Course 67564	2022/23

Bayesian Machine Learning
Exercise 5: Classification

Prof. Yair Weiss — *TA: Roy Friedman*

Deadline: January 12, 2022

1 Decision Boundaries

In this part of the exercise we will use the generative classification setting in order to visualize the (linear) decision boundary between classes[1].

Given a data set $\mathcal{D} = \{x_i\}_{i=1}^N$, we would like to fit the mean of a Gaussian to this data. Suppose that the log-likelihood we use is:

$$\log p\left(\mathcal{D}|\mu\right) = \sum_i \mathcal{N}\left(x_i \,|\, \mu, I\sigma^2\right) \tag{1.1}$$

where we have a prior over the mean μ of the form:

$$\mu \sim \mathcal{N}\left(\mu_0, I\sigma_0^2\right) \tag{1.2}$$

Recall that the posterior for μ is then given by:

$$\mu|\mathcal{D} \sim \mathcal{N}\left(\frac{\frac{1}{\sigma^2}\sum_i x_i + \frac{1}{\sigma_0^2}\mu_0}{N\frac{1}{\sigma^2} + \frac{1}{\sigma_0^2}}, \frac{1}{N\frac{1}{\sigma^2} + \frac{1}{\sigma_0^2}} I\right) \tag{1.3}$$

In the binary, 2D, linear discriminant analysis (LDA) classification scheme, the decision boundary between the classes (assuming that the covariance of both classes is I) is given by:

$$\left(\mu_+ - \mu_-\right)^T \begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{2}\left(\|\mu_+\|^2 - \|\mu_-\|^2\right) \tag{1.4}$$

$$\Leftrightarrow \left[(\mu_+ - \mu_-)\right]_1 x + \left[(\mu_+ - \mu_-)\right]_2 y = \frac{1}{2}\left(\|\mu_+\|^2 - \|\mu_-\|^2\right) \tag{1.5}$$

$$\Leftrightarrow y = \frac{\left(\|\mu_+\|^2 - \|\mu_-\|^2\right)}{2\left[(\mu_+ - \mu_-)\right]_2} - \frac{\left[(\mu_+ - \mu_-)\right]_1}{\left[(\mu_+ - \mu_-)\right]_2} x \tag{1.6}$$

where μ_+ and μ_- are the estimated means of each class and $\left[(\mu_+ - \mu_-)\right]_i$ is the i-th coordinate of the vector $(\mu_+ - \mu_-)$.

1. Let:

$$\mu_+ \sim \mathcal{N}\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix}, I\sigma_0^2\right)$$
$$\mu_- \sim \mathcal{N}\left(\begin{pmatrix} -1 \\ -1 \end{pmatrix}, I\sigma_0^2\right)$$
$$x^{(+)} \sim \mathcal{N}\left(\begin{pmatrix} 1/2 \\ 0 \end{pmatrix}, I\sigma^2\right)$$
$$x^{(-)} \sim \mathcal{N}\left(\begin{pmatrix} -1/2 \\ -1/2 \end{pmatrix}, I\sigma^2\right)$$

   with $\sigma_0^2 = 0.25$ and $\sigma^2 = 0.1$. Sample and plot 5 points from $x^{(+)}$ and another 5 from $x^{(-)}$. Find the MMSE estimate for μ_+ and μ_- under the posterior in equation 1.3 and plot the mean decision boundary, using equation 1.6. Sample and plot another 10 decision boundaries by sampling the class means from the posteriors $\mu_+|\mathcal{D}$ and $\mu_-|\mathcal{D}$

[1] For this exercise, you will essentially implement the demo we saw in recitation 9.

2 Image Classification

In this part of the exercise we will classify between images of dogs and frogs[2]. The dogs data set can be found in the file `dogs.npy` while the frogs data set can be found in the file `frogs.npy`. Both data sets contain 6000 images - 5750 for training and 250 for testing. An example of how to load the data can be found in the supplied code `ex5_utils.py`, under the name `load_im_data()`.

