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 22, 2019
**RE-CAP**

- Gradient penalties as regularizers
- Optimization approaches for norm $p < 2$
- Gradient penalties are essentially "hand-crafted"
  - Choosing gradient filters
  - Choosing penalty function (value of $p$)
  - Choosing which gradients to "group together"
- But ideally, we would like to learn these from data.

**Learned Image Priors**

- $R(X) = -\log p(X)$, where $p(X)$ is a probability distribution learned by fitting to a training set of typical images (or depth maps, flow fields, ...)
- Once we have learned $p(X)$ (and therefore $R(X)$), we will apply it at test time for inference.
LEARNED IMAGE PRIORS

- $p(X)$: $X$ is a random vector, where each sample of $X$ is a natural image.
- Unfortunately, images themselves are too high dimensional.
- So instead, learn a distribution of image patches $X$. 
LEARNED IMAGE PRIORS

- $p(X)$: $X$ is a random vector, where each sample of $X$ is a natural image patch.
- Let's say we work with $8 \times 8$ patches. We want $p(X)$ to capture the spatial correlations between neighboring pixels in this patch.
  - We're again flattening the $8 \times 8$ patch to turn $X$ into a 64-dimensional vector.

Learning a prior

- Pick a parameteric distribution for $p$: $p(X) = f(X; \theta)$
  - Here, $\theta$ are some set of parameters based on the chosen form $f$
- Collect a training set of patches $\mathcal{T} = \{X_1, X_2, X_3, \ldots X_T \}$
- Choose the values of $\theta$ based on $\mathcal{T}$
- Most logical option: choose $\theta$ so as to maximize the likelihoods of the training set samples under $p(X)$.

$$
\theta = \arg \max \prod_t f(X_t; \theta) = \arg \max \sum_t \log f(X_t; \theta)
$$
GAUSSIAN PRIORS

- Simplest choice of parametric form.

\[
p(X) = f(X; \theta = \{\mu, \Sigma\}) = \text{det}(2\pi\Sigma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(X - \mu)^T \Sigma^{-1} (X - \mu) \right)
\]

- If \( X \) is \( d \)-dimensional (\( d = 64 \) for 8x8 patches), then \( \mu \) is the \( d \)-dimensional mean vector, and \( \Sigma \) is a \( d \times d \) symmetric positive definite matrix.

- How do we fit this to a training set?

\[
\theta = \text{arg max } \prod_t f(X_t; \theta) = \text{arg max } \sum_t \log f(X_t; \theta)
\]

\[
\mu, \Sigma = \text{arg max } \frac{-T}{2} \log \text{det}(\Sigma) - \frac{1}{2} \sum_t (X_t - \mu)^T \Sigma^{-1} (X_t - \mu)
\]

- Taking derivative and setting to 0:

\[
\mu = \frac{1}{T} \sum_t X_t; \quad \Sigma = \frac{1}{T} \sum_t (X_t - \mu)(X_t - \mu)^T
\]
GAUSSIAN PRIORS

\[ p(X) = f(X; \theta = \{\mu, \Sigma\}) = \text{det}(2\pi\Sigma)^{-\frac{1}{2}} \exp\left( -\frac{1}{2} (X - \mu)^T \Sigma^{-1} (X - \mu) \right) \]

- From training set \{X_t\}

\[ \mu = \frac{1}{T} \sum_t X_t; \quad \Sigma = \frac{1}{T} \sum_t (X_t - \mu)(X_t - \mu)^T \]

If you train this on randomly cropped patches from a set of natural images:

- You will typically get a mean vector \(\mu\) with all elements equal.
  - Kind of makes sense, randomly shifted patches, no reason any location in the patch should have a different mean. . . .
- We can also do an eigen-decomposition of \(\Sigma\) and look at the eigenvalues and eigenvectors.
  - \(\Sigma = VDV^T\). Columns of \(V\) are eigen-vectors, and \(D\) is diagonal matrix with eigenvalues.
- You will find that the eigenvalues decay very quickly
  - Second largest eigenvalue much smaller than largest. Third much smaller than second, etc.
- Each eigen-vector is a patch-shaped object. So we can visualize them.
GAUSSIAN PRIORS

- Eigenvectors of Covariance matrix

Looks like a Fourier Basis!

