Decision Theory

Marko Sahan

April 4, 2019

1 Introduction to Decision Theory

Decision process is complicated set of actions that living being makes for satisfying its needs. We want to apply same concept for inanimate thing such as computers. Before constructing some theory we must define some terms with which we will work.

Lets assume that we want to solve classification problem. For simplicity we will work with binary classification problem. We want to find such solution that will assign for each input value its class. Moreover, we want to make classification error as small as possible. Considering that input data $\mathbf{x} \in \mathbf{X} \subset \mathbb{R}^n$ and $\mathbf{y} \in \mathbf{Y}$, where $\mathbf{x}$ is an input vectors of size n and $\mathbf{y}$ is its labels assigned to the data from space $\mathbf{X}$. Each value from space $\mathbf{Y}$ is one or zero $\mathbf{y} \in \{0, 1\}$ that represents first or second class.

In order to make a classification with respect to some input data we must provide an action or in other words decision. Let $a \in \mathcal{A}$ is an action and $\mathcal{A}$ is an action space. Of-course we do not want to make random decisions. We want to make decisions with respect to some metrics that can tell us how good our decision is. Thus, we will introduce a loss function L. From the previous text it is obvious that loss function will be dependent on action $a \in \mathcal{A}$. Furthermore, from the definition of the loss function it must be also dependent on a parameter. Let $\theta \in \Theta$ is parameters' vector and Θ is parameters space. As a result loss function L can be represented as

$$L = L(\theta, a). \tag{1}$$

However its impossible to have all the data $\theta \in \Theta$ and $a \in \mathcal{A}$. As a result we will introduce following definition.

Definition 1.1. If $\pi^*(\theta)$ is believed probability distribution of θ at the time of decision making, the *Bayesian expected loss* of an action a is

$$\rho(\pi^*, a) = \mathbb{E}_{\pi^*}[L(\theta, a)], \tag{2}$$

$$= \int_{\Theta} L(\theta, a) dF^{\pi^*}(\theta) \tag{3}$$

1.1 Decision Theory and Support Vector Machine Algorithm

In this subsection will continue construction of the decision theory on the example of Support Vector Machine (SVM) method. For simplicity lets consider linearly separable dataset. From the theoretical perspective SVM constructs hyperplane in high dimensional space that separates two classes. In this case our decision is a hyperplane that will separate two classes from each other. Equation of the hyperplane can be written as $f(\mathbf{x}, \mathbf{w}, b) = \mathbf{w}^T\mathbf{x} + b$ where $\mathbf{w} \in \mathbb{R}^n$ is a set of hyperplane parameters and $b \in \mathbb{R}$ is a bias . As a result, action space is represented as

$(\mathbb{R}^n, \mathbb{R}) = \mathcal{A}$ and as a consequence tuple $(\mathbf{w}, b) \in \mathcal{A}$. From this knowledge we can define $\theta = (\mathbf{x}, \mathbf{y})$ where tuple $(\mathbf{x}, \mathbf{y}) \in \mathbf{X} \times \mathbf{Y}$ is parameters and $\Theta = \mathbf{X} \times \mathbf{Y}$ is a parameters' space. Considering updated definitions, loss function (1) can be rewritten as

$$L = L(\mathbf{x}, \mathbf{y}, \mathbf{w}, b). \tag{4}$$

Following task is to understand how good is our action (hyperplane estimation) with respect to the dataset. We can choose different types of the loss functions such as cross entropy or hinge loss, etc. The most basic approach for SVM method is hinge loss function which is defined as

$$L(\mathbf{x}, \mathbf{y}, \mathbf{w}, b) = \max(0, 1 - \mathbf{y}\hat{\mathbf{y}}(\mathbf{x}, \mathbf{w}, b)) \tag{5}$$

where $\hat{\mathbf{y}}(\mathbf{x}, \mathbf{w}, b) = \mathbf{w}^T\mathbf{x} + b$.

