RE-CAP

- Gradient penalties as regularizers
- 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, ...)
- Choose parametric for \( p(X) \), then fit parameters to a training set.
- Looked at the multi-variate Gaussian distribution. Saw that it corresponds to a "squared penalty" \( R(X) \), but with learned gradient filters and learned weights on each filter.

GAUSSIAN PRIORS

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:

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

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

- Two kinds of estimators:
  - MAP: \( \text{arg max}_X p(X|Y) \)
  - 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?

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

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

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

- 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
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 eigenvectors, and eigenvalues (rotation/skew).

They can also have the same mean, but different covariance matrices.

(Represent different sets of gradients that co-occur in image patches)

GM Mixture Model

GM Mixture Model

GM Mixture Model

GM Mixture Model

p(X)

p(X)

Think of each Gaussian component as representing different kinds of patches, by magnitude, medium magnitude, and high magnitude gradients.
**GAUSSIAN MIXTURE MODEL**

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

Training

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

$$f(x; \mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^d | \Sigma_k |}} \exp\left(-\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \right)$$

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

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

**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:

$$\Theta = \arg\max_{\Theta} \sum_{t} \log p(X_t, Z_t | \Theta) = \arg\max_{\Theta} \sum_{t} \left[ \log \pi_k + \log f(X_t; \mu_k, \Sigma_k) \right]$$

- Another way of writing this is:

$$\Theta = \arg\max_{\Theta} \sum_{t} \sum_{k} \gamma_{tk} \left[ \log \pi_k + \log f(X_t; \mu_k, \Sigma_k) \right]$$

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

- But we don’t have $Z_t$. Solution: replace $\gamma_{tk}$ 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.

**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.

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

- 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:

$$\Theta = \arg\max_{\Theta} \sum_{t} \log p(X_t, Z_t | \Theta) = \arg\max_{\Theta} \sum_{t} \left[ \log \pi_k + \log f(X_t; \mu_k, \Sigma_k) \right]$$

**GAUSSIAN MIXTURE MODEL**

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

**Expectation Maximization**

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

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

- 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.

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

- 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 \( y \) values based on that).
- Can run different ‘constrained’ versions of EM: where all means are already set, and you only learn co-variances, etc.

...more details...

**Source:** Zoran & Weiss, 2011

---

**GAUSSIAN MIXTURE MODEL**

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

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

- Unfortunately, no closed form solution to minimize this.
- Can however interpret as \( \log p(X|Y) \). In this case, this is also a GMM

\[
p(X|Y) \propto P(X|Y) = \sum_k \pi_k f(X; \mu_k, \Sigma_k)
\]

- \( \mu_k, \Sigma_k \) are the respective posteriors of each Gaussian.
- \( \pi^* \) can be seen as the "posterior" distribution on latent variable \( Z \):

\[
\pi^*_k = \frac{p(Y, Z = k)}{\sum_{k'} p(Y, Z = k')} = \frac{\pi_k f(Y; \mu_k', \Sigma_k' + \sigma^2 I)}{\sum_{k'} \pi_{k'} f(Y; \mu_{k'}, \Sigma_{k'} + \sigma^2 I)}
\]

- Given this is a GMM, we can compute the posterior mean in closed form \( \sum_k \pi_k^* \mu_k^* \).

- Approximation for MAP: choose \( k \) for which \( \pi_k^* \) is highest. Take corresponding \( \mu_k^* \).

**Source:** Zoran & Weiss, EPLL, ICCV 2011.
GAUSSIAN MIXTURE MODEL

$$X = \arg \min_{X} \|AX - Y\|^2 - \sum_{i} \log p(P_iX)$$

Half-quadratic Splitting: Create an auxiliary variable $W_i$ for each patch $P_iX$.

$$X = \arg \min_{X,(W_i)} \|AX - Y\|^2 + \frac{\beta}{2} \sum_{i} \|W_i - P_iX\|^2 - \sum_{i} \log p(W_i)$$

- 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.

$$X = \arg \min_{X,(W_i)} \|AX - Y\|^2 + \frac{\beta}{2} \sum_{i} \|W_i - P_iX\|^2$$

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

- Updates to $W_i$: Denoising each patch separately under the GMM prior.

$$W_i = \arg \min_{W_i} \frac{\beta}{2} \|W_i - P_iX\|^2 - \log p(W_i)$$

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