IFT-6390 Fundamentals of Machine Learning
Professor: Ioannis Mitliagkas
Students: Gustavo Patiño, Laura Sánchez

Homework 2 - Theoretical part

- This homework must be done and submitted to Gradescope and can be done in groups of at most 2 students. You are welcome to discuss with students outside of your group but the solution submitted by a group must be its own. Note that we will use Gradescope's plagiarism detection feature. All suspected cases of plagiarism will be recorded and shared with university officials for further handling.

- You need to submit your solution as a pdf file on Gradescope using the homework titled `(6390: GRAD) Theoretical Homework 2`.

1. **Bias-Variance decomposition** [2 points]

   Consider the following data generation process: an input point x is drawn from an unknown distribution and the output y is generated using the formula
   $$y = f(x) + \epsilon,$$
   where f is an unknown deterministic function and $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$. This process implicitly defines a distribution over inputs and outputs; we denote this distribution by p.

   Given an i.i.d. training dataset $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ drawn from p, we can fit the hypothesis h_D that minimizes the empirical risk with the squared error loss function. More formally,
   $$h_D = \underset{h \in \mathcal{H}}{\arg\min} \sum_{i=1}^{n} (y_i - h(x_i))^2$$
   where $\mathcal{H}$ is the set of hypotheses (or function class) in which we look for the best hypothesis/function.

The expected error[1] of h_D on a fixed data point (x', y') is given by $\mathbb{E}[(h_D(x') - y')^2]$. We will show that this error can be decomposed as a function of two meaningful terms:

- The <u>bias</u>, which is the difference between the expected value of hypotheses at x' and the true value $f(x')$. Formally,

$$bias = \mathbb{E}[h_D(x')] - f(x')$$

- The <u>variance</u>, which is how far hypotheses learned on different datasets are spread out from their mean $\mathbb{E}[h_D(x')]$. Formally,

$$variance = \mathbb{E}[(h_D(x') - \mathbb{E}[h_D(x')])^2]$$

Show that the expected prediction error on (x', y') can be decomposed into a sum of 3 terms: $(bias)^2$, *variance*, and a *noise* term involving ϵ. You need to justify all the steps in your derivation.

Solution:

We want to calculate $\mathbb{E}[(h_D(x') - y')^2]$, that means, how far is the test point from the real distribution, $y'(x) = f(x') + N(\mu, \sigma^2)$.
We will need to use this definitions:

$$Var[h_D(x')] = \mathbb{E}[(h_D(x'))^2] - (\mathbb{E}[h_D(x')])^2 \Leftrightarrow$$

$$\Leftrightarrow \mathbb{E}[(h_D(x'))^2] = Var[h_D(x')] + (\mathbb{E}[h_D(x')])^2$$

Also,

$$\mathbb{E}[y'] = \mathbb{E}[f(x') + \epsilon] = f(x')$$

$$Var[y'] = \mathbb{E}[(y')^2] - (\mathbb{E}[y'])^2 \Leftrightarrow$$

$$\Leftrightarrow \mathbb{E}[(y')^2] = Var[y'] + (\mathbb{E}[y'])^2 = \mathbb{E}[(y' - \mathbb{E}[y'])^2] + (\mathbb{E}[y'])^2 = \mathbb{E}[(y' - f(x'))^2] + (f(x'))^2$$

Developing, the expected error, using the definitions above, and assuming that h_D and y' are independent, we can get:

$$\mathbb{E}[(h_D(x') - y')^2] = \mathbb{E}[h_D^2(x') - 2h_d(x')y' + y'^2] =$$

$$= \mathbb{E}[h_D^2(x')] + \mathbb{E}[y'^2] - 2\,\mathbb{E}[h_D(x')]\,\mathbb{E}[y'] =$$

