Introduction to Machine Learning (67577)

Exercise 3
Classification

May 2018

Contents

Submission guidelines

- submission is due 16.5.2019 (Take in mind that in 9.5 you will get a new exercise).
- The assignment is to be submitted via Moodle only.
- Files should be zipped to **ex_3_First_Last.zip** (First is your first name, Last is your last name. In English, of course).
- The .zip file size should not exceed 10Mb.
- Each question is enumerated, use these numbers when answering your questions.
- Starting grade : 5

Linear Regression and the Likelihood function

In a strange turn of events, we defined the Logistic Regression algorithm using the likelihood instead of our old squared loss, as we did in the Linear Regression algorithm. As one can understand, the squared loss sounds like a good performance measure from a geometric point of view. But is there a probabilistic point of view that would lead us to use the same optimization problem as in the Linear Regression algorithm?

1. Let $\sigma \in \mathbb{R}$, s.t. $\sigma > 0$. Consider the case in which the distribution of $Y|X = x$ is $\mathcal{N}(h(x), \sigma^2)$ for any $x \in \mathbb{R}$. In this case the likelihood function is:

$$l_{likelihood}(h, (x, y)) \quad = \quad \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(y - h(x))^2}{2\sigma^2}\right)$$

   (a) (5 Pts.) As we consider a regression algorithm that is based on the likelihood model, follow the Performance Measure bullet in recitation 6 and write the objective of our new algorithm (Based on m iid samples).

   (b) (5 Pts.) Show that maximizing the likelihood objective, which you deduced in (a), is equivalent to minimizing the objective of the linear regression.

2. (Optional) Now assume that $y|x \sim$: $Laplace(h(x), \sigma)$. (Link)

   (a) Repeat 1a under this assumption.

   (b) Write the objective as a minimization problem (min_h $(1/m) \sum_{i=1}^{m} l_?(h, (x, y))$ where $l_?$ is some loss you need to define).

ROC Curve and Random Classification

3. Assume we have a binary classification problem (i.e. $\mathcal{Y} = \{1, 0\}$) and for any $t \in [0, 1]$ we define the following classifier h_t:

   - Draw a number z from a uniform distribution on $[0, 1]$.
   - If $z \geqslant t$ then return $h_t(\mathbf{x}) = 1$, otherwise return $h_t(\mathbf{x}) = 0$.

Minor comment: note that the classifier is totally oblivious to $\mathbf{x}$ and it is non-deterministic in the sense that it may return different answers for the same $\mathbf{x}$. While this deviates from our definition of a hypothesis, we can slightly adjust our formalism to consider this type of hypotheses, as done above.

(a) (5 Pts.) Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$. Denote the conditional probability of the classifier's output given the label by $\mathbb{P}_{\mathcal{D}}(h_t(\mathbf{x}) \mid y)$. Note that this distribution takes into account both the randomness in $\mathcal{D}$ and the classifier.
Under this notation, the true-positive rate of the classifier is $\text{TPR}(t) = \mathbb{P}_{\mathcal{D}}(h_t(\mathbf{x}) = 1 \mid y = 1)$. What does t need to be in order to obtain $\text{TPR}(t) = p$? (use p in your answer)

(b) (5 Pts.) The false-positive rate of the classifier is $\text{FPR}(t) = \mathbb{P}_{\mathcal{D}}(h_t(\mathbf{x}) = 1 \mid y = 0)$. When $\text{TPR}(t) = p$, what is $\text{FPR}(t)$? Recall that the ROC-curve is the line that we get if we plot the true-positive rate vs. the false-positive rate. How does the ROC-curve look like for our classifier?

Another Minor comment: Under the notations we used in the tirgul (see the tirgul notes), our hypothesis h is the hypothesis that returns z and then h_t classifies positively if h returned a number larger than t.

4. Monotonicity of the ROC curve

(a) (optional) Give an example in which - there exists a region in $[0, 1]$ that when you **decrease** the threshold in it only the FPR increases while the TPR stays the same, and another one for a case in which only the TPR increases.

Remark 1 *In a case where for a specific value of FPR there exists multiple values of TPR, the ROC uses the supremum of these TPR values.*

(b) (5 Pts.) Prove that the ROC curve is a monotonic non-decreasing function of the FPR.

Remark 2 *We defined the ROC through the threshold. You may show this claim by showing the effect of raising the threshold by $\epsilon > 0$ on the TPR and FPR.*

5. All together now- LDA, ROC and the logit transformation- (30 pts)

(a) (5 Pts.) Assume an LDA model of the data, $\mathcal{X} = \mathbb{R}^d$, with two Gaussians, meaning that $\mathcal{Y} = \{0, 1\}$. Write the Bayes optimal classifier $h_{\mathcal{D}}$ as an affine classifier using some transformation ($h_{\mathcal{D}} = sign(\langle (x; 1), w \rangle)$.
* $(x; 1)$ is a $d+1-dimensional$ vector that is a concatenation of the $d-dimensional$ vector x and the scalar 1 in the $d + 1$ coordinate.

