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

Generative vs. Discriminative Learning

Generative learning:

- Model the **joint** probability distribution $p(x, y)$ for features $x \in \mathcal{X}$ and hidden states $y \in \mathcal{Y}$
- Inference rule:

$$h(x) \in \operatorname*{arg\,max}_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} p(y' \mid x) \ell(y', y)$$

- Learning: if the distribution $p_\theta(x, y)$ is known up to parameters $\theta \in \Theta$ only $\Rightarrow$ estimate θ^* from training data $\mathcal{T}^m = \{(x^i, y^i) \in \mathcal{X} \times \mathcal{Y} \mid i = 1, \ldots, m\}$

Discriminative learning(1):

- Model only the **conditional** distributions $p(y \mid x)$
- Learning: if they are known up to parameters $\theta \in \Theta$ only $\Rightarrow$ estimate θ^* by maximising the conditional likelihood on the training data

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

- Inference rule: as above

Generative vs. Discriminative Learning

Discriminative learning(2):

- Model the class of inference rules $h \in \mathcal{H}$ directly.
- Estimate the best inference rule h^* by minimising the empirical risk on the training data

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

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

$y = 0, 1$, $y \sim Bernoully(\alpha)$ and $x \in \mathbb{R}^n$, $x|y = 0 \sim \mathcal{N}(\mu_0, \Sigma)$, $x|y = 1 \sim \mathcal{N}(\mu_1, \Sigma)$. The distributions are

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

$$p(x \mid y = 0) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2} (x - \mu_0) \cdot \Sigma^{-1} \cdot (x - \mu_0) \right]$$

$$p(x \mid y = 1) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2} (x - \mu_1) \cdot \Sigma^{-1} \cdot (x - \mu_1) \right]$$

Generative vs. Discriminative Learning

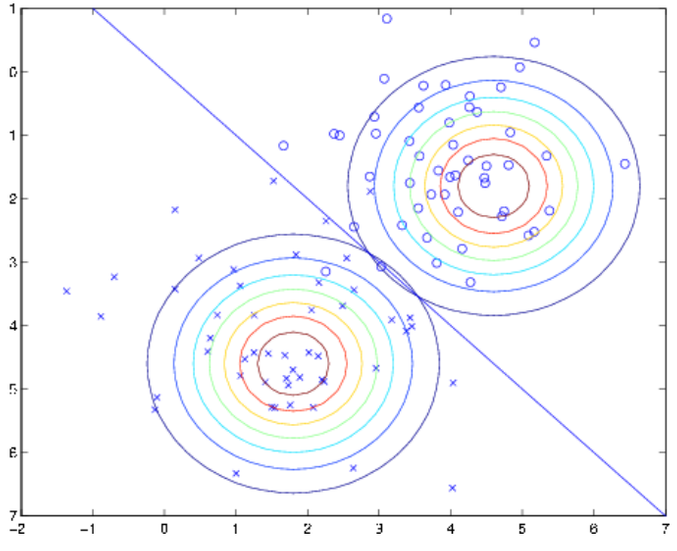
Generative learning: Maximum likelihood estimator for training data $\mathcal{T}^m = \{(x^i, y^i) \mid i = 1, \ldots, m\}$ gives

$$\alpha^* = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}\{y^i = 1\}$$

$$\mu_0^* = \frac{\sum_{i=1}^{m} \mathbf{1}\{y^i = 0\} x^i}{\sum_{i=1}^{m} \mathbf{1}\{y^i = 0\}}$$

$$\mu_1^* = \frac{\sum_{i=1}^{m} \mathbf{1}\{y^i = 1\} x^i}{\sum_{i=1}^{m} \mathbf{1}\{y^i = 1\}}$$

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



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. logistic regression, where w and b are some functions of α, μ_0, μ_1 and Σ.

Generative vs. Discriminative Learning

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

$$(w^*, b^*) \in \arg\max_{w,b} \left\{ \sum_{i=1}^{m} \mathbf{1}\{y^i = 1\}(\langle w, x^i \rangle + b) - \sum_{i=1}^{m} \log\left(1 + \exp(\langle w, x^i \rangle + b)\right) \right\}$$

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

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 significantly more robust to deviations from model assumptions.

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

- $\theta_0 = \arg\max_{\theta \in \Theta} L(\theta)$ holds, i.e. the model is identifiable, and
- 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 $\epsilon > 0$.

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 $\log(x)$:

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

Sufficient conditions that ensure the **ULLN** e.g.:

- $\Theta \subset \mathbb{R}^k$ is compact, $L(\theta, \mathcal{T}^m)$ is continuous in θ and there is a function $d(z) \geqslant \log p_\theta(z)$ $\forall \theta$ with $\mathbb{E}_{\theta_0}(d(z)) < \infty$,
- $\log p_\theta(z)$ is a concave function of θ, $\Theta \subset \mathbb{R}^k$ is convex and $\theta_0 \in \operatorname{int} \Theta$.

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 $\mathcal{T}^m = \left\{ x^i \in \mathcal{X} \,\middle|\, i = 1,2,\ldots,m \right\}$ i.i.d. generated from p_{θ_0}.

How shall we implement the MLE

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

- 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)

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 $\alpha_i(y) \geqslant 0$, for each $x^i \in \mathcal{T}^m$, s.t. $\sum_{y \in \mathcal{Y}} \alpha_i(y) = 1$
- 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:

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

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

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

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

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

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

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

Claims:

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

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 $\mathcal{T}^m = \left\{ x^i \in \mathcal{B}^n \;\middle|\; i = 1, 2, \ldots, m \right\}$, 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.

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 $\mathcal{T}^m$

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

M-step Re-estimate the model by

$$p^{(t+1)}(k) = \frac{1}{m} \sum_{i=1}^{m} \alpha_i^{(t)}(k)$$

$$p^{(t+1)}(x_j = b \mid k) = \frac{\sum_{i:\, x_j^i = b} \alpha_i^{(t)}(k)}{\sum_{i=1}^{m} \alpha_i^{(t)}(k)}$$