[1] Here the expectation is over random draws of the training set D of n points from the unknown distribution p. For example (and more formally): $\mathbb{E}[(h_D(x')] = \mathbb{E}_{(x_1,y_1)\sim p} \cdots \mathbb{E}_{(x_n,y_n)\sim p}\, \mathbb{E}[(h_{\{(x_1,y_1),\ldots,(x_n,y_n)\}}(x')]$.

$$= Var[h_D] + (\mathbb{E}[h_D(x')])^2 + \mathbb{E}[(y' - f(x'))^2] + (f(x'))^2 - 2\,\mathbb{E}[h_D(x')]f(x') =$$

And reordering the expression, we get:

$$\mathbb{E}[(h_D(x') - y')^2] = \mathbb{E}[(h_D(x') - \mathbb{E}[h_D(x')])^2] + (\mathbb{E}[h_D(x')])^2 + f(x')^2 - 2\,\mathbb{E}[h_D(x')]f(x') +$$

$$+ \mathbb{E}[(y' - f(x'))^2] = \mathbb{E}[(h_D(x') - \mathbb{E}[h_D(x')])^2] + [\mathbb{E}[h_D(x')] - f(x')]^2 + \mathbb{E}[(y' - f(x'))^2] =$$

$$= Variance + Bias + Noise$$

2. **Feature Maps**

   [8 points]

   In this exercise, you will design feature maps to transform an original dataset into a linearly separable set of points. For the following questions, if your answer is '*yes*', write the expression for the proposed transformation; and if your answer is '*no*', write a brief explanation. You are expected to provide explicit formulas for the feature maps, and these formulas should only use common mathematical operations.

   (a) [2 points] Consider the following 1-D dataset (Figure 1). Can you propose a 1-D transformation that will make the points linearly separable?

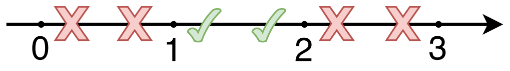

Figure 1:

Solution:
Choosing the point in the middle of the two green point, and then using a quadratic transformation, centered in this point, we can get the parable, that separates the green and red points, just by an horizontal line, between the images of the green ones, and the red ones. That is, let be, $x_{mid} = \dfrac{x_{green1} + x_{green2}}{2}$, the transformation could be $f(x_i) = (x_i - x_{mid})^2$, with x_i each point to classify. And the horizontal line, is $f(x_{green}) < y < f(x_{redmin})$, where, x_redmin makes reference to the point red closer to 2, and/or the red point closer to 1, because they will have the minimum image under the transformation.

(b) [2 points] Consider the following 2-D dataset (Figure 2). Can you propose a 1-D transformation that will make the data linearly separable?

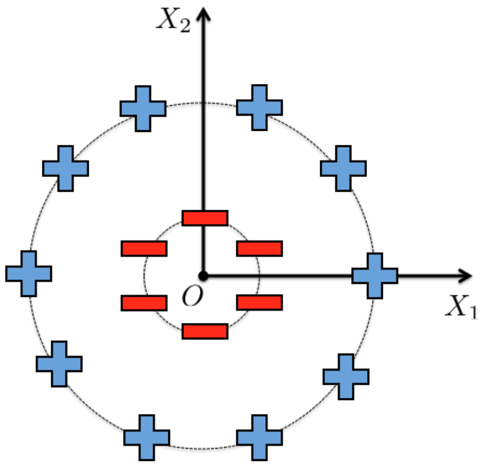

Figure 2:

Solution:
Here we will use the transformation to polar coordinates. That means: $r_i = \sqrt{x_i^2 + y_i^2}$. Then, each point will be positioned in the point on the real straight, that represents the length of the radius. The separation in 1D will be only a point in the middle of both radius.

