Statistical Machine Learning (BE4M33SSU)
Lecture 9: EM algorithm; Bayesian learning

Czech Technical University in Prague

- Expectation Maximisation algorithm
- Bayesian inference
- Variational Bayesian inference

1. The Expectation Maximisation Algorithm

Unsupervised generative learning:

- The joint p.d. $p_\theta(x,y)$, $\theta \in \Theta$ is known up to the parameter $\theta \in \Theta$,
- given training data $\mathcal{T}^m = \{x^j \in \mathcal{X} \mid i = 1,2,\ldots,m\}$ i.i.d. generated from $p_{\theta*}$.

How shall we implement the MLE

$$e_{ML}(\mathcal{T}^m) = \arg\max_{\theta\in\Theta} \frac{1}{m} \sum_{x\in\mathcal{T}^m} \log p_\theta(x) = \arg\max_{\theta\in\Theta} \mathbb{E}_{\mathcal{T}^m}\Big[\log \sum_{y\in\mathcal{Y}} p_\theta(x,y)\Big]$$

- If θ is a single parameter or a vector of homogeneous parameters $\Rightarrow$ maximise the log-likelihood directly.
- If θ is a collection of heterogeneous parameters $\Rightarrow$ apply the **Expectation Maximisation Algorithm** (Schlesinger, 1968, Sundberg, 1974, Dempster, Laird, and Rubin, 1977)

1. The Expectation Maximisation Algorithm

EM algorithm:

- Introduce auxiliary variables $\alpha_x(y) \geqslant 0$, for each $x \in \mathcal{T}^m$, s.t. $\sum_{y\in\mathcal{Y}} \alpha_x(y) = 1$
- Construct a lower bound of the log-likelihood $L(\theta, \mathcal{T}^m) \geqslant L_B(\theta, \alpha, \mathcal{T}^m)$
- Maximise this lower bound by block-wise coordinate ascent.

Construct the bound:

$$L(\theta,\mathcal{T}^m) = \mathbb{E}_{\mathcal{T}^m}\Big[\log \sum_{y\in\mathcal{Y}} p_\theta(x,y)\Big] = \mathbb{E}_{\mathcal{T}^m}\Big[\log \sum_{y\in\mathcal{Y}} \frac{\alpha_x(y)}{\alpha_x(y)} p_\theta(x,y)\Big] \geqslant$$

$$L_B(\theta,\alpha,\mathcal{T}^m) = \mathbb{E}_{\mathcal{T}^m} \sum_{y\in\mathcal{Y}} \Big[\alpha_x(y)\log p_\theta(x,y) - \alpha_x(y)\log\alpha_x(y)\Big]$$

The following equivalent representation shows the difference between $L(\theta,\mathcal{T}^m)$ and $L_B(\theta,\alpha,\mathcal{T}^m)$:

$$L_B(\theta,\alpha,\mathcal{T}^m) = \mathbb{E}_{\mathcal{T}^m}\big[\log p_\theta(x)\big] - \mathbb{E}_{\mathcal{T}^m}\big[D_{KL}(\alpha_x(y) \,\|\, p_\theta(y\,|\,x))\big]$$

We see that the lower bound is tight if $\alpha_x(y) = p_\theta(y \mid x)$ holds $\forall x$ and $\forall y$.

1. The Expectation Maximisation Algorithm

Maximise $L_B(\theta, \alpha, \mathcal{T}^m)$ by block-coordinate ascent:

Start with some $\theta^{(0)}$ and iterate

E-step Fix the current $\theta^{(t)}$, maximise $L_B(\theta^{(t)}, \alpha, \mathcal{T}^m)$ w.r.t. α-s. This gives

$$\alpha_x^{(t)}(y) = p_{\theta^{(t)}}(y \mid x).$$

M-step Fix the current $\alpha^{(t)}$ and maximise $L_B(\theta, \alpha^{(t)}, \mathcal{T}^m)$ w.r.t. θ.

$$\theta^{(t+1)} = \arg\max_{\theta \in \Theta} \mathbb{E}_{\mathcal{T}^m} \Big[\sum_{y \in \mathcal{Y}} \alpha_x^{(t)}(y) \log p_\theta(x, y) \Big]$$

This is equivalent to solving the MLE for annotated training data.

