Exercises
Advanced Machine Learning
Fall 2019

Machine Learning Laboratory
Dept. of Computer Science, ETH Zürich
Prof. Joachim M. Buhmann
Web https://ml2.inf.ethz.ch/courses/aml/

Series Sep 19th 2019
(Recap IML and important concepts)

Teaching assistant: **Carlos Cotrini**
ccarlos@inf.ethz.ch

Solution 1 (Regression):

1. **Linear Regression**

   (a) The residual sum of squares error $\mathcal{E}_{RSS}(\beta)$ is defined as:

   $$\mathcal{E}_{RSS}(\beta) = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \mathbf{x}_i^\top \beta)^2 = \|\mathbf{y} - \mathbf{X}\beta\|_2^2 = (\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta)$$

   (b) The vector $\hat{\beta}$ that minimizes the error can be found using the derivative with respect to β:

   $$\frac{d}{d\beta}\mathcal{E}_{RSS}(\beta) = \frac{d}{d\beta}(\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta) = \frac{d}{d\beta}(\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\beta + \beta^\top\mathbf{X}^\top\mathbf{X}\beta) = -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\beta \stackrel{!}{=} 0$$

   The optimizer $\hat{\beta}$ (for non singular $\mathbf{X}^\top\mathbf{X}$) thus becomes $\hat{\beta} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$.

2. **Ridge Regression**

   (a) The term $\lambda\beta^\top\beta$ is called the regularizer. It imposes a cost on large values of β (i.e. on a more complex model) and therefore drives the solution towards a more general one. This can be used to combat overfitting on the training data and improve the prediction error on out-of-training data. λ is a design parameter that weighs both terms against each other. For $\lambda = 0$, Ridge regression becomes equivalent to linear regression (potentially overfitting on the training data, inducing large variance) and for $\lambda \to \infty$ the optimizer is $\hat{\beta} = \mathbf{0}$ (The "most general" solution, which is blind to the training data, inducing large bias). An optimal value for λ in between can be found using model selection techniques, e.g. cross validation.

   (b) The optimizer can be derived similar to the linear case:

   $$\frac{d}{d\beta}\mathcal{E}_{Ridge}(\beta, \lambda) = \frac{d}{d\beta}\left((\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^\top\beta\right) =$$

   $$= \frac{d}{d\beta}(\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\beta + \beta^\top\mathbf{X}^\top\mathbf{X}\beta + \lambda\beta^\top\beta) = -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\beta + 2\lambda\mathbb{I}\beta \stackrel{!}{=} 0$$

   $$\implies \hat{\beta}_{Ridge} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbb{I})^{-1}\mathbf{X}^\top\mathbf{y}$$

Solution 2 (Comprehension Questions):

1. **Overfitting**

   (a) i. The model is underfitting. It can be seen that the model stops improving quickly at a comparably high loss, indicating that e.g. the network is not complex enough to accurately model the data.

   ii. The model is overfitting. Notice how the trainig accuracy drops to almost zero, meaning that the model fits the training data extremely well. However, the validation loss starts to increase at some point, indicating the onset of overfitting.

   iii. The third model is reasonable. The training and validation loss both approach low values.

   (b) i. To prevent underfitting, one can increase the model complexity (e.g. add more nodes, more layers, ...).

   ii. To combat overfitting, one could add a regularizing term (e.g. as before in Ridge regression) or use other approaches such as early stopping (stopping training when the validation loss starts to increase again).

   iii. This network already performs reasonably well. It might nevertheless be worthwile to explore other approaches or models for the data.

2. **Cross Validation**

   (a) The training data is used to fit the model. In our case this corresponds to determining the optimal β vector.
   The validation set is used to select the optimal hyperparameters, in this case the regularization strength λ. It is important, to determine the regularization on a set that the model has not yet seen, since it's supposed to prevent overfitting. (Imagine computing both values on the same set, an optimal solution with $\mathcal{E} = 0$ could be achieved by setting $\lambda = 0$ and then massively overfitting the training data. How well would such a model work on unseen data?)
   The test set is used to get an 'unbiased' estimate of the generalization error of our previously optimized model. The separation of the sets is important for the same reasons as before. We are interested in how well our model reflects the information (signal) in the data and not in its adaptivity to fluctuations during training (noise). This can only be done on data that the model has not already been adapted to.

