Decision Theory

Marko Sahan

April 20, 2019

Part I
Introduction

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 exists data $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$ and $\mathbf{y} \in \mathcal{Y}$, where $\mathbf{x}$ are vectors of size n and $\mathbf{y}$ are its labels assigned to the data from space $\mathcal{X}$. In this work we will use several definitions of the values from space $\mathcal{Y}$. Each value from space $\mathcal{Y}$ can be represented as a one hot representation that consist from ones and zeros $\mathbf{y} \in \{(0,1)^T, (1,0)^T\}$ or as a scalars $y \in \{0,1\}$ (bold $\mathbf{y}$ is a vector that is why if we talk about a scalar we have to redefine $\mathbf{y}$ as y). These representations can be different with respect to the overviewed methods. If we $\mathbf{y}$ is defined as a one hot representation them first class is represented as $\mathbf{y} = (1,0)^T$ and second class is represented as $\mathbf{y} = (0,1)^T$. If $\mathbf{y}$ is represented as scalar, then first class is $y = 1 \Leftrightarrow \mathbf{y} = (1,0)^T$ and second class is $y = 0 \Leftrightarrow \mathbf{y} = (0,1)^T$.

In order to make a classification with respect to some 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 0.1. 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 Datasets

We have already defined spaces $\mathcal{X}$ and $\mathcal{Y}$ above. However, we must define the data with which we are going to work. In this project we work with tuples of variables $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$. For example $\mathbf{x}$ can be a text document and $\mathbf{y}$ can be its label that will show if the document is relevant or not.

Considering the data $\tilde{\mathbf{X}}$ and its labels $\tilde{\mathbf{Y}}$ are from spaces $\mathcal{X}$ and $\mathcal{Y}$ respectively. These are the data that form a training set $(\tilde{\mathbf{X}} \times \tilde{\mathbf{Y}})$. After we train our model we can continue training our model with the data from $\mathbf{X} \in \mathcal{X}$. However it is important to understand that we have no labels for the set $\mathbf{X}$. We can ask for a help from an annotator that can give us those labels. We assume that getting labels needs some time and is very expensive. For example we can think that our model is not good enough and we want to ask a group of lawyers to classify if some more unlabeled documents are relevant or not. Data labeling will take both some time and will be quite costly.

In this work, set $\mathbf{X}$ will be used for the active learning section. We want to show that if we choose the documents for labeling from the set $\mathbf{X}$ smartly our models will start to show better scores faster then if we choose the data for labeling randomly.

Basing on the definition of the datasets above we can assume that $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$ are random variables. If both $\mathbf{x}$ and $\mathbf{y}$ are random variables than tuples $(\mathbf{x}, \mathbf{y})$ must be samples from a joint probability density function $p(\mathbf{x}, \mathbf{y})$. With the usage of conditional probability rule $p(\mathbf{x}, \mathbf{y})$ can be written as

$$p(\mathbf{x}, y) = p(y|\mathbf{x})p(\mathbf{x}) \tag{4}$$

where $p(\mathbf{x})$ is given and $p(y|\mathbf{x})$ is a pdf that we want to estimate.

2 Decision Theory for Supervised Learning

2.1 Decision Theory and Support Vector Machine Algorithm

In this subsection we 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 $\theta = (\mathbf{x}, y)$ where tuple $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ is parameters and $\Theta = \mathcal{X} \times \mathcal{Y}$ is a parameters' space. In this case we used $y \in \{0, 1\}$. Considering updated definitions, loss function (1) can be rewritten as

$$L = L(\mathbf{x}, y, \mathbf{w}, b). \tag{5}$$

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

$$L(\mathbf{x}, y, \mathbf{w}, b) = \max(0, 1 - y\hat{y}(\mathbf{x}, \mathbf{w}, b)) \tag{6}$$

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

On the basis of given data $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ we want to come up with a decision that will help us to predict labels of the data $\mathcal{X}$. 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 spaces $\mathcal{X}$ and $\mathcal{Y}$. Moreover we consider $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$ are random variables with joint probability density function $p(\mathbf{x}, y)$. We will also assume that $\forall i \in \{1, .., N\}$, $(\mathbf{x}_i, y_i) \in \tilde{\mathbf{X}} \times \tilde{\mathbf{Y}}$ are independent identically distributed. Probability density function $p(\mathbf{x}, y)$ can be written as

$$p(\mathbf{x}, y) = p(y|\mathbf{x})p(\mathbf{x}). \tag{7}$$

If we had those pdfs from (8) 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 $p(\mathbf{x}, y)$ can be approximated as

$$p(\mathbf{x}, y) = \frac{1}{N} \sum_{i=1}^{N} \delta(\mathbf{x} - \mathbf{x}_i, y - y_i) \tag{8}$$