(b) Chen wanted to work on her ROC abilities. Therefore she defined an hypothesis $h(x) = f(\langle (x; 1), w \rangle)$, where $f(\cdot) : \mathbb{R} \to [0, 1]$ is the inverse of the logit function (We saw it in the lecture).
Use the w that you derived in (a), in the specific case where: $\mathcal{X} = \mathbb{R}$, $\mu_0 = 4$, $\mu_1 = 6$, $\sigma^2 = 1$ and $\pi_0 = \pi_1 = 1/2$.

Remark 3 *The μ_0 was changed.*

i. (1 Pts.) Plot the pdf and the cdf distributions of both Gaussians over $\mathcal{X}$.
You may use- scipy.stats.norm.cdf in order to generate the graph.

ii. (1.5 Pts.) Draw a plot of $h(x)$ as a function of x .

iii. (5 Pts.) Draw the cdf distribution of $h(x)$ for $x \sim X|(Y = 0)$ and for $x \sim X|(Y = 1)$ in separate graphs. (You may calculate the value for 1000 points between [0,1] and connect the curve between them).

iv. (5 Pts.) Chen wants to understand the ROC axis - TPR and FPR.
In order to help her, draw in the same graph:

- One minus the CDF of $h(Z_1)$ as a function of $h(Z_1)$, where Z_1 is a random variable with the same distribution as $X|(Y = 0)$
- One minus the CDF of $h(Z_2)$ as a function of $h(Z_2)$, where Z_2 is a random variable with the same distribution as $X|Y = 1$.

Remark 4 *In the original version we wrote that we want the CDF of h(Z_1) as a function of Z_1- this was a typo.*

v. (2.5 Pts.) Use the graph in order to calculate the FPR and TPR at $t \in \{0.2, 0.4, 0.55, 0.95\}$.

Remark 5 *Make sure that you understand that t is a threshold that you apply to $h(Z_1)$ and $h(Z_2)$ and not to Z_1 and Z_2 directly.*

vi. (5 Pts.) For each of the mentioned t find the region in $\mathcal{X}$ in which our classifier, h_t, will return one. Display the lowest point in that region on a similar plot to the one you drew in (i) that considers only the pdf distributions.

vii. (5 Pts.) Draw the ROC curve of h.

Some important little questions (10 points- 2.5 points each)

6. Given two hypothesis $h_1, h_2 : \mathcal{X} \to [0, 1]$.
Definition AUC (Area under the cure)- The AUC of h_1 is the area between the ROC curve and x-axis.
The AUC is a known heuristic that is used for determining how "good" is your h.

(a) If the AUC of h_1 is higher then h_2. Is it always better to use h_1 then h_2 for any scenario? Give an example that will support your claim.

(b) If the ROC curve h_1 is higher then h_2 at each point. Is it always better to use h_1 then h_2 for any scenario? Give an example that will support your claim.

(c) Welcome to "Sigma-Beta-Gamma", you are the chief technology officer. Your engineers provide you with an hypothesis h ($h : \mathcal{X} \to [0, 1]$) and even supply you with its ROC:

$$f(z) = \begin{cases} 2z & z \leqslant 0.3 \\ 0.6 & z \in [0.3, 0.5] \\ 0.6 + (z - 0.5)/2 & z \in [0.5, 1] \end{cases}$$

where f is the TPR, and z is the FPR (We strongly suggest to draw it).
As you want to optimize both the FPR and TPR, what are the regions that you won't consider while trying to find the optimal threshold for your necessities? explain.

(d) Answer this questions for LDA and logistic regression.
When we train our model we try to approximate the real distribution of X, Y. Yes/No, explain.

Hands-on (30 pts)

7. Download the Spam dataset from here, read the info file to understand what this dataset contains. Then follow these intructions:

   (a) Draw 1000 data points from the dataset and keep them aside as a test set.

   (b) Fit a Logistic Regression model on the rest of the data, using the sklearn class. Use the default setting in the constructor (i.e. just call it with no arguments). Next we will use the predict_proba method and draw the empirical ROC curve calculated over the test set:

   - Use predict_proba on the test set and sort according to the probability of the classifier to predict $y = 1$ (see np.argsort).
   - Denote the number of positives in the test set (i.e. the number of test samples whose true label is 1) as NP. Using our sorted array from earlier, we can see for each $i \in [NP]$, how many samples the classifier needs to tag with label 1 in order to get a TPR of $\frac{i}{NP}$. Lets call this N_i.
   - The FPR we get now if we fix a TPR of $\frac{i}{NP}$ is $\frac{N_i - i}{NN}$ (where NN is the number of negatives in the test set).
   - Now consider the points $(\frac{N_i - i}{NN}, \frac{i}{NP})$ for all $i \in [NP]$, if we plot these points along with $(0, 0), (1, 1)$ we will get the ROC curve.
   - Repeat the above procedure for 10 times, plot the average curve over these 10 repetitions and hand in the plot.

