Machine Learning II Exam from 05.08.2020

This is not official; it is not endorsed by the university or the lecturer etc.

120 minutes, no auxiliary tools allowed, $20 + 10 + 25 + 20 + 25 = 100$ points

1. Multiple choice ($4 \times 5 = 20$ Points)

Answer the following multiple choice questions. *There is only one good answer per question.* To mark an answer put an $\times$ in the $\square$ next to it. For each question, no or false answer is zero points, correct answer is full points.

(a) Locally linear embedding (LLE)

- embeds the data into a high-dimensional space for subsequent classification.
- learns a parametric mapping from the inputs to the outputs.
- preserves local structure of the data.
- is nonconvex and is subject to local minima.

(b) Which of the following is **True**: Canonical Correlation Analysis (CCA)

- finds the projection of one multivariate random variable that is maximally correlated.
- finds the projection of two multivariate random variables that are maximally correlated.
- finds which dimensions of a multivariate random variable that are maximally correlated.
- finds which dimension of two multivariate random variables are maximally correlated.

(c) Which of the following is **True**: Assuming a kernel $k(x, x')$, the *weakest* condition on this kernel for the support vector data description (SVDD) and the one-class SVM to produce the same decision boundary is

- $k(x_i, x_i) = 0$ for all i.
- $k(x_i, x_j) = 0$ for all i, j.
- $k(x_i, x_i) = \text{const}$ for all i. (lecture 6, slide 23)
- $k(x_i, x_j) = \text{const}$ for all i, j.

(d) A limitation of the weighted degree kernel $k(x, x') \coloneqq \sum_{\ell=1}^{L} \beta_\ell \sum_k \mathbb{1}(u_{k,\ell}(x) = u_{k,\ell}(x'))$ is

- it is not positive definite.
- it is computationally intractable.

- it is not robust to sequence misalignment.
- it does not take into account correlation between adjacent terms of the sequence.

2. Application of Machine Learning ($5 + 5 = 10$ Points)

Consider the task of reconstructing missing entries in some historical time series. Elements of the time series are valued between 0 and 10, with "?" at time steps where the symbol could not be recovered. An example of a possible sequence is

0	?	5	5	3	8	8	?	10	9	5	9	?	?	?	3	4	...

We would like to use machine learning to learn a model that can resolve the missing entries. We have collected $N = 1000$ sequences, each of them comprising between 100 to 500 time steps. Indicate:

(a) the name of an algorithm or method presented in ML2 that can solve this problem efficiently.

(b) the way the algorithm wold be applied, in particular, how to select and represent you data for training and prediction, and what objective to minimise.

(a) Structured prediction (kernel or neural networks).

(b)

3. One-Class SVM ($5 + 5 + 15 = 25$ Points)

The non-spherical version of one-class SVM is given by the optimisation problem

$$\min_{w,\xi,\rho} \frac{1}{2}\|w\|_2^2 - \rho + C\sum_{i=1}^{N}\xi \qquad \text{subject to} \qquad w^\mathsf{T}x_i \geq \rho - \xi_i \quad \text{and} \quad \xi_i \geq 0 \quad \forall i \in \{1,\dots,N\},$$

where $x_1, \dots, x_N \in \mathbb{R}^d$ are the training data. The condition for classifying a data point x as an outlier is then given by $w^\mathsf{T}x < \rho$.

(a) *Give* a geometric interpretation of the quantity $\frac{\rho}{\|w\|_2}$.

By slide 21 from lecture 6, $\frac{\rho}{\|w\|_2}$ is the smallest distance from the origin to the separating hyperplane. By minimising $\|w\|$ we push the hyperplane as close to the data points as possible.

(b) *Write* down the Lagrangian $L(w, \rho, \xi; a, b)$ of the constrained optimisation problem above, where a and b are vectors of Lagrange multipliers associated to each set of constraints.

