CSE 659A: Advances in Computer Vision

Spring 2019: T-R: 2:30-4pm @ Cupples II/230
Instructor: Ayan Chakrabarti (ayan@wustl.edu).
http://www.cse.wustl.edu/~ayan/courses/cse659a/

Jan 24, 2019

First Homework Paper for review:

Reviews Due on Feb 7.

To be submitted on Canvas: link to paper and TeX review template is on Canvas.

RE-CAP: EM

\[ \Theta = \arg \max \sum_t \log p(X_t) = \arg \max \sum_t \log \sum_k \sigma_k f(X_t; \mu_k, \Sigma_k) \]

Expectation Maximization

**Expectation:** Given current parameters \( \Theta \), for a given patch \( X_t \), we can define \( \gamma_{tk} = p(Z = k | X_t; \Theta) \):

\[ \gamma_{tk} = \frac{p(X_t, Z_t = k)}{\sum_k p(X_t, Z_t = k')} = \frac{\sigma_k f(X_t; \mu_k, \Sigma_k)}{\sum_k \sigma_k f(X_t; \mu_k', \Sigma_k')} \]

- Basically, based on how well each mixture component explains the specific \( X_t \).
- Generate this \( \gamma \) vector for every training example.

**Maximization:** Use \( \gamma \) as proxy for “known” \( Z \), and maximize \( \Theta \) wrt that.

\[ \Theta = \arg \max \sum_t \sum_k \gamma_{tk} \log p(X_t, Z_t = k) = \arg \max \sum_t \sum_k \gamma_{tk} [\log \sigma_k + \log f(X_t; \mu_k, \Sigma_k)] \]

\[ \sigma_k = \frac{\sum \gamma_{tk}}{\sum \gamma_{tk}'}, \quad \mu_k = \frac{\sum \gamma_{tk} X_t}{\sum \gamma_{tk}}, \quad \Sigma_k = \frac{\sum \gamma_{tk} (X_t - \mu_k)(X_t - \mu_k)^T}{\sum \gamma_{tk}} \]

- Essentially, mean and co-variance are \( \gamma \)-‘weighted’ versions of the equivalent for a Gaussian.

CLASS ADMIN

- Repeat E(xpectation) and M(aximization) steps till convergence.
- Can show that each iteration increases the value of the original likelihood function.
- Guaranteed to converge. But will converge to a local optimum.
- Can be thought of as being similar to K-means. (but with 'soft-assignments')
- In practice, start with many random initializations and pick the one that converges to the best value.
- Sometimes initialize by first doing k-means (and initializing the \( \gamma \) values based on that).
- Can run different 'constrained' versions of EM: where all means are already set, and you only learn co-variances, etc.

### RE-CAP: EM

- First Homework Paper for review:
- Reviews Due on Feb 7.
- To be submitted on Canvas: link to paper and TeX review template is on Canvas.
Inference or Shrinkage (with a learned GMM model, for a patch)

\[ \frac{1}{2\sigma^2} \| Y - X \|^2 - \log p(X) = \frac{1}{2\sigma^2} \| Y - X \|^2 - \log \sum_k \sigma_k \pi_k (X; \mu_k, \Sigma_k) \]

- Unfortunately, no closed form solution to minimize this.
- Can however interpret as \( \log p(X|Y) \). In this case, this is also a GMM
  \[ p(X|Y) \propto P(X)P(Y|X) = \sum_k \sigma_k \pi_k (X; \mu_k^*, \Sigma_k^*) \]

- \( \mu_k^*, \Sigma_k^* \) are the respective posteriors of each Gaussian.
- \( \pi^* \) can be seen as the “posterior” distribution on latent variable \( Z \):
  \[ \pi_k^* = \frac{p(Y, Z = k)}{\sum_{k'} p(Y, Z = k')} = \frac{\sigma_k f(Y; \mu_k, \Sigma_k + \sigma^2 I)}{\sum_{k'} \sigma_{k'} f(Y; \mu_{k'}, \Sigma_{k'} + \sigma^2 I)} \]

- Given this is a GMM, we can compute the posterior mean in closed form \( \sum_k \sigma_k^* \mu_k^* \).
- Approximation for MAP: choose \( k \) for which \( \sum_k \sigma_k^* \mu_k^* \) is highest. Take corresponding \( \mu_k^* \).

\[ X = \arg \min_{X} \| AX - Y \|^2 - \sum_i \log p(P_i X) \]

Half-quadratic Splitting: Create an auxiliary variable \( W_i \) for each patch \( P_i X \).

\[ X = \arg \min_{X, \{W_i\}} \| AX - Y \|^2 + \frac{\beta}{2} \sum_i \| W_i - P_i X \|^2 - \sum_i \log p(W_i) \]

- Updates to \( X \): least-squares problem with squared penalties on each patch. Can write as minimizing data term, and closeness to the current estimate of each patch.

