Vision as Bayesian Inference

Alan Yuille

January 30, 2024

Lecture 3

- Matched Filters and Dictionaries.
- The K-means algorithm.
- Soft-coding: mixture of Gaussians with Expectation-Maximization (EM).
- Mixture of Von Mises Fisher.
- Mini-epitomes.

Matched Filters (1)

- Suppose we have a function $\vec{B}$ and an input image patch $\vec{I_p}$.
- We want to find the best fit of the filter to the image by *scaling* the function by a and adding a *background* term b.
- This gives a transformation $\vec{B} \mapsto a\vec{B} + b\vec{e}$, where $\vec{e} = (1/\sqrt{N})(1, ..., 1)$. $\vec{B}$ and $\vec{e}$ are normalized so that $\vec{B} \cdot \vec{B} = \vec{e} \cdot \vec{e} = 1$. If $\vec{B}$ is a derivative filter then, by definition, $\vec{B} \cdot \vec{e} = 0$.
- The goal is to find the best scaling a and background b to minimize the match:

$$E(a, b) = |\vec{I_p} - a\vec{B} - b\vec{e}|^2.$$

Matched Filters (2)

- The solution $\hat{a}, \hat{b}$ is found by setting the derivatives of E with respect to a and b, enforcing that $\vec{B}$ and $\vec{e}$ are normalized, yielding:

$$\hat{a} = \vec{B} \cdot I_p, \quad \hat{b} = \vec{e} \cdot \vec{I_p}.$$

.

- The estimated scaling $\hat{a}$ is the dot product of the function with the image. The estimated background $\hat{b}$ is the mean value of the image. The minimized energy $E(\hat{a}, \hat{b})$ is a measure of how well the filter "matches" the input image.

Dictionaries of Functions

- Matched filters leads naturally to the idea of having a "dictionary" of functions $\{\vec{B}^{\mu} : \mu \in \Lambda\}$, where different functions $\vec{B}^{\mu}$ are tuned to different types of image patches.
- An image patch is encoded by the function that best matches it, after estimating the scaling and the background. The bigger the dot product $\vec{B} \cdot \vec{I}$ the better the function $\vec{B}$ matches the image patch $\vec{I}_p$.
- This relates to popular methods of quantizing stimuli, e.g., the feature vectors of a neural network, into dictionaries of tokens. See later in the course.
- Mote that matched filters are an extreme case of sparsity. An image patch is represented by a single function while for l^1 sparsity we allows a linear combination of functions.

K-means (1)

- A popular strategy for learning a dictionary of matched filters (functions), is by using the K-means algorithm. This is a classic clustering algorithm. As we will show, it related to mixtures of Gaussians and the EM algorithm.
- For simplicity, we will set $a = 1$ and $b = 0$ (contrast and background can be reintroduced later) which gives a dictionary for a set of images $\{I_n : n = 1, ..., N\}$. We seek a set of functions which minimize $E(\{B^k\}; \{I_n\}) = \sum_{n=1}^{N} \min_k |\vec{I_n} - \vec{B^k}|^2$.
- We can find the dictionary $\{B^k\}$ by the K-means algorithm. This is not guaranteed to find the global minimum of $E(\{B^k\}; \{I_n\})$ and hence the best dictionary. But there are efficient algorithms like k++ for initialization.
- K-means is a *clustering algorithm* because it clusters data into different subgroups (one function for each subgroup).

K-means (2)

- The input to K-means is a set f unlabeled data: $D = \{x_1, ..., x_n\}$. The goal is to decompose it into disjoint classes $w_1, ..., w_k$ where k is known. The basic assumption is that the data D is clustered round (unknown) mean values $m_1, ..., m_k$.
- We defines an association variable V_{ia}. $V_{ia} = 1$ if datapoint x_i is associated to mean m_a and $V_{ia} = 0$ otherwise. we have the constraint $\sum_a V_{ia} = 1$ for all i (i.e. each datapoint is assigned to a single mean).
- This gives a decomposition of the data. $D_a = \{i : V_{ia} = 1\}$ is the set of datapoints associated to mean m_a. The set $D = \bigcup_a D_a$ is the set of all datapoints. $D_a \bigcap D_b = \phi$ for all $a \neq b$, where ϕ is the empty set.

K-means (3)

- We defines a goodness of fit:

$$E(\{V\},\{m\}) = \sum_{i=1}^{n}\sum_{a=1}^{k} V_{ia}(x_i - m_a)^2 = \sum_{a=1}^{k}\sum_{x\in D_a}(x - m_a)^2 \qquad (1)$$

- The goal of the k-means algorithm is to minimize $E(\{V\},\{m\})$ with respect to $\{V\}$ and $\{m\}$. $E(.,.)$ is a non-convex function and no known algorithm can find its global miminum. But k-means is guaranteed to converge to a local minimum.