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

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

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

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

$$\rho(\mathbf{x}_i, \mathbf{w}, b) = \frac{1}{N} \sum_{i=1}^{N} \max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i + b). \tag{9}$$

2.2 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 2.1. However in this case our decision is to finds estimate $\hat{y} = \hat{y}(\mathbf{x}, \mathbf{W}, \mathbf{b})$ of the probability density function $p(y|\mathbf{x})$ where $\mathbf{x}$ is the input data, $\mathbf{y} = y \in \{0, 1\}$ 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 $\hat{y}$. Same as in 2.1 we can define $(\mathbf{x}, y)$ are parameters of the loss function and $\mathcal{X} \times \mathcal{Y}$ is a parameters' space. For better understanding of the variety of loss functions we will use cross entropy loss function that is defined as

$$L(\mathbf{x}, y, \mathbf{w}, b) = -y \ln \big(\hat{y}(\mathbf{x}, \mathbf{w}, b)\big) - (1 - y) \ln \big((1 - \hat{y}(\mathbf{x}, \mathbf{w}, b)\big). \tag{10}$$

Continuing using assumptions from 1 we consider $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$ are random variables with joint probability density function $p(\mathbf{x}, y)$. With the usage of the given dataset where $\forall i \in \{1, .., N\}$, $(\mathbf{x}_i, y_i) \in \tilde{\mathbf{X}} \times \tilde{\mathbf{Y}}$ are independent identically distributed we can approximate $p(\mathbf{x}, y)$ as (8). Applying definition 0.1, expected loss for the algorithm based on a neural network function

is evaluated as

$$\begin{aligned}
\mathbb{E}_{\pi^*} L &= \int_{\mathcal{X}\times\mathcal{Y}} L(\mathbf{x}, y, \mathbf{w}, b) p(\mathbf{x}, y) d(\mathbf{x}, y), \\
&= -\int_{\mathcal{X}\times\mathcal{Y}} \Big(y \ln\big(\hat{y}(\mathbf{x}, \mathbf{w}, b)\big) + (1-y)\ln\big((1-\hat{y}(\mathbf{x}, \mathbf{w}, b)\big)\Big) \frac{1}{N}\sum_{i=1}^{N} \delta(\mathbf{x}-\mathbf{x}_i, y-y_i) d(\mathbf{x}, y), \\
&= -\frac{1}{N}\sum_{i=1}^{N} \Big(y_i \ln\big(\hat{y}_i(\mathbf{x}_i, \mathbf{w}, b)\big) + (1-y_i)\ln\big((1-\hat{y}_i(\mathbf{x}_i, \mathbf{w}, b)\big)\Big). \qquad (11)
\end{aligned}$$

2.3 Decision Theory and Naive Bayes Algorithm

Naive Bayes algorithm is a bit different to the algorithm based on Neural Networks and SVM. In the case of Naive Bayes we want to estimate $p(\mathbf{W}|\mathbf{x}, \mathbf{y})$ where $\mathbf{W} \in \mathcal{W}$ is set of parameters of a specific probability density function of $(\mathbf{x}, \mathbf{y})$, $\mathbf{x}$ is the input data, $\mathbf{y}$ is the label in a one hot representation assigned to $\mathbf{x}$. The reason why we look for an estimate of the $p(\mathbf{W}|\mathbf{x}, \mathbf{y})$ but not $p(\mathbf{y}|\mathbf{x}, \mathbf{w})$ is due to the fact that in the case of $p(\mathbf{y}|\mathbf{x}, \mathbf{W})$ we would get an expression where normalization constant would be dependent on the set of parameters $\mathbf{W}$. That fact would make our computations very complicated. 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 $p(\mathbf{W}|\mathbf{x}, \mathbf{y})$ as follows

$$p(\mathbf{W}|\mathbf{x}, \mathbf{y}) = \frac{p(\mathbf{y})p(\mathbf{x}|\mathbf{y}, \mathbf{W})p(\mathbf{W})}{\int_{\mathcal{W}} p(\mathbf{x}, \mathbf{y}|\mathbf{W})p(\mathbf{W})d\mathbf{W}} \qquad (12)$$

where $\mathcal{W}$ is a space of $\mathbf{W}$.

