EXAM 23.03.18

Question 1

1). Given a training set D and an hypotheses space H: $h \in H$ overfits training data if exists $h' \in H$ such that $error_D(h) < error_D(h')$ and $error_D(h) > error_D(h')$.

2). In decision tree is very probable to have solutions that are overfitted, for example if we have two different solutions for a problem and one tree is deeper than another for sure we have overfitting. We have two methods to reduce overfitting:

- Reduced error pruning: We split our training set into training set and validation set. We evaluate the impact on validation set of pruning each possible node and we greedily remove the node that most improves validation accuracy.
- Rule post pruning: We generate all the tree. We move our tree into a logical form and we compute the accuracy. We start pruning, if the accuracy increases we continue, otherwise we stop.

Question 2

1). The Naive Bayes classifier is based on the property of conditional independence. X is conditionally independent from Y given Z if $p(X, Y|Z) = p(X|Z)p(Y|Z)$.

Given a target function $f: X \to V$ with $V = \{V_1 \times V_2 \times \dots V_N\}$ knowing that each sample x can be represented as $\langle a_1, a_2 \dots a_N \rangle$

$$p(v|x, D) = p(v|a_1 \dots a_N, D) = \frac{p(a_1 \dots a_N|v, D)p(v|D)}{p(a_1 \dots a_N|D)}$$

$V_{NB} = \underset{v \in V}{\operatorname{argmax}}\ p(a_1 \dots a_N|v, D)p(v|D)$. Now we introduce the conditional independence hypothesis: $p(a_1 \dots a_N|v, D) = \prod_{n=1}^{N} p(a_n|v, D)$, each sample is mutually conditionally independent from another one. $V_{NB} = \underset{v \in V}{\operatorname{argmax}}\ p(v|D) \prod_{n=1}^{N} p(a_n|v, D)$.

2). The target function is $f: X \to Y$ with $X = \{title \times abstract \times author \times site\}$ and $Y = \{HL, KR, PL\}$. The dataset is $D = \{(d_i, y_i)_{i=1}^{n}\}$.

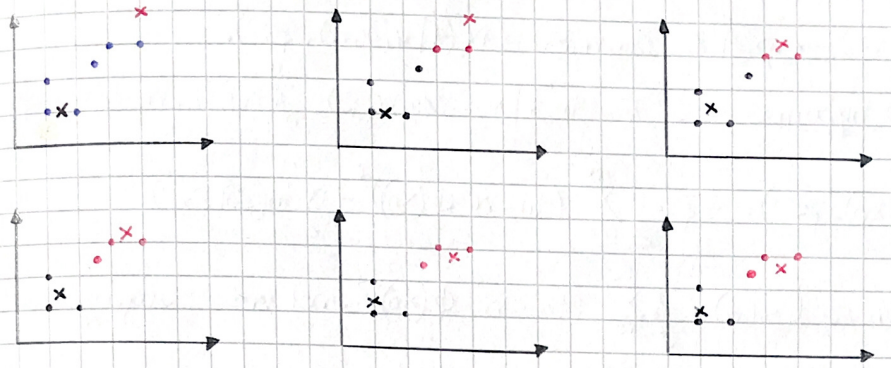
$V_{NB} = \underset{v \in V}{\operatorname{argmax}}\ p(v|D) \prod_{i=1}^{N} p(d_i|v, D)$ knowing that $p(d_i|v, D) = \prod_{i=1}^{length(d)} p(d_i = w_k|v, D)$

$$\hat{p}(d_i|v, D) = \frac{Tf_{ij} + 1}{Tf_j + 2}$$

Question 3

1). In an unsupervised learning task we have a target function $f: X \to Y$ and a dataset $D = \{(x_n)\}_{n=1}^{N}$. It is possible to see that in this case we don't have labels, so the dataset is composed only by input values.

2).



3). It is possible to see that initially one instance is classified as a point of the "blue centroid" but after the first iteration the instance will be classified as red.

Question 4

1). We compute the K-nearest neighbors of a new instance $x \notin D$.

We assign to x the most common label among the majority of neighbors.

$$p(c|x, D, K) = \frac{1}{K} \sum_{x_n \in N_K(x_n, D)} \mathbb{I}(t_n = c) \qquad \text{with } \mathbb{I}(e) = \begin{cases} 1 \text{ if } e \text{ is true} \\ 0 \text{ otherwise} \end{cases}$$

with $N_K(x_n, D)$ K-nearest neighbors of x_n.

Question 5

1). Backpropagation (BackProp) is an algorithm to compute the gradient, is not a learning algorithm and is used in ANNs. In this mode we propagate the gradient throw all the net. In the forward step we compute the values of the parameter of the functions while in the backward step we compute the gradient.

SGD is a learning algorithm that is used to find the minimum or the maximum of a function, is an iterative method and is an optimization strategy very common.

2). FORWARD STEP

Requires $W^{(l)}$ weights matrices, x input value, $b^{(l)}$ bias parameters and t target value.

$h^{(0)} = x$
for $k = 1 \dots l$ do:
  $a^{(k)} = b^{(k)} + W^{(k)} h^{(k-1)}$
  $h^{(k)} = f(a^{(k)})$
$h^{(l)} = y$
$J = L(t, y)$

BACKWARD STEP

$g \leftarrow \nabla_y J = \nabla_y L(t, y)$
for $k = l$ to 1:
  $g \leftarrow \nabla_{a^{(k)}} J = g \odot f'(a^{(k)})$
  $\nabla_{b^{(k)}} J = g$
  $\nabla_{W^{(k)}} J = g (h^{(k-1)})^T$
  $g \leftarrow \nabla_{h^{(k-1)}} J = (W^{(k)})^T g$

Question 6

1). We have a target function $f: X \to Y$ with $X \subseteq \mathbb{R}^d$ and $Y = \mathbb{R}$. Our model is $y(x,w) = \sum_{j=1}^{M} w_j \phi_j(x)$ that is linear in w and $\mathcal{D} = \{(x_n, t_n)\}_{n=1}^{N}$.

We know that $t = y(x,w) + \varepsilon$ with ε additive Gaussian noise. If we know that:

$p(\varepsilon|\beta) = \mathcal{N}(\varepsilon|0, \beta^{-1}) \to p(t|x_1 \dots x_N, w, \beta) = \mathcal{N}(t|y(x,w), \beta^{-1})$.

Now we use the iid hypothesis: $p(\{t_1 \dots t_N\}|x_1 \dots x_N, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n, w^T\phi(x_n), \beta^{-1}) =$

$$= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|w^T\phi(x_n), \beta^{-1}) = -\beta \cdot \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T\phi(x_n))^2 - \frac{N}{2} \ln(2\pi\beta^{-1})$$

The error function is $E_D(w) = \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T\phi(x_n))^2$ and $w^* = \operatorname{argmin}_w E_D(w)$

2). We can use least squares: $E_D(w) = \frac{1}{2}(t - \phi w)^T(t - \phi w)$ and $w^* = \phi^+ t$.

We can also use a sequential algorithm: $\hat{w} \leftarrow \hat{w} + \eta[t_n - w^T\phi(x_n)]\phi(x_n)$