(c) [4 points] Using ideas from the above two datasets, can you suggest a 2-D transformation of the following dataset (as shown in Figure 3) that makes it linearly separable? If '*yes*', also provide the kernel corresponding to the feature map you proposed.
Solution:
Using the previous transformations, first we transform to the polar coordinates, getting the points on the real straight, and then, we apply the transformation in (a), but, here r_{mid} will be the middle point of the first transformations of the red points. So, if the radius for the blue points, is like seems in the graph, the middle distance among the red ones, the image of the blue point, will be 0, that means, the blue point, will be the vertex of the parable. And again we will use the horizontal line that separates

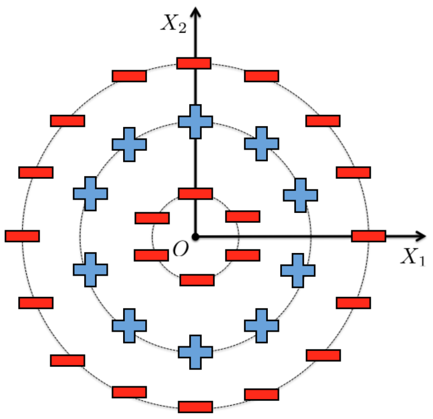

Figure 3:

images, for blue, and the red ones.
This is, $r_i = \sqrt{x_i^2 + y_i^2} \quad i = 1, 2, 3$, then we will have, r_1, r_3 as red points transformations, and r_2 as blue point transformation, with $r_1 < r_2 < r_3$. Finally, $f(r_i) = (r_i - r_{mid})^2$ and they will be separated for an horizontal line, that is $f(r_2) < y < f(r_1)$ and $f(r_1) = f(r_3)$

3. **Optimization** Optimisation [10 points]

Assume a quadratic objective function of the form:

$$f(x) = \frac{1}{2}x^T Ax + x^T b + a,$$

where $x \in \mathbb{R}^d$, $b \in \mathbb{R}^d$, $a \in \mathbb{R}$ and A is a $d \times d$ symmetric, positive definite matrix. This means that the matrix A admits the eigendecomposition $A = U\Lambda U^T$, where $U \in \mathbb{R}^{d\times d}$ is an orthonormal matrix and $\Lambda \in \mathbb{R}^{d\times d}$ is a diagonal matrix. The i-th column vector of U, denoted by $u_i \in \mathbb{R}^d$, represents the i-th eigenvector of A. The i-th diagonal element of Λ, denoted by $\lambda_i \in \mathbb{R}$, represents the i-th eigenvalue of A. We will assume here that all eigenvalues are unique. Furthermore, without loss of generality, the eigenvectors and eigenvalues in the decomposition can be ordered in such a way that

$$\lambda_1 > \lambda_2 > \ldots > \lambda_d > 0.$$

(a) Find all of the stationary points of $f(x)$ analytically, i.e. through a closed-form expression (Justify).

Solution:

We need to do the first derivative for the quadratic expression, that is:

$$\frac{\partial f(x)}{\partial x} = \frac{1}{2}(x^T A^T + x^T A) + b^T + 0 = \frac{1}{2}(x^T(A^T + A)) + b^T = 0$$

Like A is symetric, then $A = A^T$ and the final expression remains:

$$\frac{1}{2}(x^T(2A)) + b^T = 0 \Leftrightarrow x^T A = -b^T \Leftrightarrow$$
$$A^T x = -b \Leftrightarrow Ax = -b \Leftrightarrow x^* = -A^{-1}b$$

If we want to express using the eigenvalue descomposition, then,

$$x^* = -U\Lambda^{-1}U^T b$$

(b) Which of those stationary points are minima, maxima and saddle-points? Give a mathematically rigorous explanation why.

Solution:

We need to calculate the second derivative, and analize the sign. Then:

$$\frac{\partial \frac{\partial f(x)}{\partial x}}{\partial x} = \frac{\partial(x^T A + b^T)}{\partial x} = A^T = A$$

by definition of A this is positive defined, so, x^* are minimum.

