GENERALISATION BOUNDS FOR GENERATIVE/DISCRIMINATIVE LEARNING II

DEEP LEARNING (SS2020)
2. COMPUTER LAB (10P)

1. Introduction

This is the second of two home works aiming at an experimental comparison of generalisation bounds for generative and discriminative learning. We will consider a simple Gaussian classification problem and two learning approaches – generative learning by maximum likelihood estimate and discriminative learning by a feed forward network.

2. Model & Notations

As before, we consider a Gaussian classification problem with features $\boldsymbol{X} \in \mathbb{R}^d$ and two classes $Y \in \{0, 1\}$. The joint distribution over features and classes is defined by the prior class probabilities and the distributions for the features conditioned on the classes, which are assumed as multivariate normal distributions.

$$\begin{gathered}\boldsymbol{X} \mid Y \sim \mathcal{N}(\boldsymbol{\mu}_Y, \boldsymbol{C}_Y) \\ \mathbb{P}(Y=0) = \pi_0 \,, \mathbb{P}(Y=1) = \pi_1 .\end{gathered} \tag{1}$$

In contrast to the first homework, we will assume that the covariance matrices of the two distributions are different, i.e. $\boldsymbol{C}_0 \neq \boldsymbol{C}_1$. The optimal classifier is the Bayes classifier $h^*(x) = \mathrm{H}\big(\frac{\pi_1 \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{C}_1)}{\pi_0 \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{C}_0)} - 1\big)$, which is a quadratic classifier

$$h^*(\boldsymbol{x}) = \mathrm{H}\big[d_1(\boldsymbol{x}) - d_0(\boldsymbol{x}) + a_1 - a_0\big] \tag{2}$$

with $d_i(\boldsymbol{x})$ and a_i defined by

$$d_1(\boldsymbol{x}) = -(\boldsymbol{x} - \mu_1)^T \boldsymbol{C}_1^{-1} (\boldsymbol{x} - \mu_1) \qquad a_1 = -\log\det \boldsymbol{C}_1 + 2\log \pi_1 \tag{3}$$

$$d_0(\boldsymbol{x}) = -(\boldsymbol{x} - \mu_0)^T \boldsymbol{C}_0^{-1} (\boldsymbol{x} - \mu_0) \qquad a_0 = -\log\det \boldsymbol{C}_0 + 2\log \pi_0 \tag{4}$$

We will use standard 0/1 loss. There is no closed form expression for the risk of this classifier. We will estimate it empirically on a sufficiently large test set. A classifier is trained using i.i.d training sets $\mathcal{T}^m = \{(\boldsymbol{x}^j, y^j) \mid j = 1, \dots, m\}$ generated from the model.

3. Generative learning

Generative learning is straightforward here. Given training data $\mathcal{T}^m$, we estimate the model parameters $\boldsymbol{\mu}_0$, $\boldsymbol{\mu}_1$, the covariance matrices $\boldsymbol{C}_0$, $\boldsymbol{C}_1$ and the prior class probabilities $\pi_{0/1}$. Given the estimate, the predictor h_m is obtained from (2).

4. Discriminative learning

We will use a neural network with one hidden layer with ReLU activations and one output neuron with sigmoid activation interpreted as (conditional) probability for class 1. The network will be learned by vanilla gradient descent with constant step width using the negative log class probability as loss.

5. Assignments

Use the provided model and source templates for your experiments. The package contains:

- `tools.py` — tools for working with the ground truth model: loading, generating, plotting the decision boundary of a classifier versus the optimal classifier.
- `model.pkl` — python "pickle" file with the `dict` defining the true model with keys `['clps', 'mues', 'covs', 'dim']`. The mues are stored as an array $2 \times$ `dim` and the covariance matrices are stored as an array $2 \times$ `dim` $\times$ `dim`. Since `dim`=2 in the provided data, this may be confusing.
- `template.py` — An example that loads the model, defines and trains some simple network. This is to provide you an implementation template and to illustrate the tools, in particular visualization of a classifier. You are free to follow the pattern or to propose you own.

Assignment 1. (4p)
Implement a neural network with `input_size` inputs, one hidden layer with `hidden_size` units and ReLU activations and, finally, the logistic regression model in the last layer, i.e. a linear transform and sigmoid. For training, the network should compute the average loss (negative log likelihood) for the whole training set (no mini-batches). Initialize all network parameters randomly, e.g. uniformly in $[-1, 1]$.