First eigen-vector also has all equal values. Corresponds to patch mean.

\[ \mu \propto V_1 \]

\[ V_i^T \mu = 0 \text{ for } i > 1 \]
GAUSSIAN PRIORS

\[ p(X) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2} (X - \mu)^T \Sigma^{-1} (X - \mu)\right) \]

\[ \Sigma = VDV^T \]

\[ R(X) = -\log p(X) \]

Ignoring terms that don't depend on \( X \)

\[ R(X) = \frac{1}{2} (X - \mu)^T \Sigma^{-1} (X - \mu) \]

\[ R(X) = \frac{1}{2} (X - \mu)^T (VD^{-1}V^T) (X - \mu) \quad \text{Remember, } V^TV = I, V^{-1} = V^T \]

\[ R(X) = \frac{1}{2D_{11}} \|V_1^TX - V_1^T\mu\|^2 + \sum_{i=2}^{d} \frac{1}{2D_{ii}} \|V_i^TX\|^2 \]

- The first term corresponds to the patch mean. \( D_{11} \) is really large, so its reciprocal is really small.
- Also, this "DC" component behaves least like a Gaussian. So with Gaussian priors, we often don't include the patch mean in the regularization.
GAUSSIAN PRIORS

\[ p(X) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left( -\frac{1}{2} (X - \mu)^T \Sigma^{-1} (X - \mu) \right) \]

\[ \Sigma = VDV^T \]

\[ R(X) = \sum_{i=2}^{d} \frac{1}{2D_{ii}} \|V_i^T X\|^2 \]

- Takes us back to a squared-penalty regularizer.

**Vector to Spatial Representation**

\[ V_i^T X = \sum_n V_i[n] X[n] \]

- \( V_i \) for \( i > 2 \) correspond to high-frequency components of \( X \). In other words, patch gradients.

- In fact, if we applied \( R(\cdot) \) on all overlapping patches in an image, \( V_i^T X \) would correspond to convolving the image with \( V_i \) (reshaped to 8x8) as a filter.

- So we are back to squared penalty on gradients ...

- But this time, we're "learning" the set of gradient filters and corresponding weights \( \lambda \propto D_{ii}^{-1} \).
Bayesian Interpretation (for Denoising)

\[ \Phi(X) + R(X) = \frac{1}{2\sigma^2} \|X - Y\|^2 - \log p(X) \]

\[ \Rightarrow p(X|Y) \propto \exp \left( -\frac{\|X - Y\|^2}{2\sigma^2} \right) \exp \left( -\frac{1}{2} (X - \mu)^T \Sigma^{-1} (X - \mu) \right) \]

- Turns out \( p(X|Y) \) is also a Gaussian distribution. Fit the two square terms to a single square term:

\[
\mu_{X|Y} = \left( \Sigma^{-1} + \sigma^{-2} I \right)^{-1} \left( \frac{Y}{\sigma^2} + \Sigma^{-1} \mu \right)
\]

\[
\Sigma_{X|Y} = \left( \Sigma^{-1} + \sigma^{-2} I \right)^{-1}
\]

- Two kinds of estimators:
  - MAP: \( \arg \max_X p(X|Y) \)
  - Posterior Mean: \( \int X p(X|Y) dX \)

- For a Gaussian distribution, both are the same (= what you get from minimizing \( \Phi(X) + R(X) \))

- Reasonable first step, but we know Gaussians are a poor fit to the distributions of natural images.

