Fondements de l'Apprentissage Machine (IFT 3395/6390)

Final Examn

Fall 2016

Professor: Pascal Vincent

Thursday, december 15th 2016
Duration: 2h45

Allowed documents: 4 sheets of paper.

First name:
Last name:
Code permanent:
IFT3395 or IFT6390:

Exam total is 100pts. Give brief but precise answers directly in the spaces left blank. **For the whole exam we suppose inputs $x \in \mathbb{R}^d$. Good luck!**

1 Graphical concepts (20 pts)

We learn parameters θ of a binary classification function $f_{\lambda,\theta}(x)$ which also has hyper-parameters λ. This function $f_{\lambda,\theta}(x)$ yields **an estimation of the probability that x belongs to class 1** (as opposed to class 0).

Let $\hat{R}$ the empirical risk associated to such a function on a data-set D of example pairs (x,t) with $t \in \{0,1\}$ indicating the class:

$$\hat{R}(f_{\lambda,\theta}, D) = \sum_{x,t \in D} L(f_{\lambda,\theta}(x), t)$$

Consider the following **"graphical concepts"** :

1. Decision boundary
2. Decision region of class 0
3. Decision region of class 1
4. Set of points misclassified by $f_{\lambda,\theta}$
5. Set of points correctly classified by $f_{\lambda,\theta}$
6. Error landscape (or cost landscape)
7. Learning (or training) curve

Consider the following **equations**, and to the left of each equation, write the number of the "graphical concept" to which it corresponds (*if* there is one in the above list).

$$\begin{aligned}
O(\theta) &= \hat{R}(f_{\lambda,\theta}, D_n) \\
\mathcal{A} &= \{x \in \mathbb{R}^d | f_{\lambda,\theta}(x) < 0.5\} \\
\hat{R}^*(\lambda) &= \min_\theta \hat{R}(f_{\lambda,\theta}, D_n) \\
\mathcal{B} &= \{(x,t) \in D_n | \, (f_{\lambda,\theta}(x) - 0.5)\,(t - 0.5) > 0\} \\
\\
\mathcal{C} &= \{x \in \mathbb{R}^d | f_{\lambda,\theta}(x) = 1 - f_{\lambda,\theta}(x)\} \\
\frac{\partial \hat{R}}{\partial \theta} &= \sum_{(x,t) \in D_n} \frac{\partial}{\partial \theta} L(f_{\lambda,\theta}(x), t) \\
\mathcal{E} &= \{x \in \mathbb{R}^d | f_{\lambda,\theta}(x) = 0.5\}
\end{aligned}$$

2 Capacity and training curves (30 pts)

1. Explain the differences between parameters and hyper-parameters in machine learning algorithms.

2. Most learning algorithms have hyperparameters that allow to control their "capacity". Explain in your own words this notion of capacity.

3. Name an algorithm and one of its hyper-parameters that, if increased, will **increase** the effective capacity of the algorithm.

4. Draw on a same graph the typical shapes of learning curves: both training curve (solid line) and validation curve (dashed line) that we will obtain as we vary that hyper-parameter (to be put on the horizontal axis).

   (a) Clearly label your axes.

   (b) On the horizontal axis, identify the optimal value of the hyper-parameter

   (c) On the horizontal axis, clearly identify the region corresponding to OVERFITTING and the region corresponding to UNDERFITTING.

   (d) On the horizontal axis, clearly identify the region corresponding to TOO LOW CAPACITY and the region corresponding to TOO HIGH CAPACITY.

5. Name an algorithm and one of its hyper-parameters that, if increased, will **decrease** the effective capacity of the algorithm.

6. Draw a graph with the learning curves for this algorithm and hyper-parameter (same instructions as previously)

7. Let D_{train} and D_{valid} the training set and test set (containing respectively $|D_{train}|$ and $|D_{valid}|$ examples). Write the pseudo-code that will allow to plot such learning curves (for trainset and validation) as a function of a hyper-parameter that we will call λ. Make sure to clearly distinguish between training examples and validation examples. Define all your notations (in particular for the training procedure).

3 Neural networks and gradient back-propagation (30 pts)

A special kind of neural network called an *auto-encoder* receives an input $x \in \mathbb{R}^d$ (column ector), has a hidden layer of size k, and computes an output as follows:

$$r(x) = Vs(Ux + b)) + c$$

where U and V are weight matrices, b and c are bias vectors, and s is a scalar function applied elementwise at the level of the hidden layer (e.g. s could be a sigmoïd or a tanh, or something else).

Output $r(x)$ is called a "reconstruction" of input x and has the same dimension as x.

Such a network is traditionally trained on the objective of reconstructing its input well, by minimizing a loss such as the squared error between an input and its reconstruction: i,e. minimizing $L(r, x) = \|r - x\|^2$.

Note that there is no "target" other than the input, it is thus unsupervised learning.

1. What can such a network be used for?

2. Specify the set θ of its parameters, and the *dimensions* of each.

3. Write the equation of the empirical risk that training of such a network will try to minimize on a training set $D_n = \{x^{(1)}, \ldots, x^{(n)}\}$.

4. Write the high-level equation for updating the parameters θ of such a network via batch gradient descent on this empirical risk (with a learning rate ϵ).