(c) Find the location, x^*, and value, $f(x^*)$, of the global minimum.
Solution:
The global minimum is the point $(x^*, f(x^*)) = (-U\Lambda^{-1}U^T b, f(x^*))$, then, we need to calculate $f(x^*)$. Replacing in the formula, we get:

$$f(x^*) = \frac{1}{2}(-U\Lambda^{-1}U^T b)^T A(-U\Lambda^{-1}U^T b) + (-U\Lambda^{-1}U^T b)^T b + a =$$

$$= \frac{1}{2}b^T U\Lambda^{-1}U^T AU\Lambda^{-1}U^T b - b^T U\Lambda^{-1}U^T b + a =$$

$$= \frac{1}{2}b^T U\Lambda^{-1}U^T b - b^T U\Lambda^{-1}U^T b + a =$$

$$= a - \frac{1}{2}b^T U\Lambda^{-1}U^T b = a - \frac{1}{2}b^T A^{-1} b$$

Then the position for the minimum is:

$$(A^{-1}b, a - \frac{1}{2}b^T A^{-1} b)$$

Observation: $(U\Lambda^{-1}U^T)^T = (U^T)^T(\Lambda^{-1})^T U^T$, and like Λ is diagonal, then also, its inverse, and then $(\Lambda^{-1})^T = \Lambda^{-1}$

(d) Find the gradient of $f(x)$ at some point x. What are the dimensions of the gradient?
Solution:
For any point x_k, the gradient is: $\bigtriangledown f(x_k) = A^T x_k + b^T$ with $\bigtriangledown f(x_k) \in \mathbb{R}^d$

(e) Show how the gradient descent update rule looks like in this case by substituting $f(x)$ with its quadratic form above. Use the following notation: x_0 represents our point at initialization, x_1 represents our point after one step, etc.
Solution:
Let be:

$$x_{k+1} = x_k - \eta \bigtriangledown f(x_k) = x_k - \eta A^T x_k - \eta b = (I - \eta A^T)x_k - \eta b$$

Doing the iterations:

$$x_1 = (I - \eta A^T)x_0 - \eta b$$

$$x_2 = (I - \eta A^T)x_1 - \eta b = (I - \eta A^T)[(I - \eta A^T)x_0 - \eta b] - \eta b =$$

$$= (I - \eta A^T)^2 x_0 - (I - \eta A^T)\eta b - \eta b$$

$$x_3 = (I-\eta A^T)x_2 - \eta b = (I-\eta A^T)[(I-\eta A^T)^2 x_0 - (I-\eta A^T)\eta b - \eta b] - \eta b =$$

$$= (I - \eta A^T)^3 x_0 - \sum_{j=0}^{2} (I - \eta A^T)^j \eta b$$

So, doing induction until k, we get:

$$x_k = (I - \eta A^T)^k x_0 - \sum_{j=0}^{k-1} (I - \eta A^T)^j \eta b$$

This summatory corresponds to a geometric serial, then we can replace the value of the finity summatory, for the value of that one, that is,

$$\sum_{j=0}^{k-1} (I - \eta A^T)^j \eta b = (I - (I - \eta A^T)^k) A^{-1} b$$

then, the result for x_k is:

$$x_k = (I-\eta A^T)^k x_0 - (I-(I-\eta A^T)^k)A^{-1}b = (I-\eta A^T)^k x_0 + ((I-\eta A^T)^k - I)A^{-1}b$$

(f) Consider the <u>squared distance from optimum</u>, $d(x_k) = \|x_k - x^*\|_2^2$. Find an exact expression (equality) of $d(x_k)$ that only depends on x_0 (not on other iterates x_i for $i > 0$), the number of iterations, k, as well as the eigenvectors, u_i, and eigenvalues, λ_i of A.
Solution:

$$d(x_k) = \|(I - \eta A^T)^k x_0 + ((I - \eta A^T)^k - I)A^{-1}b - (-A^{-1}b)\|_2^2 =$$

$$= \|(I - \eta A^T)^k x_0 + ((I - \eta A^T)^k - I)A^{-1}b + A^{-1}b)\|_2^2$$

$$= \|(I - \eta A^T)^k x_0 + ((I - \eta A^T)^k - I + I)A^{-1}b)\|_2^2$$

$$== \|(I - \eta A^T)^k x_0 + ((I - \eta A^T)^k)A^{-1}b)\|_2^2$$

$$= \|(I - \eta \sum_{i=1}^{d} u_i^T \lambda_i u_i)^k x_0 + ((I - \eta \sum_{i=1}^{d} u_i^T \lambda_i u_i)^k (\sum_{i=1}^{d} u_i^T \lambda_i u_i)^k b)\|_2^2$$

