Weekly Report 10

Utkarsh Patel
18EC30048

Topics Covered

1 Discriminant Functions

For each class $C_i, i = 1, \dots, K$, we define a discriminant function $g_i(\boldsymbol{x})$ and instance $\boldsymbol{x}$ is classified as class C_j such that $j = \underset{i}{\operatorname{argmax}}\, g_i(\boldsymbol{x})$. In Bayesian framework, we used posteriors to represent this discriminant functions. However, instead of computing the density functions within a class, it is much simpler to estimate the boundary. This approach becomes quite useful when we know that we can separate the instances just from the boundary. Various orders of discriminant function dictate the estimate we are using for such boundaries.

- **Linear Discriminant Functions**
  Let $\boldsymbol{x} \in \mathbb{R}^d$. For each class C_i, we define a weight vector $\boldsymbol{w}_i \in \mathbb{R}^d, w_{i0} \in \mathbb{R}$, such that discriminant function is given as $g_i(\boldsymbol{x}|\boldsymbol{w}_i, w_{i0}) = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0} = \sum_{j=1}^{d} w_{ij}x_j + w_{i0}$. This requires $\Theta(d)$ time and space complexity in computation.
- **Quadratic Discriminant Functions**
  This is a more general form, however rarely used. In this, the discriminant function is given as $g_i(\boldsymbol{x}|\boldsymbol{W}_i, \boldsymbol{w}_i, w_{i0}) = \boldsymbol{x}^T\boldsymbol{W}_i\boldsymbol{x} + \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0}$. This requires $\Theta(d^2)$ time and space complexity in computation.

2 Two Class Problems

When only two classes are present in the training examples, instead of defining a separate discriminant function for each of the class, it is better to use only one discriminant function, let's say given as, $g(\boldsymbol{x}|\boldsymbol{w}, w_0) = \boldsymbol{w}^T\boldsymbol{x} + w_0$. If $g(\boldsymbol{x}) > 0$, we say that it belongs to class C_1, otherwise it belongs to class C_2. Basically, $g(\boldsymbol{x}) = 0$ is a hyperplane that divides the instance space into two halves, where instances lying in same half belong to same class. It is similar to a perceptron learning used in neural networks.

3 Augmenting and Normalizing Input Instances

One problem with the linear discriminant function defined above is that the mapping is not linear (doesn't obeys superposition principal). In an effort to make this a linear transformation, so that we can use various tools of linear algebra for analysis, we augment the input instance $\boldsymbol{x} \to \boldsymbol{X}$ such that $\boldsymbol{X} = \binom{\boldsymbol{x}}{1}$ and modify the weight vector $\boldsymbol{w} \to \boldsymbol{W}$ such that $\boldsymbol{W} = \binom{\boldsymbol{w}}{w_0}$. Hence, after this augmentation, the discriminant function reduces to $g(\boldsymbol{X}|\boldsymbol{W}) = \boldsymbol{W}^T\boldsymbol{X}$. The objective is to compute such a vector $\boldsymbol{W}$ such that if $\boldsymbol{X} \in C_1$, we would have $g(\boldsymbol{X}) > 0$, and if $\boldsymbol{X} \in C_2$, then $g(\boldsymbol{X}) < 0$. For tracking error in our prediction, it would be much simpler if we apply another transformation to the instance $\boldsymbol{X}$ given as $\boldsymbol{Y} = \begin{cases} \boldsymbol{X}, & \boldsymbol{X} \in C_1 \\ -\boldsymbol{X}, & \boldsymbol{X} \in C_2 \end{cases}$. After this transformation, it can be observed that correctly classified instances give $g(\boldsymbol{Y}) > 0$. If $g(\boldsymbol{Y}) < 0$, that means that given instance has been misclassified.

4 Error in Classification using Linear Discriminant

We can define several errors with respect to linear discriminant. Let's define a set M which contains misclassified $\boldsymbol{Y}$s. Then, error functions can be:

- $E_1(\boldsymbol{W}) = -\sum_{\boldsymbol{Y}\in M} \boldsymbol{W}^T\boldsymbol{Y}$, this error function is not smooth and is influence more by $\boldsymbol{Y}$ having larger norm.
- $E_2(\boldsymbol{W}) = -\sum_{\boldsymbol{Y}\in M} (\boldsymbol{W}^T\boldsymbol{Y})^2$, this error function is smooth but still has non-uniform influence.
- $E_3(\boldsymbol{W}) = -\sum_{\boldsymbol{Y}\in M, \boldsymbol{W}^T\boldsymbol{Y}\le b} \frac{(\boldsymbol{W}^T\boldsymbol{Y}-b)^2}{\|\boldsymbol{Y}\|^2}$, smooth as well as equal influence.

5 Getting the Optimal Weight Vector $\boldsymbol{W}$

We use gradient descent for computing the optimal weight vector. The algorithm is as follows:

```
GRADIENT-DESCENT
1.  Initialise $\boldsymbol{W} \leftarrow RandomVector(dims = \boldsymbol{Y}.shape)$
2.  Iterate until convergence
        o  Let E be the error obtained in this iteration over entire sample
        o  $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta\frac{\partial E}{\partial \boldsymbol{W}}$; η being learning rate
```

The convergence can be achieved faster (as it has been observed in practice), if we use batch instead of entire sample for updating the weights.