In general, \( \sum_i \| W_i - P_i X \|^2 \) can be written as \( \alpha \| X - \hat{X} \|^2 \) where \( \hat{X} \) is formed by overlap-averaging all patches \( W_i \) (and assuming \( \sum P_i^2 P_i \propto I \)).

- Updates to \( W_i \): Denoising each patch separately under the GMM prior.

\[ W_i = \arg \min_{W_i} \frac{\beta}{2} \| W_i - P_i X \|^2 - \log p(W_i) \]

- Co-variance matrices of different components learned using EM (with mean restricted to be zero, and on ‘mean’ subtracted patches)

- More principled:
  \[ X = \arg \min_{X} \| AX - Y \|^2 - \sum_i \log p(P_i X) \]

- \( X \) represents whole image. \( P_i \) is a linear operator that “crops” out the \( i^{th} \) patch.

- Solve this by half-quadratic splitting: Zoran & Weiss, EPLL, ICCV 2011.

Inference or Shrinkage (with a learned GMM model, for a patch)

\[ \frac{1}{2\sigma^2} \| Y - X \|^2 - \log p(X) = \frac{1}{2\sigma^2} \| Y - X \|^2 - \log \sum_k \sigma_k \pi_k (X; \mu_k, \Sigma_k) \]

- We’re essentially choosing a “different” Gaussian for each observed patch, and shrinking based on that.
- This is for denoising, and a shrinking each patch separately.
- One option: just denoise all overlapping patches separately, and then overlap average.
- More principled:

- Solve this by half-quadratic splitting; Zoran & Weiss, EPLL, ICCV 2011.

Source: Zoran & Weiss, 2011

\[ \frac{1}{2\sigma^2} \| Y - X \|^2 - \log p(X) = \frac{1}{2\sigma^2} \| Y - X \|^2 - \log \sum_k \sigma_k \pi_k (X; \mu_k, \Sigma_k) \]

- Co-variance matrices of different components learned using EM (with mean restricted to be zero, and on ‘mean’ subtracted patches)

- More principled:

- Solve this by half-quadratic splitting; Zoran & Weiss, EPLL, ICCV 2011.

Source: Zoran & Weiss, 2011
### GAUSSIAN MIXTURE MODEL

<table>
<thead>
<tr>
<th></th>
<th>Krishnan et al.</th>
<th>EPLL-GMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel 1 $17 \times 17$</td>
<td>25.84</td>
<td>27.17</td>
</tr>
<tr>
<td>Kernel 2 $19 \times 19$</td>
<td>26.38</td>
<td>27.70</td>
</tr>
</tbody>
</table>

- Sample results (from Zoran & Weiss)
- Was state-of-the-art for a long time!

### SPARSE DICTIONARY

$$R(X) = \min_{\alpha_i} \|X - \sum_{k=1}^{K} \alpha_k D_k\|^2,$$

such that at most $T$ values of $\alpha_i$ are non-zero.

- $\{D_k\}$ are learnable parameters. Each $D_k$ is the same shape as $X$. Number of dictionary "atoms" $K$ is greater, and $T$ much smaller, than the dimensionality of $X$.
- Basically, says every patch can be explained as a linear combination of a small number ($T$) dictionary atoms.
- If $T = 1$, says every patch should be a scaled version of one of $K$ templates.
- During inference, need to solve an optimization problem to find the best $\alpha_i$ for a given patch $X$.
- Learning $\{D_k\}$ from a training set also requires iterative optimization. Most popular algorithm: K-SVD (Aharon et al., 2006).