Naive Bayes method introduces very strong assumption in equation (12). This assumption says that features of vector $\mathbf{x} = (x_1, x_2, ..., x_n)^T$ are conditionally independent. As a result estimation of $p(\mathbf{W}|\mathbf{x}, \mathbf{y})$ can be approximated as

$$\tilde{p}(\mathbf{W}|\mathbf{x}, \mathbf{y}) = \frac{1}{Z} p(\mathbf{y})p(\mathbf{W}) \prod_{i=1}^{n} \big(p(x_i|y_1, \mathbf{w}_i)^{y_1} p(x_i|y_2, \mathbf{w}_i)^{y_2} \big), \qquad (13)$$

where $\mathbf{y} = (y_1, y_2)$ is a one-hot representation of $\mathbf{y}$, $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n)$, Z is a normalizing constant. We want to maximize probability $\tilde{p}(\mathbf{W}|\mathbf{x}, \mathbf{y})$. As a result, using 13, loss function L will be represented as

$$\begin{aligned}
L(\mathbf{y}, \mathbf{x}, \mathbf{w}) &= -\log\big(\tilde{p}(\mathbf{W}|\mathbf{x}, \mathbf{y}), \qquad (14) \\
&= \log(Z) - \log\big(p(\mathbf{y})\big) - \log\big(p(\mathbf{W})\big) - \sum_{i=1}^{n} \log\big(p(x_i|y_1, \mathbf{w}_i)^{y_1} p(x_i|y_2, \mathbf{w}_i)^{y_2}\big). \qquad (15)
\end{aligned}$$

Same as in 2.1 and 2.2 we will assume that we can approximate $p(\mathbf{x}, \mathbf{y})$ as 8. From this moment everything is ready for deriving expected loss function. Expected loss function for Naive Bayes

method is derived as

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

$$\int_{\mathcal{X}\times\mathcal{Y}} \Big(\xi(\mathbf{W},\mathbf{y}) - \sum_{i=1}^{n} \log\big(p(x_i|y_1,\mathbf{w}_i)^{y_1} p(x_i|y_2,\mathbf{w}_i)^{y_2}\big) \frac{1}{N}\sum_{j=1}^{N} \delta(\mathbf{x}-\mathbf{x}_j, \mathbf{y}-\mathbf{y}_j) d(\mathbf{x},\mathbf{y}),$$

$$\frac{1}{N}\sum_{j=1}^{N}\Big(\xi_j(\mathbf{W},\mathbf{y}_j) - \sum_{i=1}^{n}\log\big(p(x_{i,j}|y_{1,j},\mathbf{w}_i)^{y_{1,j}} p(x_{i,j}|y_{2,j},\mathbf{w}_i)^{y_{2,j}}\big)\Big) \tag{16}$$

where $\xi(\mathbf{W},\mathbf{y}) = \log(Z) - \log\big(p(\mathbf{y})p(\mathbf{W})\big)$ **and** $\xi_j(\mathbf{W},\mathbf{y}_j) = \log(Z) - \log\big(p(\mathbf{y}_j)p(\mathbf{W})\big)$**.**

Due to the fact that NB method theory is slightly different with 2.1 and2.2 we introduce here and example when

$$p(x_i|y_k,\mathbf{w}_i) = \mathcal{N}(w_{1,i}, w_{2,i}) \tag{17}$$

where $\mathbf{w}_i = (w_{1,i}, w_{2,i})$, $k \in \{0,1\}$ and $p(\mathbf{W})$ is a uniform distribution. Expected loss (16) for Gaussian likelihood case can be written as

$$\mathbb{E}_{\pi^*} L = \frac{1}{N}\sum_{j=1}^{N}\Big(\xi_j(\mathbf{W},\mathbf{y}_j) + \sum_{i=1}^{n}\big(y_{1,j}\big(\log(\frac{1}{\sqrt{2\pi}w_{2,i}}) + \frac{(x_{i,j}-w_{1,i})^2}{2w_{2,i}^2}\big) + y_{2,j}\big(\log(\frac{1}{\sqrt{2\pi}w_{2,i}}) + \frac{(x_{i,j}-w_{1,i})^2}{2w_{2,i}^2}\big)\big)\Big)$$

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

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

and

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

2.4 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.