As part of the classification tasks, you will need to calculate the accuracy of the models you trained. The accuracy of a model is defined as:

$$\text{accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[y_i - \hat{y}_i\right] \tag{2.1}$$

where y_i is the true label for the i-th sample and $\hat{y}_i$ is the label predicted by your model.

2.1 Generative Classification of Images

In this section we will create a generative classifier for images; specifically, we will use the quadratic discriminant analysis (QDA) we saw in recitation 9. To do so, we will fit a multivariate Gaussian to each class of images independently and then compare the likelihoods in order to classify between them:

$$\hat{y} = \arg\max_{c\in\{-1,1\}} \log p\left(y=c|x\right) \tag{2.2}$$

$$= \arg\max_{c\in\{-1,1\}} \log\left[p\left(y=c\right)p\left(x|y=c\right)\right] \tag{2.3}$$

where:

$$p\left(x|y=c\right) = \mathcal{N}\left(x \,|\, \mu_c, \Sigma_c\right) \tag{2.4}$$

In this exercise we will assume that $p\left(y=-1\right) = p\left(y=1\right) = \frac{1}{2}$.

One subject we haven't talked about in class is of using Bayesian estimation in order to find the covariance of a Gaussian. The reason this hasn't come up in class is because it involves distributions over matrices, which we didn't want to get into. However, in practice it is quite easy to incorporate a prior over the covariance. The appropriate conjugate prior for the covariance of a Gaussian is called the inverse Wishart distribution (denoted by $\mathcal{W}^{-1}$) which has two parameters: $\nu \in \mathbb{R}_+$ and $\Psi \in \mathbb{R}^{d\times d}$ which is a PD matrix. The inverse-Wishart distribution is a distribution over PD matrices, which makes it perfect for modeling the covariance of a Gaussian distribution. In this exercise we will assume that:

$$\Sigma_c \sim \mathcal{W}^{-1}\left(\nu + d - 1, \Psi\right) \tag{2.5}$$

where $\Psi = \nu \cdot I\beta$ for some $\beta > 0$. The MMSE estimate for the Gaussian distribution will then be[3]:

$$\hat{\Sigma}_c = \frac{\nu \cdot I\beta + \sum_{i:\, y_i=c}\left(x_i - \mu_c\right)\left(x_i - \mu_c\right)^T}{\nu + N} \tag{2.6}$$

Notice that when $\nu = 0$, we get the ML solution. On the other hand, if $\nu \to \infty$ then the classifier becomes an LDA classifier as all classes will have the same covariance.

In the supplied `ex5_utils.py` there is an implementation of the `Gaussian` class for Bayesian estimation of a multivariate Gaussian, that should be used for classification in this section. The same file contains examples of how to use this class. We will use $\Psi = \nu \cdot I\beta$ with $\beta = 0.05$ and different values of ν:

2. For each class (dogs and frogs) fit a Gaussian with $\nu = [0, 1, 5, 10, 25, 50, 75, 100]$. Use the definition from equation 2.3 to classify both the training and test sets. Plot the accuracy of the QDA classifier as a function of the value of ν on both the training and test sets

3. What is the test accuracy for $\nu = 0$? What about $\nu = 25$? Why is there such a difference?

[2] Both sets of images are taken from the CIFAR10 data set.

[3] As with the Dirichlet distribution we saw, here we can think of the prior as us observing "ν points with $I\beta$ covariance" before ever seeing the data points.

2.2 Discriminative Classification of Images

In this section we will use a Gaussian process regression model to classify images with the "classification-by-regression" method. To do this, we will label the regression targets as ± 1 - positive for the first class and negative for the second class. In other words, we will try to fit the following model:

$$f_\theta(x) = \sum_i \alpha_i k(x_i, x) \tag{2.7}$$

where the training data is $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ where the $y_i \in \{-1, 1\}$, according to the class.