### CNN-DENOISER BASED PRIORS

- With half-quadratic splitting, only place we use prior is for "denoising" no matter the application.

$$X = \arg\min_X \|AX - Y\|^2 - \log p(X)$$

$$\rightarrow X = \arg\min_X \|AX - Y\|^2 + \frac{\beta}{2}\|X - Z\|^2 - \log p(Z)$$

- Use prior to solve for $Z$ in each iteration

$$Z = \arg\min_Z \frac{\beta}{2}\|X - Z\|^2 - \log p(Z)$$

Which can be seen as denoising the current estimate of $X$ assuming Gaussian noise with variance $\frac{1}{\beta}$.

- Instead of a prior $p(Z)$, learn a CNN that learns to map $X$ to $Z$ for a specific value of $\beta$.
  i.e., learn a network $f_\beta : X \rightarrow Z$.
- Train this on a dataset of image pairs $(X, Z)$, where $Z$ is a clean image and $X$ is version with noise variance $\beta^{-1}$ added.
- Need to decide your $\beta$ schedule a-priori, and then train a separate network for each value of $\beta$ you will use.


- For more info on sparse representations: CSE 585T.
MARKOV RANDOM FIELDS

- Represent joint probability distributions as an undirected graph.
- Particularly useful when you have many variables (like in an image, value at each pixel is a variable).
- Model shows conditional independence structure between variables.
- Family of MRF inference algorithms available that can be applied to many problems.
- Can be used to represent both discrete and continuous valued variables.
  - After briefly talking about a continuous model example, we will focus mainly on MRFs for discrete-valued variables.
- MRF-based models are widely used in vision algorithms even today: including in combination with CNNs.

A mathematical definition is:

\[ G = (V, E) \]

where \( V \) is a set of nodes, each corresponding to a different random variable. The edges in the graph represent the independence structure of the joint distribution of all variables. In general, all nodes depend on all other nodes even if there is no edge between them.

Under the joint distribution represented by the Graph

- Any two variables \( x_1, x_2 \) are independent, "conditioned" on all remaining variables:
  \[
  x_1 \perp x_2 \mid V \setminus \{x_1, x_2\} \\
  P(x_1, x_2 \mid V \setminus \{x_1, x_2\}) = P(x_1 \mid V \setminus \{x_1, x_2\})P(x_2 \mid V \setminus \{x_1, x_2\})
  \]

if there is no edge between them

Under the joint distribution represented by the Graph

- "Markov Blanket": Minimal set of nodes \( M(x) \) for a given node \( x \), so that \( x \) is independent of all other nodes conditioned on them.
  \[
  x \perp V \setminus \{x\} \mid M(x)
  \]
- In a MRF, the markov blanket of a node is the set of its immediate neighbors:
  \[
  M(x) = \{y : (x, y) \in E\}
  \]
Form of Probability Distribution

- Clique: Any sub-set of nodes such that all pairs have an edge between them.
- Maximal Clique: Any clique so that adding any other node would make it not a clique.

What are the maximal cliques in these graphs?

- Consider the set \( \{ C_i \} \) of all maximal cliques in the graph.
- Remember that the same node may show up in multiple maximal cliques.

Joint distribution over all variables can be factored into functions of each maximal clique:

\[
P(V) \propto \prod_i \psi_i(C_i)
\]

Each \( \psi_i \) is called a potential function.

\( Z \) is a normalizing constant so that \( P(V) \) integrates or sums to 1. Sometimes called the partition function (is a function of parameters of \( \psi_i \)).

Question: If each \( \psi_i(C_i) \) were a properly normalized probability distribution:

\[
\int \psi_i(C_i) dC_i = 1
\]

would \( Z = 1 \)?

NO! Because the same variable may show up in different cliques. (Yes, if all cliques are disjoint).

\[
P(V) = \frac{1}{Z} \prod_i \psi_i(C_i)
\]

\( Z \) is a normalizing constant so that \( P(V) \) integrates or sums to 1.

In general, very computationally expensive to calculate \( Z \). Thankfully, we don’t need it for inference.

But we do need it if we are learning \( \psi_i \) from data.

Think of the patch models we’ve been learning. You can think of each patch as a clique. We’ve been learning a common \( \psi_i(C_i) \) for all patches (as Gaussians, GMMs, etc.).

But the product of probability distributions on overlapping patches is not a "proper" distribution on the image, because its not properly normalized.