Lecture 4

- The Expectation-Maximization Algorithm
- Super-pixels: Generative versus Affinities
- Affinities and Spectral Clustering
- Segmentation by Weighted Aggregation

The EM Algorithm (1)

- The EM algorithm is a way to estimate parameters θ of a model if some variables x can be observed, but others h are hidden/latent/missing (terms differ depending on the research community).
- A classic paper (Dempster, Laird, and Rubin 1977) showed that EM was a general way to formulate problems of this type (many existing algorithms were special cases of EM). Special cases, not known to Dempster et al.,, included Hidden Markov Models (HMMs) and the Boltzmann Machine (BM).
- Suppose we have data x which is generated by a probabilistic model $P(x|h,\theta)$ with a prior $p(h)$ for the hidden variables h. This gives a distribution $p(x,h|\theta)$ from which we can compute the marginal distribution $p(x|\theta) = \sum_h p(x,h|\theta)$.
- The goal is to estimate $\hat{\theta} = \arg\max P(x|\theta)$ (i.e. the maximum likelihood estimate of θ). This can be formulated in terms of minimizing $-\log p(x|\theta)$.

The EM Algorithm (2)

- To obtain EM we introduce a new variable $q(h)$ which is a distribution over the hidden variables. We define a *free eergy function* $F(\theta, q) = -\log p(x|\theta) + \sum hq(h) \log \frac{q(h)}{p(h|x,\theta)}$. The second term is the Kullback-Leibler divergence, which has the property that it is non-negative and is zero only if $q(h) = p(h|x, \theta)$. This implies that minimizing $F(\theta, q)$ with respect to θ and q is equivalent to minimizing $-\log p(x|\theta)$ with respect to θ (by setting $q(h) = p(h|x, \theta)$).
- The EM algorithm consists of minimizing $F(\theta, q)$ with respect to θ and $q(.)$ alternatively. (These correspond to the two steps of the k-means algorithm.) The algorithm is specified most simply by re-expressing $F(\theta, q) = \sum_h q(h) \log q(h) - \sum q(h) \log p(h, x|\theta)$ (which exploits $p(h, x|\theta) = p(h|x, \theta)p(x|\theta$ and $\sum hq(h) \log p(x|theta) = \log p(x|theta)$.
- The algorithm starts with an initialization. Then follows by repeating the two staps: (1) Fix θ and estimate $\hat{q}(.)$ by $p(h|x, \theta)$, which requires computing $P(h, x|\theta)/p(x|\theta)$. (2) Fix $q(.)$ and estimate $\hat{\theta} = \arg\min | - \sum_h q(h) \log p(h, x|\theta)$.
- Step 1 minimizes $F(\theta, q)$ with respect to q and Step 2 minimizes $F(\theta, q)$ with respect to θ. Hence each step is guaranteed to reduce the free energy and hence the algorithm converges to a minimum of the free energy. The free energy is a non-convex function, so typically we need a good initialization to ensure that the EM algorithm obtains a good result (i.e. results in a solution which is close to the global minimum),

The EM Algorithm (3)

- Dempster et al. did not formulate EM in terms of minimizing a free energy. This formulation was due to Hathaway (Statistics community) and Hinton and Neal (neural network community). The free energy formulation is better because it enables many variants and approximations (e.g., do iterations of steepest descent with respect to $q(.)$ or θ, restrict $q(.)$ to take a specific form – like a factorizable distribution – to make the E and M steps possible. Note: for many problems it is hard to compute the E and M steps (it is easy for mixtures of Gaussians).
- Here is another variant. Suppose we have $p(\theta|D) = \sum_h p(\theta, h|D)$, where D is the data. Introduce a distribution $q(h)$ and define a free energy $F(\theta, q) = -\log p(\theta|D) + \sum_h q(h) \log \frac{q(h)}{p(h|\theta,D)}$
- Then, similar to previous slide, we minimize with respect to $q(h)$ and θ alternatively. This gives two steps: (1) Fix q^t, set $\theta^{t+1} = \arg\min_\theta\{-\sum_h q(h) \log p(h, \theta|D)\}$. (2). Fix θ^t, set $q^{t+1}(h) = p(h|\theta^t, D)$. *Initialize and iterate both steps until convergence.*

The EM Algorithm (4)

- Another variant (similar to k-means) is where we have data $\{x_n : n = 1, ..., N\}$ generated by a distribution $p(x_n|h_n, \theta)$, where the hidden variables h_n are different for each n (this is the typically case). We introduce distributions $q_n(h_n)$. The goal is to minimize the negative log-likelihood of all the data $-\sum_{n=1}^{N} \log p(x_n|\theta)$, where $p(x_n|\theta) = \sum_{h_n}(x_n, h_n|\theta)$.
- We define a free energy $F(\theta, \{q_n()\}) = -\sum_{n=1}^{N} \log p(x_n|\theta) + \sum_{n=1}^{N} q_n(h_n) \log \frac{q_n(h_n)}{p(h_n|x_n,\theta)}$. The EM algorithm consists of minimizing $F(.,)$ with respect to $\{q_n()\}$ and θ alternatively, and yield steps similar to those on the previous slides.