One of the attractive features of probabilistic models is that we also know how to report their uncertainty in their predictions. While this is not straightforwatd for classification-by-regression (which we are using in this exercise), we can still give indications of the uncertainty. When using regression to predict the values of each class, we can also calculate the variance of that prediction, as we have seen many times before. Suppose our regression y_n for a new test point x_n is given by[4]:

$$y_\mathrm{n} \sim \mathcal{N}\left(\mu_\mathrm{n}, \sigma_\mathrm{n}^2\right) \tag{2.8}$$

Given this Gaussian distribution, we can ask "how many standard deviations from the predicted value is the decision boundary". If the decision boundary (i.e. 0) is very close to the prediction, then the model will be very uncertain regarding the true sign. On the other hand, if the prediction is very far from 0, then the model will be very certain in the classification. We can formalize this using[5]:

$$d(y_\mathrm{n}) = \left|\frac{\mu_\mathrm{n}}{\sigma_\mathrm{n}}\right| \tag{2.9}$$

If $d(y_\mathrm{n})$ is much larger than 0 then the model is very sure that the prediction is correct. On the other hand, if this is very close to 0, this means that the model is very unsure in the prediction. We can use this value to find the samples the model is least certain about and, at the very least, express this uncertainty. In this exercise, we will use this uncertainty metric to display which data points from the dog test set are least and most "dog-like" according to the model, which should give us some more insight into the predictive powers of the model.

In the supplied `ex5_utils.py` file you will find an implementation of the `GaussianProcess` class that should be used for classification in this section. You can find examples of how to use this class in the same file. For this part of the exercise, we will use an RBF kernel with $\beta = 0.009$ and sample noise $\sigma^2 = 0.1$ for the regression.

4. Train a GP regression model with $N = [250, 500, 1000, 3000, \text{all}]$ samples from each class and plot the training and test accuracy as a function of N. What is the test accuracy when using all of the training data ($N = \text{all}$)?

5. Fit a GP regression model on all of the training data. Show the 25 dogs from the test set that had the highest $d(\cdot)$ and the 25 with the lowest $d(\cdot)$, as defined in equation 2.9. Are there any noticeable differences between the 25 most and least confident images of dogs?

3 Submission Guidelines

Submit a single zip file named "`ex5_<YOUR ID>.zip`". This file should contain your code, along with an "`ex5.pdf`" file in which you should add the figures/text for the practical part. Please write readable code, as the code will also be checked manually (and you may find it useful in the following exercises). In the submitted code, please make sure that you write a basic main function in a file named "`ex5.py`" that will run (without errors) and produce all of the results that you showed in the pdf of answers that you submitted. The only packages you should use are `numpy`, `scipy` and `matplotlib`. You may also reuse code from your previous exercise in order to answer the questions in this exercise, if needed.

In general, it is better if you type your homework, but if you prefer handwriting your answers, please make sure that the text is readable when you scan it.

Part of your assignment will be graded by submitting your answers through Moodle, at this link. In each of the questions, write the answer to the corresponding question for grading. These answers will be graded automatically, so write only numeric values where needed.

[4] See equation 1.2 in the summary of recitation 8 for the exact distribution

[5] Formally, if $\mathbb{E}[y_\mathrm{n}] > 0$, what we should actually calculate is $p(\mathrm{sign}(y_\mathrm{n}) = -1) = p(y_\mathrm{n} \leq 0) = \int_{-\infty}^0 p(y_\mathrm{n})\, dy_\mathrm{n}$. Since we are using a Gaussian, this will actually be quite close to $d(y_\mathrm{n})$, but otherwise it may be trickier.

4 Supplementary Code

In the file `ex5utils.py` you can find an example of how to load the supplied data as well as a few helper functions. In addition, the same file has implementations of the `Gaussian` class as well as the `GaussianProcess` class. You can use this code as you see fit, and change any part of it that you want, just be sure to submit it as well if you change it. Finally, we have also supplied an outline code which you can use to get started in `ex5.py`. You don't have to use the format we outlined, but your code must run without errors and you must submit the plots required in the exercise description.

Good luck!