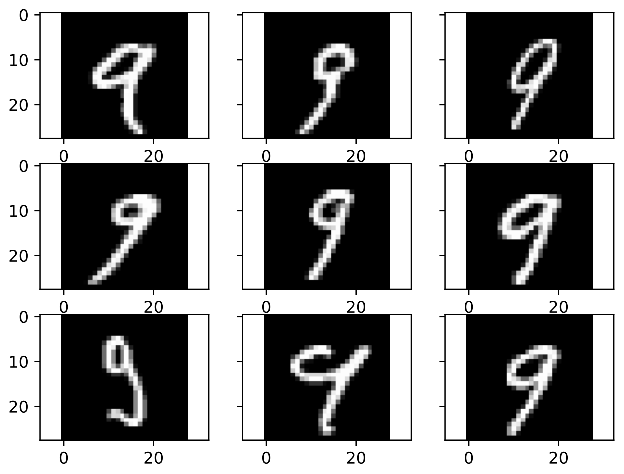
Claims:

- The sequence of likelihood values $L(\theta^{(t)}, \mathcal{T}^m)$, $t = 1, 2, \ldots$ is increasing, and the sequence $\alpha^{(t)}$, $t = 1, 2, \ldots$ is convergent (under mild assumptions).
- There is no guarantee that the EM algorithm converges to a global maximum.
- It is important to use a proper initialisation.

1. The Expectation Maximisation Algorithm

Example: Latent mode model (mixture) for images of digits

- $x = \{x_i \mid i \in D\}$ image on the pixel domain $D \in \mathbb{Z}^2$,
- $x_i \in \{0, 1, 2, \ldots, 255\}$
- $k \in K$ latent variable (mode indicator),
- joint distribution - Naive Bayes model



$$p(x, k) = p(k) \prod_{i \in D} p(x_i \mid k)$$

Learning problem: Given i.i.d. training data $\mathcal{T}^m = \{x^j \mid j = 1, 2, \ldots, m\}$, estimate the mode probabilities $p(k)$ and the conditional probabilities $p(x_i \mid k)$, $\forall x_i \in \mathcal{B}$, $k \in K$ and $i \in D$.

1. The Expectation Maximisation Algorithm

Applying the EM algorithm: Start with some model $p^{(0)}(k)$, $p^{(0)}(x_i \mid k)$ and iterate the following steps until convergence.

E-step Given the current model estimate $p^{(t)}(k)$, $p^{(t)}(x_i \mid k)$, compute the posterior mode probabilities for each image x in the training data $\mathcal{T}^m$

$$\alpha_x^{(t)}(k) = p^{(t)}(k \mid x) = \frac{p^{(t)}(k) \prod_{i \in D} p^{(t)}(x_i \mid k)}{\sum_{k'} p^{(t)}(k') \prod_{i \in D} p^{(t)}(x_i \mid k')}.$$

M-step Re-estimate the model by solving

$$\mathbb{E}_{\mathcal{T}^m}\Big[\sum_{k \in K} \alpha_x^{(t)}(k) \big[\log p(k) + \sum_{i \in D} \log p(x_i \mid k)\big]\Big] \rightarrow \max_p$$

This gives

$$p^{(t+1)}(k) = \mathbb{E}_{\mathcal{T}^m}\big[\alpha_x^{(t)}(k)\big]$$

$$p^{(t+1)}(x_i = b \mid k) = \frac{\mathbb{E}_{\mathcal{T}^m}\big[\alpha_x^{(t)}(k) \bigm| x_i = b\big]}{\mathbb{E}_{\mathcal{T}^m}\big[\alpha_x^{(t)}(k)\big]}$$

1. The Expectation Maximisation Algorithm

Additional reading:

Schlesinger, Hlavac, Ten Lectures on Statistical and Structural Pattern Recognition, Chapter 6, Kluwer 2002 (also available in Czech)

Thomas P. Minka, Expectation-Maximization as lower bound maximization, 1998 (short tutorial, available on the internet)

2. Bayesian Inference

Motivation:

- Both, ERM and generative learning by MLE are consistent under the respective regularity assumptions. Their estimation errors $R(h_m) - R(h_{\mathcal{H}})$ and $\|\theta_m - \theta^*\|$ are small in the limit of large training data sizes m. On the other hand, their estimates h_m and θ_m can deviate by large margin from the respective optima in case of small training data sizes.

  Example: We want to learn deep NNs with $> 10^6$ parameters on training data $\mathcal{T}^m$ with $m < 10^6$.

- Models should be based on our knowledge about the problem. E.g. we do not want to restrict the complexity of the model $p_\theta(x, y)$, $\theta \in \Theta$ just because we have only a small amount of training data.

2. Bayesian inference

Bayesian inference: main assumptions & ingredients

Interpret the unknown parameter $\theta \in \Theta$ as a **random** variable.

- Data distribution: parametric family of models $p(x, y \mid \theta)$, $\theta \in \Theta$,
- Prior distribution $p(\theta)$ on Θ.