Super-pixels: Generative versus Affinity

- The goal of super-pixels is to decompose an image into non-overlapping subregions, where the intensity/texture properties are roughly homogeneous within each subregion. A stronger requirement is that neighbouring sub-regions have different intensity/texture properties. If D is the image domain (e.g., a grid) then we want to decompose it into a set of sub-regions D_a, so that $D = \bigcup_a D_a$ and these sub-regions doe not overlap, $D_a \bigcap D_b = \phi$ for all $a \neq b$, where ϕ is the empty set.
- Note that this is a variant of *clustering* (as addressed by the EM algorithm), which includes spatial relations. We want to cluster different image points together so that: (i) points with similar intensity/texture properties tend to get clustered together, and (ii) points which are spatially near each other tend to get clustered together.
- Similarly as for clustering, there are two types of methods to formulate this problem. The first type is *generative*, where we assume that there is a probability model for generating the data (e.g., a mixture of Gaussians). The mixture variables specify the clusters (e.g., datapoints which are generated by the same Gaussian get put in the same cluster). The second type is to specify an *affinity measure* between datapoints, e.g., $w(x_i, x_j)$ and to group datapoints which have high affinity together.
- There are many ways of grouping by affinity. One of the best known techniques is *spectral clustering* (notes for this should be made available). We will present a simple algorithm for grouping by affinity on the next slide.

Grouping by Affinity

- We present a simple algorithm for grouping by affinity. We have a set of datapoints $\{x_n\}$ and an affinity measure $w(x_n, x_m)$, so that $w(x_n, x_m)$ is large if x_n and x_m are similar, but small if they are not.
- We define a set of subgroups $C_i(.)$ of datapoints. We construct these subgroups by an algorithm, see below, which attempts to put similar datapoints into the same cluster. This involves a threshold τ (to specify the tradeoff between having a large number of subgroups within which all the datapoints are very similar, versus a smaller number of subgroups within which the datapoints are less similar). This algorithm is intuitive but not guaranteed to converge to an optimal set of subgroups.
- Order the datapoints at random and set $C_1() = \{x_1\}$. For datapoint x_2, if $w(x_2, x_1) > \tau \sum_n w(x_2, x_n)$ then set $C_1() = \{x_1, x_2|]$, otherwise set $C_1 = \{x_1\}, C_2 = \{x_2\}$. the algorithm proceeds in the natural way. At time step t we have $t_a \leq t - 1$ subrgoups $\{C_i(.) : i = 1, ..., t_a\}$ each of which is represented by a single datapoint x_i^* (the first datapoint assigned to that subgroup). For datapoint x_t we compute its similarity $w(x_n, x_i^*)$ to each of the sub-groups and compare to $\tau \sum_n w(x_t, x_n)$. If $\max_i w(x_n, x_i^*) < \tau \sum_n w(x_t, x_n)$ then create a new class $C_{t_a+1} = \{x_t\}$. If not, assign x_n to the subgroup C_{n^*}, where $n^* = \arg\max_i w(x_n, x_i^*)$.

Generative versus Affinity

- What are the advantages of generative or affinity methods for clustering and for super-pixels? This is a hard question to answer and is problem specific. For some problems there are natural affinity measures, while for others there are natural generative models. Theoretically, if you know the generative model then it is possible to derive an affinity (Griffiths and Tenenbaum) but this of often nor very practical (and, for vision, it is very hard to specify generative models).
- Even if generative models are known it can be useful to use affinity methods to initialize the EM algorithm (the effectiveness of EM depends strongly on whether you can initialize it well). For example, consider generative models of images of cars. Here the hidden variable is the viewpoint. Cars seen from similar viewpoints have similar images, so it is possible (though not easy) to use affinity methods to group cars in different viewpoints. This helps initialize an EM algorithm, because car images clustered by viewpoint (using affinity methods) should have the same hidden variables.

Super-pixels example – and why should you care?

- The second handout for this lecture give an example of super-pixels which uses a combination of both generative models and affinities. The generative models are extensions of mixtures of Gaussians, where the variables include the spatial positions in the image as well as their image intensities (the SLIC algorithm). This divides the image into subregions which tend to be homogeneous, but all subregaions have fairly similar sizes so neighbouring subregions may also have similar image/texture properties. A second stage uses affinity methods to group together subregions whose image/texture properties are similar, which results in super-pixels which have homogeneous image/texture properties but which can differ greatly in size.
- Why should we care about super-pixels? They were an important research topic before the rise of Deep Nets. They could also be used as ways to specify the location and sizes of objects in images, which could be used to initialize object classification by Deep Nets. They suffer, at present, because there are no datasets for benchmarking them and it is not possible to train them end-to-end (although it is possible to learn affinities). But this is also an advantage because they can be applied to the many real world problems where bencharked ground truth is not available for training algorithms. There are neuroscientific theories that the goal of every vision is to break images up into psuedo-objects (von der Heyte, Niebur, Etienne-Cummings) which is in the spirit of super-pixels (though the methods they use are rather different).

Additional Notes: Missing

- Spectral Clustering – handout will be prepared.
- Segmentation by Weighted Aggregation – handour will be prepared.