K-means (4)

- **The k-means algorithm**
- 1. Initialize a partition $\{D_a^0 : a = 1 \text{ to k}\}$ of the data. (I.e. randomly partition the datapoints – or use K++).
- 2. Compute the mean of each cluster D_a, $m_a = \frac{1}{|D_a|} \sum_{x \in D_a} x$.
- 3. For i=1 to n, compute $d_a(x_i) = |x_i - m_a|^2$. Assign x_i to cluster D_{a^*} s.t. $a^* = \arg\min\{d_a(x_i), ..., d_k(x_i)\}$
- 4. Repeat steps 2 & 3 until convergence.
- This will converge to a minimum of the energy function because steps 2 and 3 each decrease the energy function (or stop if the algorithm is at a local minimum). This will divide the space into disjoint regions.
- k-means can be formulated in terms of the assignment variable. At step 2, $m_a = \frac{1}{\sum_i V_{ia}} \sum_i V_{ia} x_i$. At step 3. $V_{ia} = 1$ if $|x_i - m_a|^2 = \min_b |x_i - m_b|^2$ and $V_{ia} = 0$ otherwise.

Soft K-means. Mixture of Gaussians. (1)

- Here is "softer" version of k-means (a special case of the EM) algorithm. Assign a datapoint $\underline{x}_i$ to each cluster with probability $(P_1, \ldots, P_k)$
- 1. Initialize a partition of the datapoints.
- 2. For j=1 to n. Compute the probability that x_j belongs to ω_a.
  $P(\omega_a|x_j) = \frac{\exp -\frac{1}{2\sigma^2}(x_j - m_a)^2}{\sum_b \exp -\frac{1}{2\sigma^2}(x_j - m_b)^2}$, where $P(\omega_a|x_j) = P(x_j|\omega_a)P(\omega_a)/P(x_j)$.
- 3. Compute the mean for each cluster: $m_a = \sum_j x_j P(\omega_a|x_j)$
- 4 Repeat steps 2 & 3 until convergence.
- In this version the *hard-assign* variable V_{ia} is replaced by a *soft-assign* variable $P(\omega_a|x_i)$. Observe that $\sum_a P(\omega_a|x_j) = 1$. The "softness" is controlled by σ^2. In the limit, as $\sigma^2 \mapsto 0$, the distribution $P(\omega_a|x_j)$ becomes binary valued, and soft k-means is the same as k-means.

Soft K-means. Mixture of Gaussians. (2)

- Soft k-means can be reformulated in terms of mixtures of Gaussians and the Expectation-Maximization (EM) algorithm.
- This assumes that the data is generated by a mixture of Gaussian distributions with means $\{m\}$ and variance $\sigma^2\mathbf{I}$.
  $P(x|\{V\},\{m\}) = \frac{1}{Z}\exp\{-\sum_{ia} V_{ia}\frac{||x_i - m_a||^2}{\sigma^2}\}$.
- This is equivalent to a mixture of Gaussians:
  $P(x|V,m) = \mathcal{N}(x : \sum_a V_{ia} m_a, \sigma^2)$, where the variable V identifies the mixture component (i.e. $V_{ia} = 1$ if datapoint x_i was generated by mixture a).
- We must impose a prior $P(\{V\})$ on the assignment variable V. It is natural to choose a uniform distribution $P(V) = 1/Z$, where Z is the number of possible assignments of the datapoints to the means.

Soft K-means. Mixture of Gaussians. (3)

- This gives distributions $P(x, \{V\}|\{m\}) = P(x|\{V\}, \{m\})P(\{V\})$ which enables us to use the EM algorithm which estimates the mean variables $\{m\}$ despite the presence of unknown/missing/latent variables $\{V\}$.
- The EM algorithm can be applied to any problem where there are quantities to be estimated but also missing/latent variables. It corresponds to minimizing a free energy function, which is non-convex so global convergence cannot be guaranteed. Deriving the soft k-means algorithm by applying the EM algorithm to $P(x|V, m)$ is left as an exercise for the reader.
- Soft k-means can be extended in several ways. The simplest is to treat the covariances of the Gaussians as additional variables to be estimated.

