Statistical Machine Learning (BE4M33SSU)
Lecture 7: Generative learning, EM-Algorithm

Czech Technical University in Prague

- Generative vs. Discriminative Learning
- Maximum Likelihood Estimator, consistency
- Expectation Maximisation Algorithm

1. Generative vs. Discriminative Learning

Generative learning:

- Model the **joint** probability distributions $p_\theta(x,y)$ for features $x \in \mathcal{X}$ and hidden states $y \in \mathcal{Y}$. The distributions are parametrised by $\theta \in \Theta$.
- Inference rule (if true parameter θ_0 is known):

$$h(x) \in \arg\max_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} p_{\theta_0}(y' \mid x) \ell(y', y)$$

- Learning: if $\theta_0 \in \Theta$ is not known, estimate it from training data $\mathcal{T}^m = \big\{(x^i, y^i) \in \mathcal{X} \times \mathcal{Y} \mid i = 1, \ldots, m\big\}$ e.g. by Maximum Likelihood estimator.

Discriminative learning(1):

- Model only the **conditional** distributions $p_\theta(y \mid x)$, $\theta \in \Theta$.
- Inference rule (if true parameter θ_0 is known): as above
- Learning: if $\theta_0 \in \Theta$ is not known, estimate it by maximising the conditional likelihood on the training data $\mathcal{T}^m$.

$$\theta^* \in \arg\max_{\theta \in \Theta} \sum_{i=1}^{m} \log p_\theta(y^i \mid x^i)$$

1. Generative vs. Discriminative Learning

Discriminative learning(2):

- Model the class of inference rules $h \in \mathcal{H}$ directly.
- Optimal inference (if $p(x,y)$ is known):

$$h_0(x) = \operatorname*{arg\,min}_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} p(x,y')\ell(y',y)$$

- Estimate the best inference rule $h^* \in \mathcal{H}$ by minimising the empirical risk on the training data

$$h^* \in \operatorname*{arg\,min}_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \ell(y^i, h(x^i))$$

1. Generative vs. Discriminative Learning

Example (Gaussian Discriminative Analysis, Logistic Regression, Linear Classifier)

$y = 0, 1$, $y \sim Bernoulli(\alpha)$ and $x \in \mathbb{R}^n$, $x \mid y = 0 \sim \mathcal{N}(\mu_0, V)$, $x \mid y = 1 \sim \mathcal{N}(\mu_1, V)$, i.e.

$$p(y) = \alpha^y (1 - \alpha)^{1-y}$$

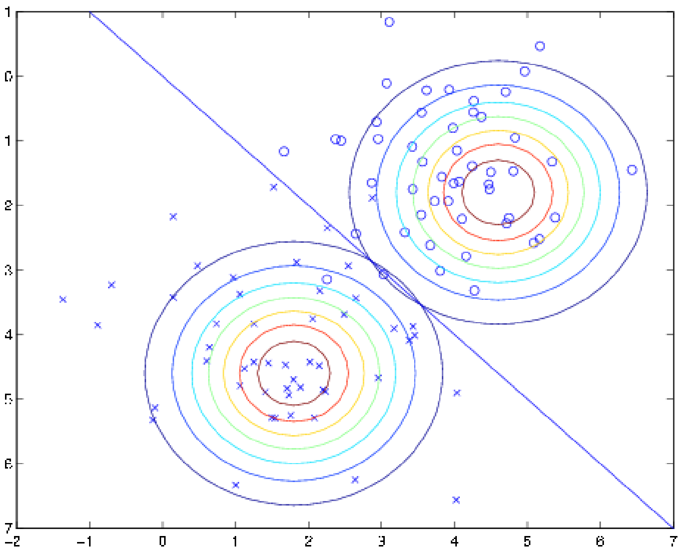
$$p(x \mid y) = \frac{1}{(2\pi)^{n/2} |V|^{1/2}} \exp\Big[-\frac{1}{2} (x - \mu_y) \cdot V^{-1} \cdot (x - \mu_y) \Big]$$