When using log conditional likelihood as the learning objective, it is numerically more stable to combine the logarithm of the sigmoid function into one function. If $a \in \mathbb{R}$ denotes the score, i.e. the output of the last linear layer, we get

$$\log p(y = 1 \mid a) = \log \mathrm{S}(a) = \log(1 + e^{-a}) \tag{5}$$

$$\log p(y = 0 \mid a) = \log(1 - \mathrm{S}(a)) = \log \mathrm{S}(-a) = \log(1 + e^{a}). \tag{6}$$

Implement the computation of the gradient of the loss in all parameters by backpropagation. Test correctness of your implementation by the following procedure. Consider varying a parameter vector $w \in \mathbb{R}^n$ (the model has several parameter vectors, considering one at a time will help to isolate errors). Keeping all other parameters fixed, compute

$$\delta = \frac{L(w+\varepsilon) - L(w-\varepsilon)}{2} \tag{7}$$

for some small random $\varepsilon \in \mathbb{R}^n$. This should match the scalar product $\langle \nabla_w L, \varepsilon \rangle$. More precisely, if L is differentiable in w in some neighbourhood of w, we expect

$$\delta - \langle \nabla_w L, \varepsilon \rangle = o(\|\varepsilon\|). \tag{8}$$

Assignment 2. (3p)
Implement gradient descent with constant step size to train the network. Verify that the loss improves during the training. Note that the expected loss of a predictor that outputs equal predictive probabilities (e.g. at a random initialization) is around $\log 2 \approx 1.4$. So the training should start from around that value and decrease it to about 0.1 or lower. Monitor also the training error of the classifier. In should be decreasing during the training progress. Plot the resulting classifier with the help of `G2Model.plot_predictor` to see that it has fitted the data well. Hint: `lr` $= 0.005$ and `epochs` $= 5000$ (the number of GD iterations on complete data) work well.

Assignment 3. (3p)
Fix a test set size n sufficiently large to guarantee a confidence interval $|R(h) - R_{\mathcal{T}^n}(h)| < 0.01$ with probability 95%. Generate a test set of this size n and use it for all subsequent experiments. Report the estimated risk R_B of the optimal Bayesian classifier.

Estimate and compare test risks of the generative model and the neural network with `hidden_size` $= 5, 10, 100$ (i.e. 4 models in total). Repeat the training for each of the four models $T = 100$ times (trials). Sample a new training set for each trial and use it for learning all 4 models. Compute the average excess risk. Our goal is to compare which learning method achieves a better excess risk in the expectation over the training set. Let the risk in the trial i be denoted by X_i, then the sample mean

$$X = \frac{1}{T}\sum_i X_i, \tag{9}$$

is a random estimate of the expected risk. Find the 95% confidence interval on this average using the following steps. Note that X_i is a bounded random variable in $[0,1]$. Let $\delta = 0.05$, then we know from Hoeffding's inequality that if

$$\delta = 2e^{-2T\varepsilon^2}, \tag{10}$$

then X is within ε of the true expected value $\mathbb{E}[X]$ with probability at last $1-\delta$. For given values of T and δ we can find ε. The confidence interval is then given by $[X-\varepsilon, X+\varepsilon]$.

Optionally, observe that the Chebyshev inequality also applies. Estimate the variance v of X_i over the trials and find ε by solving

$$\delta = \frac{v}{T\varepsilon^2}. \tag{11}$$

Since both approaches imply guaranteed confidence intervals, we can pick the one which is tighter.

Compute the above estimates of the mean values and respective confidence intervals for the four models and training set sizes $m = 50, 100, 200$. For each model plot its estimated excess risk with confidence intervals versus the training set size. Display these results in the same axis. For a pair of models, if their confidence intervals do not overlap, we can claim with high probability that one is giving a better expected excess risk than the other. If they do overlap significantly, we cannot say which model is better without making more trials.

Discussion. We encourage you to think about and discuss with us in your report the following questions. This is informal and not scored.

(1) What are your observations about generative versus discriminative learning approaches?
(2) Why do the networks not overfit despite they have large capacity? Why they generalize even better when the capacity is increased?
(3) Why the classification boundary stays smooth for bigger hidden layer size?
(4) Would the networks start to overfit if we solved the empirical risk minimization problem better, e.g. used a better optimizer?
(5) How the random initialization helps training and generalization? What other factors may influence it?