   (b) The cross validation algorithm can be summarized as follows:

Algorithm 1 K-Fold Cross Validation

initialize: Split data $\mathcal{X}$ into K equally sized subsets:

$$\mathcal{X} = \bigcup_{i=1}^{K} \mathcal{X}_i \qquad \text{s.t.} \quad \forall \mu \neq \nu \quad \mathcal{X}_\mu \cap \mathcal{X}_\nu = 0$$

for $i = 1, \ldots, K$ **do**
  Train model on data $\mathcal{X}_{-i} = \mathcal{X} \backslash \mathcal{X}_i$
  Estimate its prediction error $\mathcal{E}_i$ on $\mathcal{X}_i$
end for
Combine all estimates:

$$\mathcal{E}_{CV} = \frac{1}{K} \sum_{i=1}^{K} \mathcal{E}_i$$

The validation set itself is random and optimizing for a single validation set creates bias. By using multiple validation sets and averaging, the variance of the error estimate can be decreased. Furthermore, we make best use of the available data (Multiple independent sets are typically expensive and wasteful).

(c) There is a systematic tendency to underfit. Because we only use $\frac{K-1}{K}$ percent of the available data, the models are not as complex as they could be using the full data. This behavior is affected by the number of splits K. While for $K = 1$ we don't do cross validation at all, the special case of $K = n$ is known as Leave-One-Out-Cross-Validation (LOOCV). This method is unbiased w.r.t. to the true prediction error but may introduce more variance (due to highly correlated training sets) and is computationally more expensive. An Engineer's rule of thumb is to choose K as $\min\{\sqrt{n}, 10\}$.

3. **Generative vs. Discriminative Modeling**

(a) In the discriminative setting, one tries to model the conditional probability $P(y|X)$, whereas in a generative model, one tries to estimate the joint probability $P(y, X)$. This is a strictly more ambitious goal, since typically estimating $P(X)$ is hard. However, once $P(y, X)$ is known, one can derive $P(y|X) = \frac{P(y,X)}{P(X)} = \frac{P(y,X)}{\sum_y P(y,X)}$ using Bayes' rule (but not vice versa).

(b) A typical discriminative model shapes its decision boundaries such that they best separate the training data. It can therefore not detect outliers (that happen to be on the other side of the boundary). In the generative case, each class is modeled separately. The decision boundaries then are the equiprobability curves, i.e. the region where $P(y_1|X) = P(y_2|X)$, which is equal to $P(X|y_1)P(y_1) = P(X|y_2)P(y_2)$. They can detect outliers if the model is sufficiently accurate.

(c) If the model is well specified (you managed to estimate $P(X)$ accurately), generative models usually perform better. However, this is rarely achieved. In the cases where $P(X)$ is hard to estimate, discriminative models are typically more robust (especially with lots of data available).
Because discriminative models are supervised learners, they can't make use of unlabeled data. Generative models on the other lend themselves quite naturally to the use of unlabeled data (e.g. clustering using Gaussian mixtures).

Solution 3 (EM-Algorithm for Gaussian Mixtures):

1. The probability of datum x_i is a linear combination of all k Gaussians,

$$P(x_i|\theta) = \sum_{c=1}^{k} \pi_c P(x_i|c, \mu_c, \Sigma_c)$$

where the prior probabilities π_c corresponds to the weights. Notice that for $P(x_i|\theta)$ to be a valid pdf $\sum_{c=1}^{k} \pi_c = 1$. $P(x_i|c, \mu_c, \Sigma_c)$ is the Gaussian pdf of component c,

$$P(x_i|c, \mu_c, \Sigma_c) = \frac{1}{(2\pi)^{d/2} |\Sigma_c|^{1/2}} \exp\left(-\frac{1}{2}(x_i - \mu_c)^\top \Sigma_c^{-1}(x_i - \mu_c)\right).$$

2. The likelihood of the data set is the product of all its elements,

$$P(\mathcal{X}|\theta) = \prod_{i=1}^{n} P(x_i|\theta) = \prod_{i=1}^{n} \sum_{c=1}^{k} \pi_c P(x_i|c, \mu_c, \Sigma_c)$$