If we had available data $\mathbf{X}$ and it its labels $\mathbf{Y}$ we would not have to construct all this theory because all labels would be known and no classification problem must be solved. However, in most cases we would have little discrete subset $\tilde{\mathbf{X}} \subset \mathbf{X}$ and $\tilde{\mathbf{Y}} \subset \mathbf{Y}$. On the basis of $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ we want to come up with a decision that will help us to label unlabeled data. In terms of SVM method we want to find such hyperplane that will label input values as a first class if it is "above" the hyperplane and as a second class if it is "below" the hyperplane. At this point very important assumption will be introduced. In order to find an optimal hyperplane we assume that the data $\tilde{\mathbf{X}}$ and its labels $\tilde{\mathbf{Y}}$ fully describe dataset $\mathbf{X}$ and $\mathbf{Y}$. Moreover we want to consider $\mathbf{x} \in \mathbf{X}$ and $\mathbf{y} \in \mathbf{Y}$ are random variables with joint probability density function $p(\mathbf{x}, \mathbf{y})$. We will also assume that $\forall i \in \{1, .., N\}$, $(\mathbf{x}_i, \mathbf{y}_i) \in \tilde{\mathbf{X}} \times \tilde{\mathbf{Y}}$ are independent identically distributed. Probability density function $p(\mathbf{x}, \mathbf{y})$ can be written as

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x}). \tag{6}$$

If we had those pdfs from (7) we could easily derive its joint pdf. However, we have only $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$. As a result, with the usage of those data, probability density function $p(\mathbf{x}, \mathbf{y})$ can be approximated as

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} \delta(\mathbf{x} - \mathbf{x}_i, \mathbf{y} - \mathbf{y}_i) \tag{7}$$

where $\delta(\mathbf{x} - \mathbf{x}_i, \mathbf{y} - \mathbf{y}_i)$ is Dirac delta function which is centered in $(\mathrm{x}_i, \mathbf{y}_i)$.

Using (2) we can evaluate expected loss function for SVM as follows

$$\begin{aligned}
\mathbb{E}_{\pi^*} L &= \int_{\mathbf{X}\times\mathbf{Y}} L(\mathbf{x}, \mathbf{y}, \mathbf{w}, b) p(\mathbf{x}, \mathbf{y}) d(\mathbf{x}, \mathbf{y}), \\
&= \int_{\mathbf{X}\times\mathbf{Y}} \max(0, 1 - \mathbf{y}\hat{\mathbf{y}}(\mathbf{x}, \mathbf{w}, b)) \frac{1}{N}\sum_{i=1}^{N} \delta(\mathbf{x} - \mathbf{x}_i, \mathbf{y} - \mathbf{y}_i) d(\mathbf{x}, \mathbf{y}), \\
&= \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - \mathbf{y}_i\hat{\mathbf{y}}(\mathbf{x}_i, \mathbf{w}, b))
\end{aligned}$$

where $\hat{\mathbf{y}}(\mathbf{x}_i, \mathbf{w}, b) = \mathbf{w}^T\mathbf{x}_i + b$. Expect loss function for SVM can be written as

$$\rho(\mathbf{x}_i, \mathbf{w}, b) = \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - \mathbf{y}_i\mathbf{w}^T\mathbf{x}_i + b). \tag{8}$$

1.2 Decision Theory and Algorithm Based on Neural Network Function

Decision Theory construction for the algorithm, based on a neural network function, is mostly the same as in 1.1. However in this case our decision is to finds estimate $\hat{\mathbf{y}} = \hat{\mathbf{y}}(\mathbf{x}, \mathbf{W}, \mathbf{b})$ of the probability density function $p(\mathbf{y}|\mathbf{x})$ where $\mathbf{x}$ is the input data, $\mathbf{y}$ is the label assigned to $\mathbf{x}$, $\mathbf{W}$ is a set of neural network function parameters and $\mathbf{b}$ is a vector of biases. Once again our action space $\mathcal{A}$ will be parameters' and biases' space of $\hat{\mathbf{y}}$. Same as in 1.1 we can define $(\mathbf{x}, \mathbf{y})$ are parameters of the loss function and $\mathbf{X} \times \mathbf{Y}$ is a parameters' space. For better understanding of the variety of loss functions we will use cross entropy loss function that is defined as

$$L(\mathbf{x}, \mathbf{y}, \mathbf{w}, b) = -\mathbf{y} \ln\big(\hat{\mathbf{y}}(\mathbf{x}, \mathbf{w}, b)\big) - (1 - \mathbf{y}) \ln\big((1 - \hat{\mathbf{y}}(\mathbf{x}, \mathbf{w}, b)\big). \tag{9}$$