Prior distribution $p(\theta)$ and i.i.d. training data $\mathcal{T}^m = \{(x_i, y_i) \mid i = 1, \ldots, m\} \Rightarrow$ *posterior parameter distribution* $p(\theta \mid \mathcal{T}^m)$, given by

$$p(\theta \mid \mathcal{T}^m) = \frac{p(\theta) p(\mathcal{T}^m \mid \theta)}{p(\mathcal{T}^m)} \quad \text{with} \quad p(\mathcal{T}^m \mid \theta) = \prod_{i=1}^{m} p(x^i, y^i \mid \theta).$$

Notice:

- a point estimate of θ is no longer needed!
- the posterior distribution $p(\theta \mid \mathcal{T}^m) \propto p(\mathcal{T}^m \mid \theta)\, p(\theta)$ interpolates between the situation without any training data, i.e. $m = 0$ and the likelihood of training data for $m \to \infty$.

2. Bayesian inference

Example 1. Consider the model

$$p(x\,|\,\mu) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2\sigma^2}(x-\mu)^2\right] \quad \text{and} \quad p(\mu) = \frac{1}{\sqrt{2\pi}\sigma_0}\exp\left[-\frac{1}{2\sigma_0^2}\mu^2\right].$$

then we have

$$p(x,\mu) = p(x\,|\,\mu)p(\mu) = \frac{1}{2\pi\sigma\sigma_0}\exp\left[-\frac{1}{2\sigma^2}(x-\mu)^2 - \frac{1}{2\sigma_0^2}\mu^2\right],$$

$$p(x) = \int_{\mathbb{R}} p(x\,|\,\mu)\,p(\mu)\,d\mu = \frac{1}{\sqrt{2\pi(\sigma^2+\sigma_0^2)}}\exp\left[-\frac{x^2}{2(\sigma^2+\sigma_0^2)}\right]$$

$$p(\mu\,|\,x) = \frac{p(x\,|\,\mu)p(\mu)}{p(x)} \propto \exp\left[-\frac{x^2}{2(\sigma^2+\sigma_0^2)} - \frac{1}{2\sigma^2}(x-\mu)^2 - \frac{1}{2\sigma_0^2}\mu^2\right]$$

Notice the difference when estimating μ from a single example x:

- $e_{ML}(x) = x$.
- $\arg\max_\mu p(\mu\,|\,x) = \frac{1}{1+\sigma^2/\sigma_0^2}x$.

2. Bayesian inference

Sidestep: We consider θ as random with prior distribution $p(\theta)$, but go for a point estimate given training data $\mathcal{T}^m = \{(x_i, y_i) \mid i = 1, \ldots, m\}$:

$$\theta_m = \arg\max_{\theta \in \Theta} p(\theta \mid \mathcal{T}^m) = \arg\max_{\theta \in \Theta} p(\mathcal{T}^m \mid \theta)\, p(\theta) = \arg\max_{\theta \in \Theta} \sum_{(x,y) \in \mathcal{T}^m} \log p(x, y \mid \theta) + \log p(\theta)$$

This results in an ML estimate with an additional regulariser

$$\theta_m = \arg\max_{\theta \in \Theta} \Big[\frac{1}{m} \sum_{(x,y) \in \mathcal{T}^m} \log p(x, y \mid \theta) + \frac{1}{m} \log p(\theta) \Big]$$

Example 2. We want to learn a DNN classifier with squashing activation functions (e.g. tanh or sigmoid). Assuming a Gaussian prior $\mathcal{N}(0, \sigma)$ for the network weights, we get the learning objective

$$\frac{1}{m} \sum_{(x,y) \in \mathcal{T}^m} \log p(y \mid x, w) - \frac{1}{2m\sigma^2} \|w\|^2 \rightarrow \max_{w}$$

This enforces a considerable fraction of neurons to have small weights and thus also small activations. They will therefore operate in a quasi linear regime.

2. Bayesian inference

Retaining the posterior distribution $p(\theta \,|\, \mathcal{T}^m) \propto p(\mathcal{T}^m \,|\, \theta)\, p(\theta)$, we get the posterior probability to observe a pair (x, y) by marginalising over $\theta \in \Theta$:

$$p(x, y \,|\, \mathcal{T}^m) = \frac{1}{p(\mathcal{T}^m)} \int_{\Theta} p(\mathcal{T}^m \,|\, \theta)\, p(x, y \,|\, \theta)\, p(\theta)\, d\theta$$

This is a mixture of distributions with mixture weights $\alpha_m(\theta) \propto p(\mathcal{T}^m \,|\, \theta)\, p(\theta)$.