(g) Prove that there exist some assumptions on the hyperparameters of the algorithm, under which the sequence $d(x_k)$ converges to 0 as k goes to infinity. What are the exact necessary and sufficient conditions on the hyperparameters in order for $d(x_k)$ to converge to 0?

(h) The distance that you computed above is said to converge to 0 at an <u>exponential rate</u> (some other research communities use the term <u>linear rate</u> for the same type of convergence). We often care about the <u>asymptotic rate</u> of convergence, defined for this squared distance as

$$\rho = \exp\left(\lim_{k\to\infty} \frac{1}{2k} \ln d(x_k)\right),$$

where $\ln(\cdot)$ denotes the natural logarithm. Keep in mind that this rate depends on both the objective function, but also on the choice of hyperparameters.

Find an expression of ρ that only depends on the eigenvalues of A and the hyperparameter values.

(i) Prove that, for any choice of hyperparameter values, there exist constants $k_0 \geq 0$ and $C > 0$ such that

$$d(x_k) \leq C\rho^{2k}, \quad \forall k > k_0.$$

(j) Based on the above, we gather that in order to get fast convergence, we need a small ρ value. Find a value for the hyperparameter(s) of gradient descent that achieves the fastest asymptotic convergence rate possible.

4. **Least Squares Estimator and Ridge Regression** [10 points]

(a) In the problem of linear regression, we are given n observations $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where each input $\mathbf{x}_i$ is a d-dimensional vector. Our goal is to estimate a linear predictor $f(.)$ which predicts y given $\mathbf{x}$ according to the formula

$$f(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\theta}, \tag{1}$$

Let $\mathbf{y} = [y_1, y_2 \ldots y_n]^\top$ be the $n \times 1$ vector of outputs and $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n]^\top$ be the $n \times d$ matrix of inputs. One possible way to estimate the parameter $\boldsymbol{\theta}$ is through minimization of the sum of squares. This is the least squares estimator:

$$\arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2. \tag{2}$$

i. Show that the solution of this minimization problem is given by

$$\boldsymbol{\theta}^* = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}.$$

Solution:
We need to compute,

$$\frac{\partial(\ \|\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\|_2^2)}{\partial\theta} = 0 \Leftrightarrow -2X^T(y-X\theta) = 0 \Leftrightarrow \tag{3}$$

$$\Leftrightarrow -2X^T(y-X\theta) = 0 \Leftrightarrow X^Ty - X^TX\theta = 0 \Leftrightarrow \tag{4}$$

$$\Leftrightarrow X^Ty = (X^TX)\theta^* \Leftrightarrow \theta^* = (X^TX)^{-1}X^Ty \tag{5}$$

ii. When will the matrix $\mathbf{X}^\top\mathbf{X}$ be invertible and when will it be non-invertible? Give your answer in terms of properties of the dataset.
Solution:
This matrix will be invertible if it's determinant is different than zero, that means, all the vectors are linearly independent that is, the matrix is orthogonal.