     Remark 6 *In each repetition you new to draw a new set of 1000 data points, and therefore your will get a new test set.*
     For the last question you may plot the 10 curves you got on the same figure, instead of plotting their average.

   (c) Implement a k-nearest neighbors classifier:

   - write a file called knn.py with a class called knn. The constructor should take a parameter k that determines the number of nearest neighbors for the classifier. The class should have a fit$(X, \mathbf{y})$ method that simply stores the data and a predict$(\mathbf{x})$ method that predicts $\mathbf{x}$'s label according to the majority of its k nearest neighbors (use Euclidean distance).
     For each k in $\{1, 2, 5, 10, 100\}$ train a classifier using the same training data you did in the last section. Compare their test error over the same data you used in the logistic regression part as well (You may use a table for that).

     Remark 7 *In the logistic regression you needed to repeat the the training procedure for 10 times, and return the average error on the test set. Here, we want you to do the same - meaning, train 10 times the classifier for each k, and return for each k the average test error over the 10 runs.*
     We will also accept answers that will return all the test errors that you got over the ten different runs for each k.

- Try and explain the tradeoff between choosing low and high k- give examples to back your claim.

(d) QDA (Quadratic discriminant analysis)- In class we saw the LDA classifier which assumed that the distribution of $X|Y = y$ is Gaussian with a covariance matrix that is the same for any $y \in \{1, \ldots, k\}$. The QDA models drops the shared covariance matrix assumption, meaning that the distribution of $X|Y = y$ may have a different covariance for different values of y.

Remark 8 *You should implement the LDA and QDA by yourself, and not the sklearn package for it.*

Remark 9 *The QDA problems assumes that* $X|Y = 0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ *and* $X|Y = 1 \sim \mathcal{N}(\mu_1, \Sigma_1)$*, unlike the LDA problem where the* Σ *is shared.*

The $\pi_0, \pi_1, \mu_0, \mu_1$ *estimation is done the same as in the LDA model. As the covariance matrix is not shared, one will need to estimate it for* $X|Y = 0$ *and* $X|Y = 1$ *separately. How it is done? Check out the Covariance Matrix Estimation subsection in the summary of lecture 1.*

And now for the optimization problem of QDA- Follow the proof of Exercise 1, but replace the Σ *with* Σ_y*, as it is not shared anymore. That will lead you to write* $h_{\mathcal{D}}$ *for the QDA in a similar form to the LDA one.*

- write a file called *QDA.py* with a class called *QDA*. As in the knn implement in the class the functions *fit(X,y)* and *predict(x)*
- write a file called *LDA.py* with a class called *LDA*. As in the knn implement in the class the functions *fit(X,y)* and *predict(x)*
- Train both classifiers on the training data as done in k-nn case, and:
  i. Choose your features- as the determinant can get to 0, find a subset of at least 5 features so that:
     – the determinant of your covariance matrices won't get to 0
     – your error won't be above 0.5- as this will mean that your classifier is useless.

  ii. Compare the eigenvalues of the covariance matrices that you got in both classifiers, and explain using it how the distributions learned in the QDA are different then the one learned in the LDA.
  (If you are not sure how to answer this question please check Question 8)

  Remark 10 *While you are requested to run to run the algorithm for 10 times, you may answer the above question based on a single run.*

  iii. Compare the average train errors of both classifiers.

  Remark 11 *both= LDA, QDA.*

  iv. Compare the average test errors of both classifiers.

8. (Optional) Multivariate Gaussian-

(a) Consider a d-dimensional Multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. Define for any $\alpha \in (0, \infty)$:

$$C_\alpha = \{x \in \mathbb{R}^d : (x - \mu)^T \Sigma^{-1}(x - \mu) = \alpha\}$$

Show that any two points $x_1, x_2 \in \mathbb{R}^d$ share the same C_α iff they have the same pdf.

(b) Consider the two Gaussian distributions $\mathcal{N}(1,1)$, $\mathcal{N}(6,9)$- which one is more concentrated around its mean?

(c) Consider the distribution:

$$\mathcal{N}\left(\begin{bmatrix} 1 \\ 6 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 9 \end{bmatrix}\right)$$

which is basicly the concatenation of the two distribution from (a), while assuming they are independent.
Choose $\alpha_1, \alpha_2 \in (0, \infty)$ and plot all the points that are in C_{α_1} and C_{α_2}.

(d) In which direction the distribution mentioned in (8c) is more concentrated around the mean? (What are the eigenvectors of the covriance matrix?)

Remark 12 *Now, for the general case- use a covariance matrix the is not diagonal necessarily.*
It is important for you to remember that a covariance matrix is always symmetric and contain only non negative eigenvalues (You may try and show it).
In the Gaussian case we consider only covariance matrices that are of full rank.