- This regularizer won't give us the kind of "shrinkage" behavior we want.
GAUSSIAN MIXTURE MODEL

\[ p(X) = \sum_{k=1}^{K} \pi_k f(x; \mu_k, \Sigma_k) \]

- Each scalar \( \pi_k \geq 0, \sum_k \pi_k = 1 \)
- Each \( f \) represents a Gaussian distribution

\[
f(x; \mu_k, \Sigma_k) = \det(2\pi \Sigma_k)^{-\frac{1}{2}} \exp \left( -\frac{1}{2} (X - \mu_k)^T \Sigma_k^{-1} (X - \mu_k) \right)
\]

- So \( p(X) \) is a weighted sum of individual Gaussians, where the weights sum to 1.
- This is a properly normalized probability distribution: \( \int p(X) dX = 1 \). Why?

\[
\int p(X) dX = \int \left( \sum_k \pi_k f(x; \mu_k, \Sigma_k) \right) dX
\]

\[
eq \sum_k \pi_k \int f(x; \mu_k, \Sigma_k) dX = \sum_k \pi_k \int 1 = 1
\]
GAUSSIAN MIXTURE MODEL

\[ p(X) = \sum_{k=1}^{K} \pi_k f(x; \mu_k, \Sigma_k) \]
\[ f(x; \mu_k, \Sigma_k) = \text{det}(2\pi \Sigma_k)^{-\frac{1}{2}} \exp\left(-\frac{1}{2} (X - \mu_k)^T \Sigma_k^{-1} (X - \mu_k)\right) \]

- What is the mean of this distribution? \( \mathbb{E}X \)

\[ \int X \, p(X) \, dX = \sum_k \pi_k \mu_k \]

- What is the co-variance of this distribution? \( \mathbb{E}(X - \mu)(X - \mu)^T \)

\[ \sum_k \pi_k \Sigma_k + \sum_k \pi_k (\mu_k - \mu)(\mu_k - \mu)^T \]
GAUSSIAN MIXTURE MODEL

Scalar X

p(X)
GAUSSIAN MIXTURE MODEL

Scalar X

p(X)

x
GAUSSIAN MIXTURE MODEL

Each ellipse represents the mean and covariance of a different Gaussian component

$$(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) = 1$$

Each component can correspond to a different mean, different eigen-vectors and eigen-values (rotation/skew)
Each ellipse represents the mean and covariance of a different Gaussian component

\[(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) = 1\]

Each component can correspond to a different mean, different eigen-vectors and eigen-values (rotation/skew)

They can also have the same mean, but different co-variance matrices. (Represent different sets of gradients that co-occur in image patches)
GAUSSIAN MIXTURE MODEL

Each ellipse represents the mean and covariance of a different Gaussian component

\[(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) = 1\]

Each component can correspond to a different mean, different eigen-vectors and eigen-values (rotation/skew)

They can also have the same mean, but different co-variance matrices. (Represent different sets of gradients that co-occur in image patches)

They can also have the same means and different 'scaled' versions of the same co-variance matrix.

Think of as saying the k-mixtures represent different kinds of patches: low-magnitude, medium-magnitude, and high-magnitude gradients.
GAUSSIAN MIXTURE MODEL

\[ p(X) \]
GAUSSIAN MIXTURE MODEL
GAUSSIAN MIXTURE MODEL

In general, with sufficient components (large enough $K$), a Gaussian mixture model can approximate any distribution!

Training

$$p(X) = \sum_{k=1}^{K} \pi_k f(x; \mu_k, \Sigma_k)$$

$$f(x; \mu_k, \Sigma_k) = \text{det}(2\pi\Sigma_k)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(X - \mu_k)^T\Sigma_k^{-1}(X - \mu_k)\right)$$

- Learnable parameters: $\Theta = \{\pi_k, \mu_k, \Sigma_k\}$
- Given a training set $\mathcal{T} = \{X_1, X_2, \ldots X_T\}$, how do we learn these parameters?

$$\Theta = \arg\max_\Theta \sum_t \log p(X_t) = \arg\max_\Theta \sum_t \log \sum_k \pi_k \text{det}(2\pi\Sigma_k)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(X_t - \mu_k)^T\Sigma_k^{-1}(X_t - \mu_k)\right)$$
GAUSSIAN MIXTURE MODEL

\[ \Theta = \arg \max_{\Theta} \sum_{t} \log p(X_t) = \arg \max_{\Theta} \sum_{t} \log \sum_{k} \pi_k f(X_t; \mu_k, \Sigma_k) \]