(b) A variation of the least squares estimation problem known as ridge regression considers the following optimization problem:

$$\arg\min_{\boldsymbol{\theta}} \|\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2 \tag{6}$$

where $\lambda > 0$ is a regularization parameter. The regularizing term penalizes large components in $\boldsymbol{\theta}$ which causes the optimal $\boldsymbol{\theta}$ to have a smaller norm.

i. Derive the solution of the ridge regression problem. Do we still have to worry about the invertibility of $\mathbf{X}^\top\mathbf{X}$?
Solution:

Same idea, but adding the regularization, that is:

$$\frac{\partial(\|\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2)}{\partial\theta} \Leftrightarrow -2X^T(y-X\theta) + 2\lambda\theta = 0 \Leftrightarrow \tag{7}$$

$$\Leftrightarrow -X^Ty + X^TX\theta + \lambda\theta = 0 \Leftrightarrow (X^TX + \lambda I)\theta = X^Ty \Leftrightarrow \tag{8}$$

$$\Leftrightarrow \theta^* = (X^TX + \lambda I)^{-1}X^Ty \Leftrightarrow \tag{9}$$

Now, like the matrix X^TX is followed by the λI, this determinant of $(X^TX + \lambda I)^{-1}$ will be always different of zero and invertible.

ii. Explain why the ridge regression estimator is likely to be more robust to issues of high variance compared with the least squares estimator.

Solution:

Because the value of the regularization constant, the values for the estimation of the output are closer to the mean, then the variance is controlled.

iii. How does the value of λ affect the bias and the variance of the estimator?

Solution:

When we reduce the variance, we increase the bias. That means that with the regualrization, we are controling the variance, but the bias is increasing.

5. **Leave one out cross-validation** **Validation croisée "leave-one-out"** [10 points]

Let $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training sample drawn i.i.d. from an unknown distribution p. Recall that leave-one-out cross-validation (LOO-CV) on a dataset of size n is the k-fold cross-validation technique we discussed in class for the special case where $k = n - 1$. To estimate the risk (a.k.a. the test error) of a learning algorithm using D, LOO-CV consists in comparing each output y_i with the prediction made by the hypothesis returned by the learning algorithm trained on all the data except the ith sample (x_i, y_i).

Formally, if we denote by $h_{D\setminus i}$ the hypothesis returned by the learning algorithm trained on $D \setminus \{(x_i, y_i)\}$, the leave-one-out error is given by

$$\text{error}_{LOO} = \frac{1}{n}\sum_{i=1}^{n} \ell(h_{D\setminus i}(x_i), y_i)$$

where ℓ is the loss function.

In this exercise, we will investigate some interesting properties of this estimator.

Leave-one-out is unbiased

(a) (Solution) The definition of the empirical risk $R(f)$ is done by an average loss over a finite data set D with n examples, so:

$$R(f, D) = \frac{1}{|D|}\sum_{i=1}^{n}(h_D(x_i) - y_i)^2 \tag{10}$$

where $h_D(x)$ is the hypothesis.

(b) (Solution) The expected value of the error over the examples in the data is done by

$$\mathop{\mathbb{E}}_{D\sim p}[\text{error}_{LOO}] = \mathop{\mathbb{E}}_{D\sim p}[L(f(x), y)] = \mathop{\mathbb{E}}_{D'\sim p}[(y - h'_D(x))^2]$$

where D' denote a dataset of size $n-1$, because we are taken the training set except one value. Since a value (x_i, y_i) was not used to fit the model the LOO provides an approximately unbiased estimate for the test error.

Complexity of leave-one-out We will now consider LOO in the context of linear regression where inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are d-dimensional vectors. Similarly to exercise 4, we use $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $\mathbf{y} \in \mathbb{R}^n$ to denote the input matrix and the vector of outputs.

(c) Assuming that the time complexity of inverting a matrix of size $m \times m$ is in $\mathcal{O}(m^3)$, what is the complexity of computing the solution of linear regression on the dataset D?

(d) Using $\mathbf{X}_{-i} \in \mathbb{R}^{(n-1)\times d}$ and $\mathbf{y}_{-i} \in \mathbb{R}^{(n-1)}$ to denote the data matrix and output vector obtained by removing the ith row of $\mathbf{X}$ and the ith entry of $\mathbf{y}$, write down a formula of the LOO-CV error for linear regression. What is the complexity of evaluating this formula?