Generative learning: Denote $I_1 = \{i \mid y^i = 1\}$ and I_0 correspondingly. ML estimator for training data $\mathcal{T}^m = \big\{(x^i, y^i) \mid i = 1, \ldots, m\big\}$ gives

$$\alpha^* = \frac{1}{m} |I_1|$$

$$\mu_0^* = \frac{1}{|I_0|} \sum_{i \in I_0} x^i, \quad \mu_1^* = \frac{1}{|I_1|} \sum_{i \in I_1} x^i$$

$$V^* = \frac{1}{m} \sum_{i=1}^{m} (x^i - \mu_{y^i}) \otimes (x^i - \mu_{y^i})$$



1. Generative vs. Discriminative Learning

Discriminative learning(1): Notice that the posterior conditional probabilities can be expressed as

$$p(y \mid x) = \frac{\exp[y(\langle w, x\rangle + b)]}{1 + \exp[\langle w, x\rangle + b]},$$

i.e. a logistic regression, where w and b are some functions of α, μ_0, μ_1 and V.

Estimate w and b by maximising the conditional likelihood on training data

$$(w^*, b^*) \in \arg\max_{w,b} \Big\{ \sum_{i \in I_1} \big(\langle w, x^i\rangle + b\big) - \sum_{i=1}^{m} \log\big(1 + \exp(\langle w, x^i\rangle + b)\big) \Big\}$$

The objective is concave in w and b. Its global optimum can be found by gradient ascent.

Discriminative learning(2): The optimal inference rule is a linear classifier. $\Rightarrow$ Learn it by minimising the empirical risk. $\Rightarrow$ SVM

1. Generative vs. Discriminative Learning

Question: The three methods will provide different decision boundaries when trained on the same dataset. Which one is better?

General answer:

- Generative learning makes stronger assumptions and is more data efficient when the assumptions are (nearly) correct.
- Discriminative learning makes weaker assumptions and is less data efficient but significantly more robust to deviations from model assumptions.

2. Consistency of the Maximum Likelihood estimator

Let $\mathcal{T}^m = \{z^i \mid i = 1, \ldots, m\}$ be i.i.d. generated from $p_{\theta_0}(z)$, with $\theta_0 \in \Theta$ unknown.

Which conditions ensure consistency of the MLE $\theta^* = \arg\max_{\theta \in \Theta} \log p_\theta(\mathcal{T}^m)$?

$$\mathbb{P}_{\theta_0}\big(\|\theta_0 - \theta^*(\mathcal{T}^m)\| > \epsilon\big) \xrightarrow{m \to \infty} 0$$

Denote log-likelihood of training data $L(\theta, \mathcal{T}^m) = \frac{1}{m} \sum_{i=1}^{m} \log p_\theta(z^i)$

and expected log-likelihood $L(\theta) = \mathbb{E}_{\theta_0}\big(L(\theta, \mathcal{T}^m)\big) = \sum_{z \in \mathcal{Z}} p_{\theta_0}(z) \log p_\theta(z)$

Consider $L(\theta, \mathcal{T}^m) = L(\theta) + \big[L(\theta, \mathcal{T}^m) - L(\theta)\big]$

- Suppose, $\theta_0 = \arg\max_{\theta \in \Theta} L(\theta)$ holds, i.e. the model is identifiable,
- The Law of Large Numbers (LLN) tells us

$$\mathbb{P}_{\theta_0}\big(|L(\theta, \mathcal{T}^m) - L(\theta)| > \epsilon\big) \xrightarrow{m \to \infty} 0$$

for each θ and any $\epsilon > 0$.

Question: Is this sufficient to ensure consistency of the MLE?

2. Consistency of the Maximum Likelihood estimator

Identifiability of the model θ_0 is easy to prove if $p_{\theta_0}(z) \not\equiv p_\theta(z)$ holds $\forall \theta \neq \theta_0$.