6 Support Vector Machines

It is a linear discriminant classifier which uses Vapnik's principle (It is easier to solve for class boundaries as compared to computing class distribution. Classification can be done solely on the boundaries as well.). Outliers are the instances having low probability of occurrence. After training, the weight vector can be written in terms of training samples lying in class boundaries.

7 Linear Support Vector Machines

Linear support vector machines maximize the margin between two (or more) linearly separable classes. Let the training samples be given as $\mathcal{X} = \{\boldsymbol{x}^t, r^t\}$ where $r^t = 1$ if $\boldsymbol{x}^t \in C_1$, and $r^t = -1$ if $\boldsymbol{x}^t \in C_2$. In this framework, our objective is to find $\boldsymbol{w}, w_0$, such that $\forall t, r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) \ge 1$. Using 1 instead of 0 in this inequality increases the complexity and a margin is left in between the boundaries. The margin has per the equations is given as $L = \frac{2}{\|\boldsymbol{w}\|}$. We define our objective function as $J(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2$. Minimizing J under given constraints will maximize the margin L. As this is a constrained optimisation problem, we use Lagrange's multiplier to convert it to unconstrained optimisation problem.

$$\hat{J}(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_t \alpha_t(r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) - 1)$$

Above objective function needs to be minimised with respect to $\boldsymbol{w}, w_0$, and to maximised with respect to α_t.

$$\frac{\partial \hat{J}}{\partial \boldsymbol{w}} = 0 \to \boldsymbol{w} = \sum_t \alpha_t r_t \boldsymbol{x}_t\,; \quad \frac{\partial \hat{J}}{\partial w_0} = 0 \to \sum_t \alpha_t r_t = 0$$

Putting these values in objective function, we get

$$\hat{J}(\boldsymbol{w}) = -\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_t \alpha_t = -\frac{1}{2}\sum_t\sum_s \alpha_t\alpha_s r^t r^s (x^t)^T x^s + \sum_t \alpha_t$$

This objective function needs to be maximised with respect to α_t. If total number of samples is N, this can be done in $\Theta(N^3)$ time and $\Theta(N^2)$ space complexity. Generally, most of the α_t are zero. Therefore, instance for which α_t is non-zero, constitute support vectors. Only support vectors influence class boundaries.

8 Non-separable Case: Soft Margin Hyperplane

In some machine learning tasks, the classes might not be linearly separable. In this case, we use slack variables. Here, we need to have $\forall t, r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) \ge 1 - s^t$. Now, s^t varies over instances $\boldsymbol{x}^t$ as:

$$s^t \in \begin{cases} [0,1), \boldsymbol{x}^t \ is\ correctly\ classified \\ [1,\infty), \boldsymbol{x}^t \ is\ misclassified \\ (0,\infty), \boldsymbol{x}^t \ is\ non-separable \end{cases}$$

Soft error is defined to be $\sum_t s^t$. Now, we need to minimize $J(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \lambda\sum_t s^t$. Here, λ is the penalty factor. This results in additional Lagrange multipliers coming into picture to keep $s^t \ge 0$. Performing dual optimisation as in previous section, same objective function with additional constraints that $0 \le \alpha_t \le \lambda$.

9 Making Instances Linearly Separable

Instances $\boldsymbol{x} \in \mathbb{R}^d$ can be projected to higher-dimensions $\mathbb{R}^k$ using basis functions $\varphi_j(\boldsymbol{x}), j = 1, 2, \dots, k;\ \varphi_1(\boldsymbol{x}) = 1$, and discriminant is defined to be as $g(\boldsymbol{x}) = \sum_{j=1}^{k} w_j\,\varphi_j(\boldsymbol{x})$. However, there is no guarantee that the classes would be linearly separable in $\mathbb{R}^k$. Same dual optimisation is carried out in this case as well.

10 Kernel Machines

The discriminant function can be written as $g(\boldsymbol{x}) = \sum_t \alpha_t r_t K(\boldsymbol{x}_t, \boldsymbol{x})$
Therefore, we don't need to compute basis functions and compute dot product. Gram matrix is defined as matrix of kernel values $\boldsymbol{K}$, where $\boldsymbol{K}[i,j] = K(x^t, x^s)$. It is symmetric and positive-semidefinite matrix (all eigenvalues are non-negative).

11 Logits

Here, we define prediction as $\hat{y} = sigmoid(\boldsymbol{x})$ and a new kind of error called cross-entropy error is defined, which is given as

$$E = \sum_t y^t \ln(\widehat{y^t}) + (1 - y^t)\ln(1 - \widehat{y^t})$$

Novel Ideas Out of Lesson

There are various optimization to Gradient descent algorithm like RMSProp, Momentum, Adam, etc. which converge very fast.

About Quiz-3

Difficulty level was fair. However, there were many question for which the provided answers were wrong. Enough time was given for the quiz.