ORF525 (Spring 2016) Statistical Learning and Nonparametric Estimation **Assignment: 4**

Classification Methods

Lecturer: Han Liu *Email: hanliu@princeton.edu*
Designer: Junwei Lu

Due Date: Mar 23rd 5:00 pm.

Q1. Big Brother Is Watching You!

In the homeworks of previous weeks, we are always using statistics for good things. This time we will play as a bad guy! In the latest season of *House of Cards* (spoiler alert!!), President Underwood asks National Security Agency (NSA) to secretly spy on people of America through CCTV to help him win the presidential election. Now you are part of the conspiracy and are selected by the president to finish this job.

Since the video dataset is super big, you need to build an automatic human detector that tells us whether there is a upright human in a given photo. We treat this as a classification problem with two classes: having humans or not. You are provided with two datasets[1] `POS` and `NEG` that have photos with and without upright humans respectively.

1.1: Preprocessing the data

(a) Randomly pick out one image in `NEG` and one in `POS`. For each of the two images, implement each step below to vectorize and extract useful information. We provide functions that will be used in the following steps in a R-script `functions.r`. Please CAREFULLY read the appendix for the detailed introductions of these functions.

1. Download and install the package `png`, and use the function `readPNG` to load photos[2].
2. Use the function `rgb2gray`[3] to transform original photos to the black and white version.
3. Since photos in `NEG` have bigger sizes than those in `POS`, we need to crop them to keep consistency in dimensions. So for all photos `NEG`, use the function `crop.r` to randomly crop a 160×96 picture[4] from the original one.

[1]Photos in `POS` have 160×96 pixels, while photos in `NEG` have larger sizes in `POS`. All photos are stored as `png` files.
[2]Note that the output of `readPNG` is a three-dimensional array. The first two dimensions identify the position of pixels, and the third dimension identifies the channels.
[3]The output of `rgb2gray` will be a grayscale matrix.
[4]A image with height h and width w is denoted $h \times w$

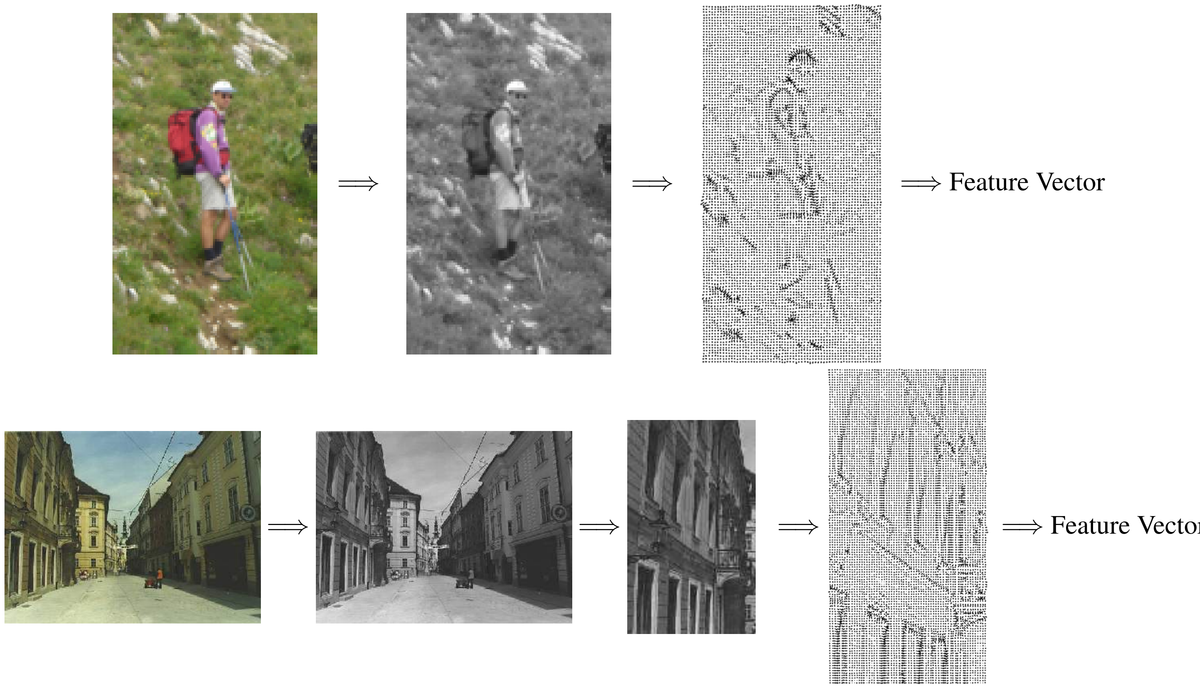

Figure 1: Illustration of feature extraction. The first row corresponds to a positive example from `POS`, while the second row corresponds to a negative one from `NEG`.

4. Use the function `grad` to obtain the gradient field of the center 128×64 part of the grayscale matrix.
5. Use the function `hog` (Histograms of Oriented Gradient) to extract a feature vector from the gradient field obtained in the previous step. Partition the height and width into 4 partitions each. Partition the angles into 6 intervals. (Your feature vector should then have $4 \times 4 \times 6 = 96$ components.) Please see the appendix for parameter configuration of this function.