5. Write the high-level equation for updating the parameters θ of such a network via stochastic gradient descent on this empirical risk (with a learning rate ϵ).

6. Express $\frac{\partial L(r,x)}{\partial r}$ as a function of x and r.

7. Let us consider stochastic gradient descent, where we update parameters after having seen but one example. We use the efficient technique of *back-propagation* to compute the gradients for this example. Computations can be decomposed in 3 successive phases: a) **forward_propagation** phase computes the reconstruction as a function of the input, as well as the loss (reconstruction error), b) **back_propagation** phase computes the gradients as such, and c) **parameter_update** phase where parameters are updated using the gradients we just computed. Below, we scrambled the order of all the operations in these 3 phases. Indicate, ***in the right order***, among the below list, the operations for each of the phases. E.g.: forward_propagation: 14, 3, 10, 7, 2, 1 (of course, this is not the correct answer!).

   (a) Operations of **forward_propagation** phase:

   (b) Operations of **back_propagation** phase:

   (c) Operations of **parameter_update** phase:

List of operations [1]:

1. $\frac{\partial L}{\partial V} = \frac{\partial L}{\partial r} h^T$
2. $b \leftarrow b - \epsilon \frac{\partial L}{\partial b}$
3. $r = Vh + c$
4. $c \leftarrow c - \epsilon \frac{\partial L}{\partial c}$
5. $U \leftarrow U - \epsilon \frac{\partial L}{\partial U}$
6. $\frac{\partial L}{\partial c} = \frac{\partial L}{\partial r}$
7. Compute $\frac{\partial L}{\partial r}$ (according to formula you gave in question 6).
8. $\frac{\partial L}{\partial U} = \frac{\partial L}{\partial a} x^T$
9. $a = Ux + b$
10. $\frac{\partial L}{\partial a} = \frac{\partial L}{\partial h} \odot s'(a)$ (where s' denotes the first derivative of function s and $\odot$ denotes elementwise product)
11. $V \leftarrow V - \epsilon \frac{\partial L}{\partial V}$
12. $\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a}$
13. $h = s(a)$
14. $L = \|r - x\|^2$
15. $\frac{\partial L}{\partial h} = V^T \frac{\partial L}{\partial r}$

4 Number of parameters (20 pts)

Machine learning algorithms, during their *training* phase, learn (compute, find, optimize or memorize) a certain number of *parameters*, that will later be used to compute predictions on new test points. These parameters are composed of a certain number of scalars (real value numbers). E.g.: if parameters are a 20×30 matrix and a vector of size 10, then the total number of scalar parameters learned is 610. For each algorithm and case listed below, write **to the left of its question number** the **total number of scalar parameters** learned or memorized (needed when making a prediction for a new test point). Exclude hyper-parameters from this count.

We consider the following problem dimensions: a training set containing **800** examples in **dimension 10** (10 features, $x \in \mathbb{R}^{10}$), except if a sub-question specifies it differently.

1. Note: here we adopted the convention that the partial derivative of a scalar with respect to a vector or a matrix has the same dimensions as that vector or matrix.

1. Simple regression problem; linear (actually affine) regression algorithm.
2. Binary classification problem (2 classes); perceptron algorithm
3. Binary classification problem (2 classes); logistic regression algorithm
4. Binary classification problem (2 classes); linear SVM algorithm
5. Binary classification problem (2 classes); kernel SVM algorithm (with non linear kernel)
6. Regression problem with $x \in \mathbb{R}$ (1 dimension); polynomial regression algorithm with polynomial of degree 2.
7. Regression problem with $x \in \mathbb{R}^2$ (2 dimensions); polynomial regression algorithm with polynomial of degree 2.
8. Regression problem with $x \in \mathbb{R}^4$; Histogram with each dimension subdivided in 10 intervals (specified in advance by hyper-parameters).
9. Density estimation problem (with $x \in \mathbb{R}^{10}$). Single *isotropic Gaussian*
10. Density estimation problem. Single *Gaussian* with *diagonal covariance*
11. Density estimation problem. Single *Gaussian* with full covariance
12. Multiclass classification problem (3 classes); Bayes classifier using *diagonal Gaussians* (with no other constraint or parameter sharing. Be careful not to forget anything!)
13. Density estimation problem; Parzen density estimator with an isotropic Gaussian kernel whose variance has been fixed in advance (as a hyper-parameter), trained on a data set containing 800 examples (think of what is necessary to keep in memory to make a prediction on a test point)
14. Multiclass classification problem (3 classes); Parzen window classifier with an isotropic Gaussian kernel whose variance has been fixed in advance.
15. Multiclass classification problem (3 classes); Classical binary decision tree with 5 decision nodes and 6 leaves (that only need to emit the majority class in their region).
16. k-means algorithm with k=3
17. Principal Component Analysis (PCA) producing the 5 leading principal components.
18. Binary classification problem (2 classes): Adaboost using Perceptrons, after 5 iterations.
19. Multiclass classification problem (3 classes); neural network of type MLP (multilayer perceptron) with one hidden layer of 5 neurons.
20. Density estimation problem. Mixture of 3 diagonal Gaussians.