Gauss-Markov theorem.
2. Complete the following proof (given in class) and explain why the given steps of the proof are true:

Proof. Let $A : \mathit{support}(Y) \to \mathit{support}(X)$

$$\begin{aligned}
\mathbb{E}\|x - A(y)\|^2 &= \int_x \int_y p(x,y)\|x - A(y)\|^2 dy dx && (1)\\
&= \int_y p(y) \int_x p(x|y)\|x - A(y)\|^2 dx dy && (2)\\
\text{We require that } \forall y \ \frac{\partial}{\partial A(y)} &\left[\int_x p(x|y)\|x - A(y)\|^2 dx\right] = 0 && (3)\\
&\iff ?? && (4)
\end{aligned}$$

Solution

The Gauss Markov theorem states that Let $A : \mathit{support}(Y) \to \mathit{support}(X)$. Define $MSE(A) = \mathbb{E}[||A(y) - x||^2]$ and $A^* = \mathbb{E}[x|y]$, then for all possible A, $MSE(A^*) \leq MSE(A)$.

The first step of the proof is simply the definition of the expectation. The second step is just the law of total probability applied to $p(x,y)$, where $p(y)$ doesn't depend on x and so we can take it out of the integral over x. In the third step we notice that the original expectation is minimized if the integral over x is minimized for all y's - so if we differentiate w.r.t. $A(y)$ and set to zero, then we can find the function A which minimizes each element of the integral and so minimizes the entire expectation. The following is the completion of the proof:

$$2\int_x p(x|y)(x - A(y))dx = 0$$

$$\int_x p(x|y)x dx = \int_x p(x|y)A(y)dx$$

$$\mathbb{E}[x|y] = A(y)\int_x p(x|y)dx$$

$$A^* = \mathbb{E}[x|y]$$

The first step simply takes the derivative. The second step seperates the integral into two integrals. The third step shows that the left-hand side is just the definition of the expectation. Finally, $\int_x p(x|y)dx = 1$ since $p(x|y)$ is a probability distribution.

2.2 MLE Calculation

In class we saw that the Expectiation of the log likelihood of the Gaussian Mixture Model can be written as:

$$\mathbb{E}[LL(S,\theta)] = \sum_{i=1}^{n}\sum_{y=1}^{k} c_{i,y} log(\pi_y N(x_i; \mu_y, \Sigma_y))$$

In the exercise, we implemented EM for the Gaussian Scale Mixture (GSM) model, and calculated the maximization step for the scaling constants $\{r_y\}_{y=1}^{k}$. The GSM model assumes that image patches are sampled from a mixture of k Gaussians, with the distributions $N(0, r_y^2\Sigma)$.

Derive the update for r_y^2 using MLE.

Some linear algebra identities and the MVN distribution might help get you started (A is an $n \times n$ matrix, c is a scalar):

$$N(x; \mu, \Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

$$det(cA) = c^n det(A)$$

$$(cA)^{-1} = c^{-1}A^{-1}$$

Solution

We want to maximize the expected log likelihood w.r.t. one of the r's, say r_1. We can rewrite the expected log likelihood in the following way (the constant is w.r.t. the r's and d is the dimension of the image patch):

$$E[LL(S, \theta] = const + \sum_{i=1}^{n} \sum_{y=1}^{k} c_{i,y}(log(e^{-\frac{1}{2r_y^2} x_i^T \Sigma^{-1} x_i}) - log(\sqrt{|2\pi r_y^2 \Sigma|}))$$

$$= const + \sum_{i=1}^{n} \sum_{y=1}^{k} c_{i,y}(-\frac{1}{2r_y^2} x_i^T \Sigma^{-1} x_i - log(\sqrt{|2\pi r_y^2 \Sigma|}))$$

$$= const + \sum_{i=1}^{n} \sum_{y=1}^{k} c_{i,y}(-\frac{x_i^T \Sigma^{-1} x_i}{2r_y^2} - dlog(r_y))$$

Now, we differentiate w.r.t. r_1 to get our solution:

$$\frac{\partial E[LL(S, \theta]}{\partial r_1} = \sum_{i=1}^{n} c_{i,1}(\frac{x_i^T \Sigma^{-1} x_i}{r_1^3} - \frac{d}{r_1}) = 0$$

And solving for r_1 gives us our desired maximization update step:

$$r_1^2 = \frac{\sum_{i=1}^n c_{i,1} x_i^T \Sigma^{-1} x_i}{d \sum_{i=1}^n c_{i,1}}$$

3 Clustering (25 Points)

3.1 k-means

1. k-means++ is a method to initialize the centroids in the k-means algorithm. How is initialization performed in k-means++?