2.4.1 Decision Tree

In this section we expect our decision three to give us an estimate $\hat{\mathbf{y}}(\mathbf{x},\mathbf{w}) \in \{(0,1)^T, (1,0)^T\}$ where $\mathbf{w}$ is a vector that describes tree (depth, branches, etc.), $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$. It is very important to mention that for different trees $\mathbf{w}$ can be different. Thus, for consistency we will assume that for all $\mathbf{w} \in \mathcal{W}$ exists upper bound, where $\mathcal{W}$ is redefined as a space of tree parameters. As a result we will make all $\mathbf{w}$ same length. If $\mathbf{w}$ has spare slots, they will be filled with zeros. Parameter space will be same as in 2.1-2.3, whereas action $a \in \mathcal{A}$ will be represented as $\mathcal{A} = \mathcal{W}$. In order to understand when our tree is optimal we can use zero-one loss function. Zero-one loss function is defined as

$$L(\mathbf{x},\mathbf{y},\mathbf{w}) = \begin{cases} 1, & \mathbf{y} \neq \hat{\mathbf{y}}(\mathbf{x},\mathbf{w}) \\ 0, & \mathbf{y} = \hat{\mathbf{y}}(\mathbf{x},\mathbf{w}) \end{cases}. \tag{18}$$

With the usage of the given data 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 (8). As a result, expected loss function for decision tree can be derived as

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

where $\sum_{i=1}^{N} L(\mathbf{x}_i, \mathbf{y}_i, \mathbf{w})$ is (18).

If we want to minimize expected loss we have to follow next steps. While construction decision tree we choose such feature $x_i \in (x_1, ..., x_n)^T = \mathbf{x}$ that will bring as the highest information about the system. This feature will form first layer, then we add another feature with the highest informational gain and construct second layer. Basing of this method we construct nodes and add more and more layers (branches).

In the following part we are going to work with a set of decision trees. For this purposes we will define our decision tree as $\hat{\mathbf{y}} = \big(T_1(\mathbf{x}, \mathbf{w}_l), T_2(\mathbf{x}, \mathbf{w}_l)\big)^T$ where index l represents set of parameters for $l-$th three and indices $1, 2$ represent first and second value of the one-hot vector.

2.4.2 Random Forest

With the usage of the given data $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ we want to come up with a decision that will help us to predict labels of the data $\mathcal{X}0$. We want to consider $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$ 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 (7).

With the usage of $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ we will make $\{1, ..., L\}$, $L \in \mathbb{N}$ sets where $\forall l \in L$, $\tilde{\mathbf{X}}_l \subset \tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}_l \subset \tilde{\mathbf{Y}}$. The data $\tilde{\mathbf{X}}_l$ and $\tilde{\mathbf{Y}}_l$ are created with random uniform sampling from $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$. We also want each subset to contain strictly 60% of the data from $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$. As a result parameter space for random forests will form tuples of sets $(\tilde{\mathbf{X}}_l, \tilde{\mathbf{Y}}_l)$. Basing on this theory we will construct L decision trees $\hat{y} = T(\mathbf{x}, \mathbf{w}_l)$, where $\mathbf{x} \in \tilde{\mathbf{X}}_l$. As a result for $l-$th decision tree

$$\mathbb{E}_{\pi^*} L = \frac{1}{N_l} \sum_{i=1}^{N_l} L(\mathbf{x}_{i,l}, \mathbf{y}_{i,l}, \mathbf{w}_l) \delta(\mathbf{x} - \mathbf{x}_{i,l}, \mathbf{y} - \mathbf{y}_{i,l}) \tag{19}$$

where $(\mathbf{x}_{i,l}, \mathbf{y}_{i,l}) \in (\tilde{\mathbf{X}}_l, \tilde{\mathbf{Y}}_l)$ and N_l is a number of the data in $\tilde{\mathbf{X}}_l$ and $\tilde{\mathbf{Y}}_l$. If we assume $\mathbf{w}_l$ as a random variable then L decision trees form samples from probability density function $p(\mathrm{y}|\mathbf{x}, \mathbf{w})$. In other words

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}_l) = T_1(\mathbf{x}, \mathbf{w}_l)^{y_1} T_2(\mathbf{x}, \mathbf{w}_l)^{y_2} \tag{20}$$

where label $\mathbf{y}$ is written as a one-hot representation $\mathbf{y} = (y_1, y_2)^T$. Thus, we can say that classification probability $p(\mathbf{y}|\mathbf{x})$ can be written as

$$p(\mathbf{y}|\mathbf{x}) = \int_{\mathbf{w}\in\mathcal{A}} p(\mathbf{y}|\mathbf{x}, \mathbf{w}) p(\mathbf{w}) d\mathbf{w}. \tag{21}$$

where $\mathcal{A}$ is an action space. With the usage of samples $\mathbf{w}_l$ we can approximate $p(\mathbf{y}|\mathbf{x})$ as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{L}\sum_{l=1}^{L} T_1(\mathbf{x}, \mathbf{w}_l)^{y_1} T_2(\mathbf{x}, \mathbf{w}_l)^{y_2} \tag{22}$$

where each decision tree $\big(T_1(\mathbf{x}, \mathbf{w}_l), T_2(\mathbf{x}, \mathbf{w}_l)\big)^T$ is constructed with the usage of (19).