Mixture of Von Mises Fisher

- An important alternative, which arises in neural network models, is if we require that the data has unit norm $|x_i| = 1, \forall i$ (i.e. lies on the unit sphere). This can be used to avoid needing to estimate the scaling and background by setting $b = 0$ and normalize the images by $I(x) \mapsto \frac{I(x)}{|I(x)|}$ (so that $I(x)$ has unit norm).
- The Von Mises Fisher distribution is $P(x|k, \lambda_k) = \frac{\exp\{\lambda_k m_k \cdot x\}}{Z(\lambda_k)}$, where $x| = |m_k| = 1$, and σ_k is a positive constant. The mixture of Von Mises Fisher is $P(x|\{\lambda_k\}) = \sum_k P(x|k, \{\lambda_k\})P(k)$.
- Von Mises Fisher is closely related to the Gaussian distribution (with spherical covariance). The exponent of this Gaussian is $-\frac{(x-m_k)^2}{2\sigma^2}$ and if we impose $|x| = |m_k| = 1$ then this becomes $\frac{(x \cdot m_k - 1)}{\sigma^2}$ and by identifying λ_k with $1/\sigma_k^2$ we recover Von Mises Fisher. Von Mises Fisher is the a Gaussian defined of data that lies on the unit sphere.

Mini Epitomes (1)

- This is another way to learn a dictionary with a more complicated generative model with more hidden variables.. It is motivated by the fact that images are shift-invariant (unless they are carefully aligned). Recall, see powerpoints, that we want invariance to $I(x) \mapsto aI(x - x_0) + b$, where x_0 is a shift.
- Let $\{\mathbf{x}_i\}_{i=1}^N$ be a set of possibly overlapping patches of size $h \times w$ pixels cropped from a large collection of images.
- The dictionary comprises K mini-epitomes $\{\mu_k\}_{k=1}^K$ of size $H \times W$, with $H \geq h$ and $W \geq w$. The length of the vectorized patches and epitomes is then $d = h \cdot w$ and $D = H \cdot W$, respectively.
- We approximate each image patch $\mathbf{x}_i$ with its best match in the dictionary by searching over the $N_p = h_p \times w_p$ (with $h_p = H - h + 1$, $w_p = W - w + 1$) distinct sub-patches of size $h \times w$ fully contained in each mini-epitome. Typical sizes we employ are 8×8 for patches and 16×16 for mini-epitomes, implying that each mini-epitome can generate $N_p = 9 \cdot 9 = 81$ patches of size 8×8.

Mini Epitomes (2)

- We model the appearance of image patches using a Gaussian mixture model (GMM). We employ a generative model in which we activate one of the image epitomes μ_k with probability $P(l_i = k) = \pi_k$, then crop an $h \times w$ sub-patch from it by selecting the position $p_i = (x_i, y_i)$ of its top-left corner uniformly at random from any of the N_p valid positions.
- We assume that an image patch $\mathbf{x}_i$ is then conditionally generated from a multivariate Gaussian distribution $P(\mathbf{x}_i|\mathbf{z}_i, \theta) = \mathcal{N}(\mathbf{x}_i; \alpha_i \mathbf{T}_{p_i} \mu_{l_i} + \beta_i \mathbf{1}, c_i^2 \mathbf{\Sigma}_0)$.
- The label/position latent variable vector $\mathbf{z}_i = (l_i, x_i, y_i)$ controls the Gaussian mean via $\nu_{\mathbf{z}_i} = \mathbf{T}_{p_i} \mu_{l_i}$. Here $\mathbf{T}_{p_i}$ is a $d \times D$ projection matrix of zeros and ones which crops the sub-patch at position $p_i = (x_i, y_i)$ of a mini-epitome. The scalars α_i and β_i determine an affine mapping on the appearance vector and account for some photometric variability and $\mathbf{1}$ is the all-ones $d \times 1$ vector. Here $\bar{x}$ denotes the patch mean value and λ is a small regularization constant (we use $\lambda = d$ for image values between 0 and 255).

Mini Epitomes (3)

- Conceptually we formulate learning multi-epitomes as an EM problem. We specify a distribution $P(x|z,\theta)$ where x is an image patch, θ is the parameters we want to learn (the means of the mini-epitomes) and z are the latent variables, specifying which mini-epitome generated x and which location in the mini-epitome). We specify a uniform prior $p(z)$ and then apply EM. Some simplifications are made as described below.
- We choose $\pi_k = 1/K$ and fix the $d \times d$ covariance matrix $\mathbf{\Sigma}_0^{-1} = \mathbf{D}^T\mathbf{D} + \epsilon\mathbf{I}$, where $\mathbf{D}$ is the gradient operator computing the $x-$ and $y-$ derivatives of the $h \times w$ patch and ϵ is a small constant.
- To match a patch $\mathbf{x}_i$ to the dictionary, we seek the mini-epitome label and position $\mathbf{z}_i = (l_i, x_i, y_i)$, as well as the photometric correction parameters (α_i, β_i) that maximize the probability, or equivalently minimize the squared reconstruction error (note that $\mathbf{D1} = \mathbf{0}$).
- The squared reconstruction error is: $R^2(\mathbf{x}_i; k, p) = \frac{1}{c_i^2}\left(\|\mathbf{D}\left(\mathbf{x}_i - \alpha_i\mathbf{T}_p\mu_k\right)\|^2 + \lambda(|\alpha_i| - 1)^2\right)$, where the last regularization term discourages matches between patches and mini-epitomes whose contrast widely differs.