   Explain the logic behind k-means++. What situations is it trying to avoid?

2. Prove that k-means converges in a finite number of steps.

Solution

1. k-means++ weights the data points according to their distance from the closest centroid drawn so far, then uses those weights as probabilities of drawing the next centroid. The first centroid is chosen uniformly over all of the data points, and then the weights of each remaining point are decided using the following formula:

   $$w(x_i) = min_{\mu \in centroids}(||x_i - \mu||^2)$$

   Then, the next centroid is sampled from a multinomial distribution over all of the data points, with probabilities proportional to the weights. This procedure insures that points that are far away from centroids have a higher probability of being picked as a new centroid, while having outliers still not very likely of being picked (assuming there are enough non-outliers that are relatively far away). It avoids initialization where there are two centroids close together, which has a higher chance of converging to a suboptimal solution...

2. The proof can be seen in the solution of the theoretical part of ex3.

3.2 Spectral Clustering

In spectral clustering we view the data as a graph with n vertices, where the edges are weighted as some function of the distance between the data points.

1. Write the Ratio-Cut objective function for $k = 2$.

2. Why do we use the Ratio-Cut objective function instead of simply looking for a minimal cut in the graph?

Solution

1. For two clusters, A and B, the objective function is: $RatioCut(A, B) = Cut(A, B) \cdot (\frac{1}{|A|} + \frac{1}{|B|}) = \frac{\sum_{x \in A, y \in B} w(x,y)}{(\frac{1}{|A|} + \frac{1}{|B|})}$.

2. Simply looking for a minimal cut in the graph usually gives us a cut where a single point is in A and the rest of the points are in B. This is because summing over the neighborhood of one point may be smaller then summing over a cut where the two sets are of similar size. Ratio Cut penalizes these sorts of situations, making the minimum be two sets of similar sizes, like we want.

4 Structured Prediction (25 Points)

4.1 Markov Chains

You have successfully modeled the weather as a Markov chain and have learned the probabilities for a day being sunny, rainy or snowy, given the day before. These are the probabilities you've learned:

$$\begin{aligned}
Pr(sunny \to sunny) =& 0.7\\
Pr(sunny \to rainy) =& 0.3\\
Pr(sunny \to snowy) =& 0\\
Pr(rainy \to sunny) =& 0.3\\
Pr(rainy \to rainy) =& 0.5\\
Pr(rainy \to snowy) =& 0.2\\
Pr(snowy \to sunny) =& 0\\
Pr(snowy \to rainy) =& 0.7\\
Pr(snowy \to snowy) =& 0.3
\end{aligned}$$

Today was rainy.

1. What is the probability of it raining in two days?
2. What is the probability of it being rainy in the distant future?

Solution

1. We simply need to sum over the probabilities. We can get to rainy in two days by going through sunny, rainy or snowy (and our prior is that we started in rainy). Denote $sunny = x$, $rainy = y$ and $snowy = z$:

   $$Pr(y \to y)Pr(y \to y) + Pr(y \to x)Pr(x \to y) + Pr(y \to z)Pr(z \to y) =$$

   $$0.5^2 + 0.3^2 + 0.2 \cdot 0.7 = 0.48$$

2. This Markov chain is irreducible and aperiodic (you can reach any state from any state with a positive probability and all states have a period of 1). This means that there is a unique stationary distribution and that in the distant future we are guaranteed to converge to it. So the fact that today was rainy is irrelevant and we just need to find the stationary distribution.

   We will denote $Pr(sunny) = x$, $Pr(rainy) = y$ and $Pr(snowy) = z$. Putting the above probabilities into three equations gives us (along with our probability constraint):

   $$\begin{aligned}
   x =& 0.7x + 0.3y\\
   y =& 0.3x + 0.5y + 0.7z\\
   z =& 0.2y + 0.3z\\
   1 =& x + y + z
   \end{aligned}$$

   And solving for y gives us $Pr(rainy) = \frac{7}{16}$.

4.2 Hidden Markov Models

HMMs are defined by the transition distribution (τ) and the emission distribution (ϵ), assuming we added a "START" state at time $t = 0$.

1. Write the definition of $V_t(i)$, the value which we calculate in the Viterbi dynamic programming algorithm.
2. Write the recursive formula for calculating $V_t(i)$, given V_{t-1}.
3. Write the value of $V_1(i)$.

Solution

1. $V_t(i) = \max_{\{y \in S^t | y_t = i\}}(Pr(y|x_{1:t}))$.
2. $V_t(i) = \max_{j \in [|S|]}(V_{t-1}(j)\tau_{j,i}\epsilon_{i,x_t})$.
3. $V_1(i) = \tau_{y_0,i}\epsilon_{i,x_1}$.