For each of the two images, provide a picture showing each step above i.e. the original picture $\rightarrow$ the black and white picture $\rightarrow$ the cropped picture (for `NEG` only) $\rightarrow$ the gradient field $\rightarrow$ the feature vector. An example of the procedure is illustrated in Fig.1. For the feature vector, report its first six components.

(**Hint 1:** You can use `writePNG(X, target = 'filename.png')` to write some image array `X` to a png file. For exporting the plot generated by `grad`, you can use the following code:

```
setEPS()
postscript("test.eps")
g=grad(X, 128, 64, T)
dev.off()
```

The code above puts the 128×64 gradient field in the object `g`, and also saves the plot in a new image file `test.eps`.

Hint 2: You may find the following function useful to you.

1. `rgb2gray(X)` transforms colored pictures to their black and white versions. `X` is a three dimensional array, where the first two dimensions index the position of the pixel and the last dimension denotes the four channels (R, G, B and A). Search the key word "RGBA" if you are interested in image data representation. The output

will be a grayscale matrix that corresponds to the black and white version of the original picture.

2. `crop.r(X, h, w)` randomly crops a sub-picture that has height h and width w from `X`. The output is therefore a sub-matrix of `X` with `h` rows and `w` columns.

3. `crop.c(X, h, w)` crops a sub-picture that has height h and width w at the center of `X`. The output is therefore a sub-matrix of `X` with `h` rows and `w` columns. This function helps the `hog(...)` function. For, the cropping in your assignment, use `crop.r()`.

4. `grad(X, h, w, pic)` yields the gradient field at the center part of the given grayscale matrix `X`. The center region it examines has height `h` and width `w`. It returns a list of two matrices `xgrad` and `ygrad`. The parameter `pic` is a boolean variable. If it is `TRUE`, the generated gradient filed will be plotted. Otherwise the plot will be omitted.

5. `hog(xgrad, ygrad, hn, wn, an)` returns a feature vector in the length of `hn*wn*an` from the given gradient field. `(xgrad[i,j], ygrad[i,j])` gives the grayscale gradient at the position `(i,j)`. `hn` and `wn` are the partition number on height and width respectively. `an` is the partition number on the angles (or the interval $[0, 2\pi)$ equivalently).

Hint 3: Histogram of Oriented Gradient: Here we give a brief introduction of what `hog(xgrad, ygrad, hn, wn, an)` does. First of all, it uniformly partitions the whole picture into `hn*wn` small parts with `hn` partitions on the height and `wn` partitions on the width. For each small part, it counts the gradient direction whose angle falls in the intervals $[0, 2\pi/\texttt{an}), [2\pi/\texttt{an}, 4\pi/\texttt{an}), ..., [2(\texttt{an}-1)\pi/\texttt{an}, 2\pi)$ respectively. So `hog` can get `an` frequencies for each small picture. Applying the same procedure to all the small parts, `hog` will have `hn*wn*an` frequencies that constitute the final feature vector for the given gradient field.)

(b) Now, apply the above procedure to obtain feature vectors for each image in the dataset. Concatenate the feature vectors together into the rows of a dataframe. Add an additional column indicating whether each row is in `POS` or `NEG`. This will be the dataset you will use for Question 1.2.

(**Hint:** Use the `dir()` function to get the names of all the files in a directory. The functions `rbind()` and `cbind()` combine vectors together row-wise and column-wise, respectively.)

1.2: Detect the upright men

In this question, we will apply both SVM and Logistic regression to train the model and compare their performance.

1. Download the package `kernlab`. Use the function `ksvm` to train the model. Given the tuning parameter C and the number of folds k, `ksvm` can return the cross-validation error. Examine how the cross-validation error changes as we tune C in the range of $[0.0001, 100]$ as follows:

   - Construct a sequence of 100 values of C such that $\ln(C)$ is an arithmetic series with $\ln(10^{-4})$ as the minimum and $\ln(10^2)$ as the maximum.
   - Plot out the misclassification error against $\ln(C)$
   - Find out the optimal C in the sequence that yields the lowest misclassification error.

2. Use the function `glmnet` to train the model via logistic regression and plot out the regularization path. More importantly, use the function `cv.glmnet` to do the cross validation with the option `type.measure="class"`. Plot the results.

3. Compare the lowest cross-validation error of SVM and logistic regression. Do they differ significantly?

Q2. Bayes Classifier

Suppose that $\mathbb{P}(Y = 1) = 1/3$, $\mathbb{P}(Y = -1) = 2/3$ and $X|Y = -1 \sim \text{Uniform}(-10, 5)$ and $X|Y = 1 \sim \text{Uniform}(-5, 10)$.

(a) Find an expression for the Bayes classifier and find an expression for the Bayes risk.

(b) Consider the classifier $h(x) = \text{sign}(\alpha + \beta x^2)$ where $\alpha, \beta \in \mathbb{R}$. Fine at least one classifier (α^*, β^*) minimizes the risk and what is its risk?