$L(w, \rho, \xi; a, b) := \frac{1}{2}\|w\|_2^2 - \rho + C\sum_{i=1}^{N}\xi + \sum_{i=1}^{N} a_i(\rho - \xi_i - w^\mathsf{T}x_i) - \sum_{i=1}^{N} b_i\xi_i.$

(c) *Derive* the dual program for the one-class SVM. *Show* that it has the form

$$\min_a \frac{1}{2}\sum_{i,j=1}^{N} a_i a_j x_i^\mathsf{T} x_j \qquad \text{subject to} \qquad 0 \le a_i \le C \quad \forall i \in \{1,\dots,N\} \quad \text{and} \quad \sum_{i=i}^{N} a_i = 1$$

We have

$$\frac{\partial}{\partial w} L(w,\rho,\xi;a,b) = 0 \iff w = \sum_{i=1}^{n} a_i x_i$$

$$\frac{\partial}{\partial \rho} L(w,\rho,\xi;a,b) = 0 \iff 1 = \sum_{i=1}^{n} a_i$$

$$\frac{\partial}{\partial \xi_j} L(w,\rho,\xi;a,b) = C - a_j - b_j \stackrel{!}{=} 0 \quad \forall j \in \{1,\dots,N\}.$$

The dual problem thus is

$$\max_{a,b} \min_{w,\rho,\xi} L(w,\rho,\xi;a,b)$$

$$\text{subject to } \; w = \sum_{i=1}^{n} a_i x_i,\; 1 = \sum_{i=1}^{n} a_i,\; C - a_i - b_i = 0,\; a_i, b_i \ge 0 \quad \forall i \in \{1,\dots,N\},$$

which, by plugging in the primal variables, is equal to

$$\max_{a,b} \frac{1}{2}\| \sum_{i=1}^{n} a_i x_i \|_2^2 + \underbrace{\sum_{i=1}^{N} a_i \rho - \rho}_{=0} + \underbrace{\sum_{i=1}^{N} (C - a_i - b_i)\xi}_{=0} - \sum_{i=1}^{N} a_i \left(\sum_{j=1}^{n} a_j x_j \right)^\mathsf{T} x_i$$

$$\text{subject to} \quad 1 = \sum_{i=1}^{n} a_i,\; C - a_i - b_i = 0,\; a_i, b_i \ge 0 \quad \forall_{i=1}^{N},$$

which reduces to

$$\max_{a,b} -\frac{1}{2}\sum_{i,j=1}^{N} a_i a_j x_i^\mathsf{T} x_j \quad \text{subject to} \quad C - a_i \ge 0,\; a_i \ge 0 \quad \forall_{i=1}^{N},$$

which is

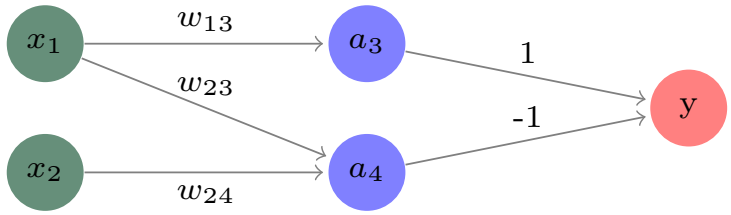
$$\min_{a,b} \frac{1}{2}\sum_{i,j=1}^{N} a_i a_j x_i^\mathsf{T} x_j \quad \text{subject to} \quad C \ge a_i \ge 0 \; \forall_{i=1}^{N}.$$

4. Neural Networks and Backpropagation ($5 + 5 + 10 = 20$ Points)

Let x_1 and x_2 be two observed variables. Consider the two-layer network that takes these two variables as input and builds the prediction y by computing iteratively:

$$z_3 \coloneqq w_{13} x_1, \quad z_4 \coloneqq w_{14} x_1 + w_{24} x_2, \quad a_3 \coloneqq \exp(z_3), \quad a_4 \coloneqq \exp(z_4), \quad y = a_3 - a_4.$$

(a) *Draw* the neural network graph associated to these computations.



We now consider the loss function $\ell(y,t) \coloneqq \frac{1}{2}(y-t)^2$, where t is a target variable that the neural network learns to approximate.

(b) Using the rules for backpropagation, compute the derivatives $\frac{\partial \ell}{\partial w_{13}}$, $\frac{\partial \ell}{\partial w_{14}}$ and $\frac{\partial \ell}{\partial w_{14}}$ required for gradient descent.

$$\frac{\partial \ell}{\partial w_{13}} = \frac{\partial \ell}{\partial y}\frac{\partial y}{\partial a_3}\frac{\partial a_3}{\partial z_3}\frac{\partial z_3}{\partial w_{13}} = (y-t)\cdot 1 \cdot a_3 \cdot x_1 = (y-t)a_3 \cdot x_1$$

and similarly $\frac{\partial \ell}{\partial w_{14}} = (t-y)a_4 \cdot x_1$ and $\frac{\partial \ell}{\partial w_{24}} = (t-y)a_4 \cdot x_2$.