- Cost function is non-convex. (Sum-Log-Sum-Exp)

**Expectation Maximization**

- Think of \( Z = \{1, 2, \ldots k\} \) as a latent discrete variable which says which component an \( X \) belongs to.

\[
p(X) = \sum_{k=1}^{K} p(X, Z = k), \quad p(X, Z = k) = p(Z = k) \quad p(X|Z = k) = \pi_k \quad f(X; \mu_k, \Sigma_k)
\]

- Generative model: To generate a random patch \( X \), we first sample the value of \( Z \) according to the multinomial distribution \( \pi_k \). Then, we sample a value of \( X \) from the corresponding Gaussian.

- Say, in our training set, we knew the "true" value of this latent variable \( Z_t \) for every patch \( X_t \). Training would then try to maximize:

\[
\Theta = \arg \max_{\Theta} \sum_{t} \log p(X_t, Z_t|\Theta) = \arg \max_{\Theta} \sum_{t} \left[ \log \pi_{Z_t} + \log f(X_t; \mu_{Z_t}, \Sigma_{Z_t}) \right]
\]
GAUSSIAN MIXTURE MODEL

\[ \Theta = \arg \max_{\Theta} \sum_t \log p(X_t) = \arg \max_{\Theta} \sum_t \log \sum_k \pi_k f(X_t; \mu_k, \Sigma_k) \]

Expectation Maximization

• Say, in our training set, we knew the "true" value of this latent variable \( Z_t \) for every patch \( X_t \). Training would then try to maximize:

\[ \Theta = \arg \max \sum_t \log p(X_t, Z_t | \Theta) = \arg \max \sum_t [\log \pi_{Z_t} + \log f(X_t; \mu_{Z_t}, \Sigma_{Z_t})] \]

• Another way of writing this is:

\[ \Theta = \arg \max \sum_t \sum_k \gamma_{t:k} [\log \pi_k + \log f(X_t; \mu_k, \Sigma_k)] \]

where \( \gamma_{t:k} \) is a one-hot vector: 1 if \( Z_t = k \) and 0 otherwise.

• But we don't have \( Z_t \). Solution: replace \( \gamma_{t:k} \) with distribution over values of \( Z_t \), i.e., \( P(Z_t = k|X_t, \Theta) \), based on the "current" values of \( \Theta \)

• And maximize with respect to that!

• Repeat this iteratively.
GAUSSIAN MIXTURE MODEL

\[ \Theta = \arg \max_{\Theta} \sum_t \log p(X_t) = \arg \max_{\Theta} \sum_t \log \sum_k \pi_k f(X_t; \mu_k, \Sigma_k) \]

Expectation Maximization

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

\[
\gamma_{t:k} = \frac{p(X_t, Z_t = k)}{\sum_{k'} p(X_t, Z_t = k')} = \frac{\pi_k f(X_t; \mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} f(X_t; \mu_{k'}, \Sigma_{k'})}
\]

- 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.

\[
\Theta = \arg \max_{\Theta} \sum_t \sum_k \gamma_{t:k} \log P(X_t, Z_t = k) = \arg \max_{\Theta} \sum_t \sum_k \gamma_{t:k} \left[ \log \pi_k + \log f(X_t; \mu_k, \Sigma_k) \right]
\]

\[
\pi_k = \frac{\sum_t \gamma_{t:k}}{\sum_k \sum_t \gamma_{t:k'}}, \quad \mu_k = \frac{\sum_t \gamma_{t:k} X_t}{\sum_t \gamma_{t:k}}, \quad \Sigma_k = \frac{\sum_t \gamma_{t:k} (X_t - \mu_k)(X_t - \mu_k)^T}{\sum_t \gamma_{t:k}}
\]

- Essentially, mean and co-variance are \( \gamma \)—'weighted' versions of the equivalent for a Gaussian.
GAUSSIAN MIXTURE MODEL

- 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.