The log likelihood then becomes

$$L(\mathcal{X}|\theta) = \log(P(\mathcal{X}|\theta)) = \sum_{i=1}^{n} \log \sum_{c=1}^{k} \pi_c P(x_i|c, \mu_c, \Sigma_c)$$

3. The assignment variable M_{ic} tells us, which component generated datum x_i. The joint likelihood can thus be represented as

$$P(\mathcal{X}, M|\theta) = \prod_{i=1}^{n}\prod_{c=1}^{k}(\pi_c P(x_i|c,\mu_c,\Sigma_c))^{M_{ic}}$$

Notice that the term $(\pi_c P(x_i|c,\mu_c,\Sigma_c))^{M_{ic}}$ becomes 1 if $M_{ic}=0$ and does not contribute to the product. For $M_{ic}=1$ it remains unchanged. The log likelihood is computed as

$$L(\mathcal{X}, M|\theta) = \log(P(\mathcal{X}, M|\theta)) = \sum_{i=1}^{n}\sum_{c=1}^{k} M_{ic}\log(\pi_c P(x_i|c,\mu_c,\Sigma_c))$$

4. Using the definition of γ_{ic} and the definition of the expected value, we get

$$\gamma_{ic} = \mathbb{E}_{M|\mathcal{X},\theta^o}[M_{ic}] = P(M_{ic}=1|x_i,\theta^o)\cdot 1 + P(M_{ic}=0|x_i,\theta^o)\cdot 0 = P(M_{ic}=1|x_i,\theta^o) = P(c|x_i,\theta^o)$$

This can be brought into the desired form using Bayes' rule:

$$P(c|x_i,\theta^o) = \frac{P(c|\theta^o)P(x_i|c,\theta^o)}{P(x_i|\theta^o)} = \frac{\pi_c^o P(x_i|c,\mu_c^o,\Sigma_c^o)}{\sum_{\nu=1}^{k}\pi_\nu^o P(x_i|\nu,\mu_\nu^o,\Sigma_v^o)} = \gamma_{ic}$$

5. (a) We can calculate $Q(\theta)$ as follows, where we use the linearity of expectation in $(*)$ and the fact that $\log(\pi_c P(x_i|c,\mu_c,\Sigma_c)$ does not depend on M in $(**)$.

$$Q(\theta) = \mathbb{E}_{M|\mathcal{X},\theta^o}[L(\mathcal{X},M|\theta)] = \mathbb{E}_{M|\mathcal{X},\theta^o}[\sum_{i=1}^{n}\sum_{c=1}^{k} M_{ic}\log(\pi_c P(x_i|c,\mu_c,\Sigma_c))] \overset{*}{=}$$

$$= \sum_{i=1}^{n}\sum_{c=1}^{k}\mathbb{E}_{M|\mathcal{X},\theta^o}[M_{ic}\log(\pi_c P(x_i|c,\mu_c,\Sigma_c))] \overset{**}{=} \sum_{i=1}^{n}\sum_{c=1}^{k}\underbrace{\mathbb{E}_{M|\mathcal{X},\theta^o}[M_{ic}]}_{:=\gamma_{ic}}\log(\pi_c P(x_i|c,\mu_c,\Sigma_c)) =$$

$$= \sum_{i=1}^{n}\sum_{c=1}^{k}\gamma_{ic}\log(\pi_c P(x_i|c,\mu_c,\Sigma_c))$$

(b) The optimizer $\mu_c \in \arg\max_{\mu_c} Q(\theta)$ can be calculated by setting the derivative to zero:

$$\frac{\partial}{\partial\mu_c}Q(\theta) = \frac{\partial}{\partial\mu_c}\sum_{i=1}^{n}\sum_{c=1}^{k}\gamma_{ic}\log(\pi_c P(x_i|c,\mu_c,\Sigma_c)) \overset{*}{=} \sum_{i=1}^{n}\frac{\partial}{\partial\mu_c}\left[\gamma_{ic}\log(\pi_c P(x_i|c,\mu_c,\Sigma_c))\right] =$$

$$= \sum_{i=1}^{n}\frac{\partial}{\partial\mu_c}\left[\gamma_{ic}\log\left(\frac{1}{(2\pi)^{d/2}|\Sigma_c|^{1/2}}\exp\left(-\frac{1}{2}(x_i-\mu_c)^\top\Sigma_c^{-1}(x_i-\mu_c)\right)\right)\right] =$$