(c) We now consider the activations a_3 and a_4 and produce them from the mixture coefficients

$$p_3 \coloneqq \frac{a_3}{a_3+a_4} \qquad \text{and} \qquad p_4 \coloneqq \frac{a_4}{a_3+a_4}.$$

We define the new loss function

$$\ell(p_3,p_4,t) \coloneqq -\log(p_3 f_3(t) + p_4 f_4(t)),$$

where f_3 and f_4 are some positive valued functions. Compute the derivative of the new loss function with respect to the variables z_3 and z_4. In your derivation, you can make use of the posterior probabilities

$$\pi_3 \coloneqq \frac{p_3 f_3(t)}{p_3 f_3(t) + p_4 f_4(t)} \qquad \text{and} \qquad \pi_4 \coloneqq \frac{p_4 f_4(t)}{p_3 f_3(t) + p_4 f_4(t)}.$$

$$\begin{aligned}\frac{\partial \ell}{\partial z_3} &= \frac{\partial \ell}{\partial p_3}\frac{\partial p_3}{\partial a_3}\frac{\partial a_3}{\partial z_3} + \frac{\partial \ell}{\partial p_4}\frac{\partial p_4}{\partial a_3}\frac{\partial a_3}{\partial z_3} = -\frac{\pi_3}{p_3}\frac{a_4}{(a_3+a_4)^2}a_3 - \frac{\pi_4}{p_4}\frac{-a_4}{(a_3+a_4)^2}a_3 \\ &= \frac{\pi_4}{p_4}p_4 p_3 - \frac{\pi_3}{p_3}p_4 p_3 = \pi_4 p_3 - \pi_3 p_4.\end{aligned}$$

and similarly $\frac{\partial \ell}{\partial z_3} = \pi_3 p_4 - \pi_4 p_3$.

5. Structured Kernels ($7 + 18 = 25$ Points)

Let two documents s and t be represented by the set of English words that compose them. For example:

$s =$ `set(['man', 'his', 'resting', 'the', 'car', 'been', 'has', 'in'])`,

$t =$ `set(['longer', 'is', 'table', 'the', 'book', 'on', 'no'])`,

Let W be a very large wet of *all* possible English words. The kernel for two documents s and t is defined as:

$$k(s,t) = \sum_{w \in W} \mathbb{1}_{\{w \in s \text{ and } w \in t\}} .$$

(a) *Implement* a function that computes the kernel for any pair of documents s and t. The implementation should be efficient (i.e. not iterate over all words in W).

```
def kernel(s,t):
    k = len(s.intersection(t))
    return k
```

(b) We would like to implement a rudimentary machine learning model that is based on this kernel. Our model learns the mean of the training data in feature space and predicts the squared distance of new data points to the mean. Considering a data set $x_1, \ldots, x_N$, with mean in feature space $m = \frac{1}{N} \sum_{i=1}^{N} \varphi(x_i)$, the squared distance of new data points x to the mean is given by

$$\left\| \varphi(x) - \frac{1}{N} \sum_{i=1}^{N} \varphi(x_i) \right\|_2^2 = k(x,x) - \frac{2}{N} \sum_{i=1}^{N} k(x, x_i) + \underbrace{\frac{1}{N^2} \sum_{i,j=1}^{N} k(x_i, x_j)}_{=:a} .$$

Implement the functions `fit` and `predict` below that receive some training and test data respectively (given as a list of documents).

```
class Dist2mean:
      def fit(self, Xtrain):
            self.N = len(Xtrain)
            kernelsum = 0
            for i in Xtrain:
                  for j in Xtrain:
                        kernelsum += kernel(i, j)
            self.a = (1.0 / (self.N * self.N)) * kernelsum
            self.training = Xtrain
      def predict(self, Xtest):
            Dtest = []
            for d in Xtest:
                  auto = kernel(d, d)
                  second = 0
                  for i in self.training:
                        second += kernel(d, i)
                  Dtest.append(auto - (2.0 / self.N) * second + self.a)
            return Dtest
```

The grey code was given.

Thanks to everyone contributing to this account of the exam and its solutions :)