Continuing using assumptions from 1.1 we consider $\mathbf{x} \in \mathbf{X}$ and $\mathbf{y} \in \mathbf{Y}$ are random variables with joint probability density function $p(\mathbf{x}, \mathbf{y})$. With the usage of the given dataset where $\forall i \in \{1, .., N\}$, $(\mathbf{x}_i, \mathbf{y}_i) \in \tilde{\mathbf{X}} \times \tilde{\mathbf{Y}}$ are independent identically distributed we can approximate $p(\mathbf{x}, \mathbf{y})$ as 7. Applying definition 1.1, expected loss for the algorithm based on a neural network function is evaluated as

$$\begin{aligned}
\mathbb{E}_{\pi^*} L &= \int_{\mathbf{X}\times\mathbf{Y}} L(\mathbf{x}, \mathbf{y}, \mathbf{w}, b) p(\mathbf{x}, \mathbf{y}) d(\mathbf{x}, \mathbf{y}), \\
&= -\int_{\mathbf{X}\times\mathbf{Y}} \Big(\mathbf{y} \ln\big(\hat{\mathbf{y}}(\mathbf{x}, \mathbf{w}, b)\big) + (1 - \mathbf{y}) \ln\big((1 - \hat{\mathbf{y}}(\mathbf{x}, \mathbf{w}, b)\big)\Big) \frac{1}{N} \sum_{i=1}^{N} \delta(\mathbf{x} - \mathbf{x}_i, \mathbf{y} - \mathbf{y}_i) d(\mathbf{x}, \mathbf{y}), \\
&= -\frac{1}{N} \sum_{i=1}^{N} \Big(\mathbf{y}_i \ln\big(\hat{\mathbf{y}}_i(\mathbf{x}, \mathbf{w}, b)\big) + (1 - \mathbf{y}_i) \ln\big((1 - \hat{\mathbf{y}}_i(\mathbf{x}, \mathbf{w}, b)\big)\Big).
\end{aligned} \tag{10}$$

1.3 Decision Theory and Naive Bayes Algorithm

Naive Bayes algorithm same as neural network based algorithms or SVM tries to find an estimate $\tilde{p}(\mathbf{y} = k|\mathbf{x}, \mathbf{w})$ of probability density function $p(\mathbf{y} = k|\mathbf{x})$ where $\mathbf{w}$ is set of parameters of a pdf, $\mathbf{x}$ is the input data, $\mathbf{y}$ is the label assigned to $\mathbf{x}$ and k is a class. Before continuing with a loss function construction we would like to go trough Naive Bayes (NB) method. Assuming binary classification problem. With the usage of Bayes rule we can rewrite $p(\mathbf{y} = k|\mathbf{x})$ where $k \in \{0, 1\}$ as follows

$$p(\mathbf{y} = k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{y} = k) p(\mathbf{y} = k)}{\int_{\mathbf{Y}} p(\mathbf{x}|\mathbf{y}) p(\mathbf{y}) d\mathbf{y}}. \tag{11}$$

However, Naive Bayes method introduces in equation (11) very strong assumption that features of vector $\mathbf{x} = (x_1, x_2, ..., x_n)^T$ are conditionally independent. As a result estimation of $p(\mathbf{y} = k|\mathbf{x})$ is

$$\tilde{p}(\mathbf{y} = k|\mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{w})} p(\mathbf{y} = k) \prod_{i=1}^{n} p(x_i|\mathbf{y} = k, \mathbf{w}_i = \mathbf{w}_{i,k}), \tag{12}$$

where $p(\mathbf{y} = k)$ is probability density function of the labels, $p(x_i|\mathbf{y} = k, \mathbf{w} = \mathbf{w}_{i,k})$ is a probability density function of the data features conditioned with label, $Z(\mathbf{w})$ is a normalizing parameter for class k and $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n)$. It is important to highlight that we can have parametric distribution $p(x_i|\mathbf{y} = k, \mathbf{w}_i = \mathbf{w}_{i,k})$ that is dependent on parameters vector $\mathbf{w}_{i,k}$. We must understand that for each class $\mathbf{y} = k$ parameters vector $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n)$ will be different. As a result for each feature $i \in \{1, 2..., n\}$ we introduce index k and write $\mathbf{w}_i = \mathbf{w}_{i,k}$ for $\mathbf{y} = k$. In

this case loss function will be derived from (12). We want to maximize our probability estimate $\tilde{p}(\mathbf{y} = k|\mathbf{x}, \mathbf{w})$. Thus, loss function L will be represented as

$$L(\mathbf{y}, \mathbf{x}, \mathbf{w}) = -\log\big(\tilde{p}(\mathbf{y} = k|\mathbf{x}, \mathbf{w})\big), \tag{13}$$

$$= \log(Z(\mathbf{w}) - \log\big(p(\mathbf{y} = k)\big) - \sum_{i=1}^{n} \log\big(p(x_i|\mathbf{y} = k, \mathbf{w}_i = \mathbf{w}_{i,k})\big). \tag{14}$$