Mini Epitomes (4)

- We can compute in closed form for each candidate match $\nu_{\mathbf{z}_i} = \mathbf{T}_{p_i}\mu_{l_i}$ in the dictionary the optimal $\hat{\beta}_i = \bar{x}_i - \hat{\alpha}_i\bar{\nu}_{\mathbf{z}_i}$ and $\hat{\alpha}_i = \frac{\tilde{\mathbf{x}}_i^T\tilde{\nu}_{\mathbf{z}_i} \pm \lambda}{\tilde{\nu}_{\mathbf{z}_i}^T\tilde{\nu}_{\mathbf{z}_i} + \lambda}$, where $\tilde{\mathbf{x}}_i = \mathbf{D}\mathbf{x}_i$ and $\tilde{\nu}_{\mathbf{z}_i} = \mathbf{D}\nu_{\mathbf{z}_i}$ are the whitened patches.
- The sign in the nominator is positive if $\tilde{\mathbf{x}}_i^T\tilde{\nu}_{\mathbf{z}_i} \geq 0$ and negative otherwise. Having computed the best photometric correction parameters, we can evaluate the reconstruction error $R^2(\mathbf{x}_i; k, p)$.
- In order to learn the parameters we use the EM algorithm.Given a large training set of unlabeled image patches $\{\mathbf{x}_i\}_{i=1}^N$, our goal is to learn the maximum likelihood model parameters $\theta = (\{\pi_k, \mu_k\}_{k=1}^K)$ for the epitomic GMM model.. As is standard with Gaussian mixture model learning, we employ the EM algorithm and maximize the expected complete log-likelihood.
- The loglikelihood is $L(\theta) = \sum_{i=1}^N \sum_{k=1}^K \sum_{p\in\mathcal{P}} \gamma_i(k,p) \cdot \log\left(\pi_k \mathcal{N}\left(\mathbf{x}_i; \alpha_i \mathbf{T}_p\mu_k + \beta_{i1}c_i^2\Sigma_0\right)\right)$, where $\mathcal{P}$ is the set of valid positions in the epitome.

Mini Epitomes (5)

- In the E-step, we compute the assignment of each patch to the dictionary, given the current model parameter values. We use the hard assignment version of EM and set $\gamma_i(k,p) = 1$ if the i-th patch best matches in the p-th position in the k-th mini-epitome and 0 otherwise.
- In the M-step, we update each of the K mini-epitomes μ_k by $\left(\sum_{i,p} \gamma_i(k,p) \frac{\alpha_i^2}{c_i^2} \mathbf{T}_p^T \Sigma_0^{-1} \mathbf{T}_p\right) \mu_k = \sum_{i,p} \gamma_i(k,p) \frac{\alpha_i}{c_i^2} \mathbf{T}_p^T \Sigma_0^{-1} (\mathbf{x}_i - \bar{x}_i \mathbf{1})$.
- See powerpoints for the results.

Lecture 4

- The Expectation-Maximization Algorithm
- Super-pixels: Generative versus Affinities

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 energy function* $F(\theta, q) = -\log p(x|\theta) + \sum_h q(h) \log \frac{q(h)}{p(h|x,\theta)}$. 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 $F(\theta, q) = \sum_h q(h) \log q(h) - \sum_h q(h) \log p(h, x|\theta)$ (which exploits $p(h, x|\theta) = p(h|x, \theta)p(x|\theta$ and $\sum_h q(h) \log p(x|\theta) = \log p(x|\theta)$.

The EM Algorithm (3)

- The EM algorithm starts with an initialization. Then 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.
- We 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). A good initialization often requires studying the problem being addressed.

The EM Algorithm (4)

- 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).

The EM Algorithm (5)

- 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 (6)

- 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.

The EM Algorithm (7)

- Some general comments. The free energy $F(\theta, \{q_n()\})$ is not a convex function of $\theta, \{q\}$ so the EM algorithm is not guaranteed to converge to the global minimum. You can, however, put extra constraints on the problem which enables proofs of convergence by finding initializations which are guaranteed to lie within the domain of attraction of the global minimum.
- For k-means clustering, the K++ algorithm (handout) is a good way to initialize k-means algorithms. Otherwise try random initializations and select the one which converges to the lowest possible minimum. Or, for images, use affinity measures (see next few sides) to assign similar images to the same clusters.