$$\sum_{i=1}^{n}\frac{\partial}{\partial\mu_c}\left[\gamma_{ic}\left(\text{const}-\frac{1}{2}(x_i-\mu_c)^\top\Sigma_c^{-1}(x_i-\mu_c)\right)\right] =$$

$$= \sum_{i=1}^{n}\gamma_{ic}\Sigma_c^{-1}(x_i-\mu_c) \overset{!}{=} 0$$

$$\implies \quad \sum_{i=1}^{n}\gamma_{ic}(x_i-\mu_c) = 0 \quad \implies \quad \sum_{i=1}^{n}\gamma_{ic}x_i = \sum_{i=1}^{n}\gamma_{ic}\mu_c \quad \implies \quad \mu_c = \frac{\sum_{i=1}^{n}\gamma_{ic}x_i}{\sum_{i=1}^{n}\gamma_{ic}}.$$

In $(*)$, the second sum disappears, because only the c^{th} term depends on μ_c.

(c) The update equation for Σ_c can be calculated similar to the one above.

$$\frac{\partial}{\partial \Sigma_c^{-1}} Q(\theta) = \sum_{i=1}^{n} \frac{\partial}{\partial \Sigma_c^{-1}} \left[\gamma_{ic} \log \left(\pi_c \frac{1}{(2\pi)^{d/2} |\Sigma_c|^{1/2}} \exp \left(-\frac{1}{2} (x_i - \mu_c)^\top \Sigma_c^{-1} (x_i - \mu_c) \right) \right) \right] =$$

$$= \sum_{i=1}^{n} \frac{\partial}{\partial \Sigma_c^{-1}} \left[\gamma_{ic} \left(\text{const} - \frac{1}{2} \log |\Sigma_c| - \frac{1}{2} (x_i - \mu_c)^\top \Sigma_c^{-1} (x_i - \mu_c) \right) \right] =$$

$$\sum_{i=1}^{n} \gamma_{ic} \left(\frac{1}{2} \Sigma_c - \frac{1}{2} (x_i - \mu_c)(x_i - \mu_c)^\top \right) \stackrel{!}{=} 0$$

$$\implies \quad \sum_{i=1}^{n} \gamma_{ic} \Sigma_c - \sum_{i=1}^{n} \gamma_{ic} (x_i - \mu_c)(x_i - \mu_c)^\top = 0 \quad \implies \quad \Sigma_c = \frac{\sum_{i=1}^{n} \gamma_{ic} (x_i - \mu_c)(x_i - \mu_c)^\top}{\sum_{i=1}^{n} \gamma_{ic}}.$$

(d) Using the information from the hint, we state the Lagrangian as

$$\mathcal{L} = Q(\theta) + \lambda \left(\sum_{c=1}^{k} \pi_c - 1 \right).$$

Taking the derivative w.r.t. π_c yields

$$\frac{d}{d\pi_c} \mathcal{L} = \frac{\partial}{\partial \pi_c} Q(\theta) + \lambda = \sum_{i=1}^{n} \gamma_{ic} \frac{\partial}{\partial \pi_c} \left[\log(\pi_c) + \log(P(x_i | c, \mu_c, \Sigma_c)) \right] + \lambda = \sum_{i=1}^{n} \gamma_{ic} \frac{1}{\pi_c} + \lambda \stackrel{!}{=} 0$$

$$\implies \quad \sum_{i=1}^{n} \gamma_{ic} = -\lambda \pi_c \quad \implies \quad \sum_{i=1}^{n} \underbrace{\sum_{c=1}^{k} \gamma_{ic}}_{=1} = -\lambda \underbrace{\sum_{c=1}^{k} \pi_c}_{=1} \quad \implies \quad \lambda = -n$$

Substituting λ back in we get

$$\pi_c = \frac{1}{n} \sum_{i=1}^{n} \gamma_{ic}$$

If we take a look at the results, we notice that they are quite intuitive. In the E-step, the new assignment probability of datum x_i to component c corresponds to the fraction c has on the total probability of x_i. In the M-step, the optimal μ_c and Σ_c turn out to be calculated as the usual (MLE) mean and variance weighted with the assignment probabilities, while the prior for component c is its average assignment probability.