The Bayes optimal predictor w.r.t. 0/1 loss for this model mixture is

$$h(x, \mathcal{T}^m) = \arg\max_{y \in \mathcal{Y}} \int_{\Theta} \underbrace{p(\theta)\, p(\mathcal{T}^m \,|\, \theta)}_{\alpha_m(\theta) \propto} p(x, y \,|\, \theta)\, d\theta = \arg\max_{y \in \mathcal{Y}} \int_{\Theta} \alpha_m(\theta)\, p(x, y \,|\, \theta)\, d\theta$$

Notice:

- the mixture weights $\alpha_m(\theta)$ interpolate between the situation without any training data, i.e. $m = 0$ and the likelihood of training data for $m \to \infty$.
- similar approaches for ERM lead to *Ensembling* methods (see lectures 12,13).

3. Variational Bayesian inference

Computing integrals like $\int_\Theta p(\mathcal{T}^m \,|\, \theta)\, p(\theta)\, d\theta$ is in most cases not tractable.

Variational Bayesian inference: Approximate $p(\theta \,|\, \mathcal{T}^m)$ by some simple distribution $q_\beta(\theta)$ and find the optimal parameter β by minimising the Kullback-Leibler divergence

$$D_{KL}(q_\beta(\theta) \,\|\, p(\theta \,|\, \mathcal{T}^m)) = D_{KL}(q_\beta(\theta) \,\|\, p(\theta)) - \int_\Theta q_\beta(\theta) \log p(\mathcal{T}^m \,|\, \theta)\, d\theta + c \to \min_\beta$$

use $q_\beta(\theta)$ with optimal β for prediction (e.g. for 0/1 loss)

$$h(x) = \arg\max_y \int_\Theta q_\beta(\theta)\, p(x, y \,|\, \theta)\, d\theta$$

The integrals over θ can be often further simplified by sampling $\theta_i \sim q_\beta(\theta)$

$$\int_\Theta q_\beta(\theta)\, f(\theta)\, d\theta \approx \frac{1}{m} \sum_{i=1}^{n} f(\theta_i)$$

3. Variational Bayesian inference

Example 3 (Bayesian inference for DNNs). Let us consider the optimisation task

$$\int_{\mathbb{R}^n} q_\mu(w) \log p(\mathcal{T}^m \,|\, w)\, dw - D_{KL}(q_\mu(w) \,\|\, p(w)) \to \max_\mu$$

for the following situation & assumptions:

- $p(y \,|\, x, w)$ is a classifier DNN with weights w, i.e.

  $$p(y \,|\, x, w) = \big\langle y, \operatorname{softmax}\big(\eta(x, w)\big)\big\rangle$$

  where y is the one-hot encoding of the class and $\eta(x, w)$ is the network output layer pre-activation.
- The prior distribution for the weights is $p(w) = \mathcal{N}(w; 0, \mathbb{I})$.
- We approximate the posterior weight distribution by $q_\mu(w) = \mathcal{N}(w; \mu, \mathbb{I})$

3. Variational Bayesian inference

The training objective (variational Bayesian inference) is:

$$\int_{\mathbb{R}^n} q_\mu(w) \log p(\mathcal{T}^m \,|\, w)\, dw - D_{KL}(q_\mu(w) \,\|\, p(w)) \to \max_\mu ,$$

where $\mathcal{T}^m$ denotes i.i.d. training data. We have

$$\mathbb{E}_{q_\mu(w)}\big[\log p(\mathcal{T}^m \,|\, w)\big] - D_{KL}(\mathcal{N}(\mu, \mathbb{I}) \,\|\, \mathcal{N}(0, \mathbb{I})) \to \max_\mu$$

This task can be solved by SGD w.r.t. mini-batches and sampled network weights.

- the KL-divergence can be computed in closed form,
- approximate the integral in the first term by sampling from $q_\mu(w) = \mathcal{N}(w; \mu, \mathbb{I})$ (with current $\mu^{(t)}$),
- to compute gradients w.r.t. μ, apply re-parametrisation

$$w \sim \mathcal{N}(\mu, \mathbb{I}) \Leftrightarrow w = \epsilon + \mu \text{ with } \epsilon \sim \mathcal{N}(0, \mathbb{I})$$

The SGD step reads: sample a mini-batch, sample $\epsilon \sim \mathcal{N}(0, \mathbb{I})$, set $w = \mu^{(t)} + \epsilon$, apply the network and compute the gradient w.r.t. μ and apply a learning step $\Rightarrow \mu^{(t+1)}$.