Same as in 1.1 and 1.2 we will assume that we can approximate $p(\mathbf{x}, \mathbf{y})$ as 7. From this moment everything is ready for deriving expected loss function. Expected loss function for Naive Bayes method is derived as

$$\mathbb{E}_{\pi^*} L = \int_{\mathbf{X}\times\mathbf{Y}} L(\mathbf{x}, \mathbf{y}, \mathbf{w}) p(\mathbf{x}, \mathbf{y}) d(\mathbf{x}, \mathbf{y}),$$

$$\int_{\mathbf{X}\times\mathbf{Y}} \Big(\xi_k(\mathbf{w}) - \sum_{i=1}^{n} \log\big(p(x_i|\mathbf{y} = k, \mathbf{w}_i = \mathbf{w}_{i,k})\big)\Big) \frac{1}{N} \sum_{j=1}^{N} \delta(\mathbf{x} - \mathbf{x}_j, \mathbf{y} - \mathbf{y}_j) d(\mathbf{x}, \mathbf{y}),$$

$$\frac{1}{N} \sum_{j=1}^{N} \Big(\xi_{j,k}(\mathbf{w}) - \sum_{i=1}^{n} \log\big(p(x_i|\mathbf{y} = k, \mathbf{w}_i = \mathbf{w}_{i,k})\big)\Big) \tag{15}$$

where $\xi_k(\mathbf{w}) = \log(Z) - \log\big(p(\mathbf{y} = k)\big)$ **and** $\xi_{j,k}(\mathbf{w}) = \log(Z(\mathbf{w}) - \log\big(p(\mathbf{y}_j = k)\big)$**.**

Due to the fact that NB method theory is slightly different with 1.1 and1.2 we introduce here and example when

$$p(x_i|\mathbf{y} = k, \mathbf{w} = \mathbf{w}_{i,k}) = \mathcal{N}(w_{1,i,k}, w_{2,i,k}). \tag{16}$$

Expected loss (15) for Gaussian likelihood case can be written as

$$\mathbb{E}_{\pi^*} L = \frac{1}{N} \sum_{j=1}^{N} \Big(\tilde{\xi}_{j,k}(\mathbf{w}) + \sum_{i=1}^{n} \frac{(x_{i,j} - w_{1,i,k})^2}{2w_{2,i,k}^2}\Big)$$

where $\tilde{\xi}_{j,k}(\mathbf{w})$ is normalizing parameter calculated from $\xi_{j,k}(\mathbf{w})$ and additional values of Gaussian pdf. In this case

$$w_{1,i,k} = \frac{1}{N} \sum_{j=1}^{N} x_{i,j}$$

and

$$w_{2,i,k} = \frac{1}{N} \sum_{j=1}^{N} (x_{i,j} - w_{1,i,k})^2.$$

1.4 Decision Theory and Random Forest Algorithm

In order to work with random forests we must precisely define decision trees and only then construct theory for random forests.

1.4.1 Decision Tree

In this section we expect our decision three to give us also an estimate $\hat{\mathbf{y}}(\mathbf{y}, \mathbf{x}, \mathbf{w})$ of $p(\mathbf{y}|\mathbf{x})$ where $\mathbf{w}$ is a vector of Yes or No decisions, $\mathbf{x} \in \mathbf{X} \subset \mathbb{R}^n$ and $\mathbf{y} \in \mathbf{Y}$, where $\mathbf{x}$ is an input vectors of size n and $\mathbf{y}$ is its labels assigned to the data from space $\mathbf{X}$. Parameter space will be same as in 1.1-1.3, whereas action $a \in \mathcal{A}$ will represent a vector of Yes and No decisions $\mathbf{w}$. Each feature bla bla bla...