(c) Compute the hinge risk $R_\phi(\beta) = \mathbb{E}[(1 - Y\beta X)_+]$, where $(x)_+ = \max\{x, 0\}$.

Q3. New Insight of LDA

In this problem, we will show the linear discriminant analysis is equivalent to least square estimator. Suppose $\mathbb{P}(Y = 1) = p, \mathbb{P}(Y = -1) = 1 - p$, $\boldsymbol{X}|Y = -1 \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$ and $\boldsymbol{X}|Y = 1 \sim N(\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$. Suppose we observe the samples $\mathcal{D}_1 = \{Y_i, X_i\}_{i=1}^{n_1}$ for $Y_i = -1$ and $\mathcal{D}_2 = \{Y_i, X_i\}_{i=1}^{n_2}$ for $Y_i = 1$.

3.1 Derive the Bayes classifier for this model. What is the maximum likelihood estimator of p, $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ and $\boldsymbol{\Sigma}$? Plug your MLE to the Bayes classifier and prove that it can be expressed as $\text{sign}(\widehat{\boldsymbol{w}}^T\mathbf{x} + \widehat{b})$.

3.2 Suppose we have two classes $\mathcal{D}_1 = \{Y_i, X_i\}_{i=1}^{n_1}$ and $\mathcal{D}_2 = \{Y_i, X_i\}_{i=1}^{n_2}$. Let $n = n_1 + n_2$. We re-encode the two classes as $Y_i = -n/n_1$ if it belongs to $\mathcal{D}_1$ and $Y_i = n/n_2$ if it belongs to $\mathcal{D}_2$. Let $\widehat{\boldsymbol{w}}$ be the linear discriminant analysis solution you derived in Q3.1. Let the least square estimator be

$$(\widehat{\beta}_0, \widehat{\boldsymbol{\beta}}) = \underset{\beta_0, \boldsymbol{\beta}}{\text{argmin}} \sum_{i=1}^{n} \left(Y_i - \beta_0 - \boldsymbol{X}_i^T\boldsymbol{\beta}\right)^2.$$

Prove that $\widehat{\boldsymbol{\beta}} \propto \widehat{\boldsymbol{w}}$.

3.3 Construct a concrete binary class classification data sample $\mathcal{D}_1 = \{Y_i, X_i\}_{i=1}^{n_1}$ and $\mathcal{D}_2 = \{Y_i, X_i\}_{i=1}^{n_2}$ in which the data from the two classes are linear separable but the LDA does not separate the data. This implies that LDA is not always applicable.

Q4. The Limit of Support Vector Machine

Let $\mathcal{D} = \{(y_i, \boldsymbol{x}_i)\}_{i=1}^n$ be the classification dataset, where $y_i \in \{+1, -1\}$. We say a linear separator $\boldsymbol{w}$ ($\|\boldsymbol{w}\|_2 = 1$) for $\mathcal{D}$ has margin γ if $y_i \cdot (\boldsymbol{w}^T\boldsymbol{x}_i) > \gamma$ for all $1 \le i \le n$.[5] If there exists such a separator, we say $\mathcal{D}$ is separable by a margin of γ. We know that SVM is an algorithm to maximize the margin. In this problem, we will explore the limit of SVM by deriving the relations between the margin and the linear classifier.

4.1 This problem shows the lower bound of samples we need to have reasonable classification error. We consider the unit ball $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^d | \|\boldsymbol{x}\|_2 = 1\}$. Show that there exists two disjoint sets $\mathcal{D}_0, \mathcal{D}_1 \subseteq \mathcal{X}$ with margin γ, where $1/\gamma^2 \le d$ such that given any s samples from $\mathcal{D}_0$ and s samples from $\mathcal{D}_1$, where $s = 1/(100\gamma^2)$, there exists a unit vector $\boldsymbol{w}$ satisfying:

1. The vector $\boldsymbol{w}$ separates the $2s$ samples by a margin γ;

[5] Here we ignore the intercept because we can always add one more dimension to $\boldsymbol{x}$.

Figure 2: How narrow can a margin be before no algorithm can go through? — Bob Dylan

2. But $\boldsymbol{w}$ misclassifies at least $1/3$ of the points in $\mathcal{D}_0$ and $\mathcal{D}_1$.

(**Hint:** You may proof this by constructing $\boldsymbol{x}_i$'s with margin γ for any possible labels of y_i's. Therefore you can construct a $\boldsymbol{w}$ separating any training data but it misclassifies the remaining data.)

4.2 This problem shows that the margin may decrease exponentially with the dimension of the dataset on the cube $\{0,1\}^d \backslash \{\mathbf{0}\}$. Suppose we have points $\boldsymbol{x} \in \{0,1\}^d$ and we label $\boldsymbol{x}$ as $+1$ if and only if the least i for which $\boldsymbol{x}_i = 1$ is odd. Otherwise we label $\boldsymbol{x}$ as -1. Show that we can separate these two classes by a linear separator:

$$\sum_{i=1}^{d} \frac{(-1)^{i-1} x_i}{2^{i-1}} > 0.$$

Prove that we cannot have a linear separator for the above dataset with margin at least $1/f(d)$ where $f(d)$ is bounded above by a polynomial function of d.