Let $p(z), q(z)$ be two probability distributions s.t. $p \not\equiv q$. Then

$$\sum_{z \in \mathcal{Z}} p(z) \log p(z) > \sum_{z \in \mathcal{Z}} p(z) \log q(z)$$

follows from strict concavity of the function $\log(x)$:

$$-KL(p \parallel q) = \sum_{z \in \mathcal{Z}} p(z) \log \frac{q(z)}{p(z)} < \log \sum_{z \in \mathcal{Z}} \frac{q(z) p(z)}{p(z)} = \log 1 = 0$$

Further conditions needed to ensure consistency of ML estimators:

- ensure that $L(\theta, \mathcal{T}^m)$ has a global maximum w.r.t. θ for each $\mathcal{T}^m$: e.g. if $\Theta \subset \mathbb{R}^k$ is compact and L is continuous in θ,
- ensure that the Uniform Law of Large Numbers (ULLN) holds, i.e.

$$\mathbb{P}_{\theta_0}\big(\sup_{\theta \in \Theta} |L(\theta, \mathcal{T}^m) - L(\theta)| > \epsilon\big) \xrightarrow{m \to \infty} 0$$

for any $\epsilon > 0$. E.g. if $L(\theta, z)$ can be also upper bounded: $\log p_\theta(z) \leqslant d(z)$ $\forall \theta$ with $\mathbb{E}_{\theta_0} d(z) < \infty$.

3. 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 $\mathcal{T}^m = \{x^i \in \mathcal{X} \mid i = 1, 2, \ldots, m\}$ i.i.d. generated from p_{θ_0}.

How shall we implement the MLE

$$\theta^*(\mathcal{T}^m) = \arg\max_{\theta \in \Theta} \frac{1}{m} \sum_{x \in \mathcal{T}^m} \log p_\theta(x) = \arg\max_{\theta \in \Theta} \frac{1}{m} \sum_{x \in \mathcal{T}^m} \log \sum_{y \in \mathcal{Y}} p_\theta(x, y)$$

- 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)

3. The Expectation Maximisation Algorithm

Because the original derivation of the algorithm is somewhat involved, we follow here an alternative approach by Minka (1998):

- 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:

$$L(\theta, \mathcal{T}^m) = \frac{1}{m} \sum_{x \in \mathcal{T}^m} \log \sum_{y \in \mathcal{Y}} p_\theta(x, y) = \frac{1}{m} \sum_{x \in \mathcal{T}^m} \log \sum_{y \in \mathcal{Y}} \frac{\alpha_x(y)}{\alpha_x(y)} p_\theta(x, y) \geqslant$$

$$L_B(\theta, \alpha, \mathcal{T}^m) = \frac{1}{m} \sum_{x \in \mathcal{T}^m} \sum_{y \in \mathcal{Y}} \alpha_x(y) \log p_\theta(x, y) - \frac{1}{m} \sum_{x \in \mathcal{T}^m} \sum_{y \in \mathcal{Y}} \alpha_x(y) \log \alpha_x(y)$$

3. 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.

$$\theta^{(t+1)} = \arg\max_{\theta \in \Theta} \frac{1}{m} \sum_{x \in \mathcal{T}^m} \sum_{y \in \mathcal{Y}} \alpha_x^{(t)}(y) \log p_\theta(x, y)$$

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

Claims:

- The bound is tight if $\alpha_x(y) = p_\theta(y \mid x)$,
- 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).

3. The Expectation Maximisation Algorithm

Example: A Naive Bayes model for string patterns

- $x = (x_1, \ldots, x_n)$ strings of length n over a finite alphabet $\mathcal{B}$,
- $k = 0, 1$ string pattern class,
- joint distribution - Naive Bayes model

$$p(x, k) = p(k) \prod_{j=1}^{n} p(x_j \mid k)$$

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

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

3. The Expectation Maximisation Algorithm

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

$$\alpha_x^{(t)}(k) = p^{(t)}(k \mid x) = \frac{p^{(t)}(k) \prod_{j=1}^n p^{(t)}(x_j \mid k)}{\sum_{k'} p^{(t)}(k') \prod_{j=1}^n p^{(t)}(x_j \mid k')}.$$

M-step Re-estimate the model by

$$p^{(t+1)}(k) = \frac{1}{m} \sum_{x \in \mathcal{T}^m} \alpha_x^{(t)}(k)$$

$$p^{(t+1)}(x_j = b \mid k) = \frac{\sum_{x \in \mathcal{T}^m :\, x_j = b} \alpha_x^{(t)}(k)}{\sum_{x \in \mathcal{T}^m} \alpha_x^{(t)}(k)}$$