(e) (Solution)

$$\text{error}_{LOO} = \sum_{i\neq k}(x_i^T \cdot w_k - y_i)^2$$

Taking the derivative and assuming it is equal to zero, we obtain

$$\sum_{i\neq k} x_i^T x_i w_k = \sum_{i\neq k} x_i y_i \tag{11}$$

In the same way if we take all the training set, it holds

$$\sum_i x_i^T x_i w = \sum_i x_i y_i \tag{12}$$

Eq.11 can be written as

$$\sum_i x_i^T x_i w_k - x_k^T x_k w_k = \sum_{i\neq k} x_i y_i - x_k y_k \tag{13}$$

Using Eq.12 in Eq.13

$$\sum_i x_i^T x_i w_k - \sum_i x_i^T x_i w = (h_k - y_k)x_k \tag{14}$$

where $h_k = x_k w_k$ is the hypothesis of the label k. Defining $X^T X \equiv \sum_i x_i^T x_i$ Eq.14 is equivalent to

$$(w_k - w) = (X^T X)^{-1}(h_k - y_k)x_k \tag{15}$$

Multiplying Eq.15 by x_k and remembering that $h_k = x_k w_k$ we arrive to

$$(h_k - x_k w) = x_k^T (X^T X)^{-1} x_k (h_k - y_k) \tag{16}$$

The last equation is equivalent to

$$(h_k - x_k w) = (h_k - y_k) - (x_k w - y_k) = x_k^T (X^T X)^{-1} x_k (h_k - y_k) \tag{17}$$

so

$$(h_k - y_k)[1 - x_k^T (X^T X)^{-1} x_k] = (x_k w - y_k)$$

arriving to

$$\frac{1}{n}\sum_{i=1}^{n}(h_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - w x_i}{1 - x_i^T (X^T X)^{-1} x_i}\right)^2 \tag{18}$$

In conclusion

$$error_{LOO} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - w x_i}{1 - x_i^T (X^T X)^{-1} x_i}\right)^2 \tag{19}$$

6. **Multivariate Regression** **Régression multivariée** [10 points]

We consider the problem of learning a vector-valued function $f : \mathbb{R}^d \to \mathbb{R}^p$ from input-output training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ where each $\mathbf{x}_i$ is a d-dimensional vector and each $\mathbf{y}_i$ is a p-dimensional vector. We choose our hypothesis class to be the set of linear functions from $\mathbb{R}^d$ to $\mathbb{R}^p$, that is function satisfying $f(\mathbf{x}) = \mathbf{W}^\top \mathbf{x}$ for some $d \times p$ regression matrix $\mathbf{W}$, and we want to minimize the squared error loss function

$$J(\mathbf{W}) = \sum_{i=1}^{n} \|\mathbf{W}^\top \mathbf{x}_i - \mathbf{y}_i\|_2^2 \tag{20}$$

over the training data.

Let $\mathbf{W}^*$ be the minimizer of the empirical risk:

$$\mathbf{W}^* = \arg\min_{\mathbf{W} \in \mathbb{R}^{d \times p}} J(\mathbf{W}).$$

(a) Solution

$$J(\mathbf{W}) = \sum_{i=1}^{n}\sum_{j=1}^{p}(\mathbf{W}_j^T\mathbf{x}_i - \mathbf{y}_i)^2$$

therefore the derivative in relation to $\mathbf{W}_j$ is

$$\frac{\partial J}{\partial \mathbf{W}_j} = 2\sum_{i=1}^{n}(\mathbf{W}_j^T\mathbf{x}_i - \mathbf{y}_i)\mathbf{x}_i$$

(b) Show that solving the problem from the previous question is equivalent to independently solving p independent classical linear regression problems (one for each component of the output vector), and give an example of a multivariate regression task where performing **independent** regressions for each output variables is not the best thing to do.

Solution:

Since we can get individual regressions, like a vector of regressions, that means:

$$y = x_1 w_1$$

,

$$y = x_2 w_2$$

$$....$$

$$y = x_d w_d$$

if all the variables are giving the same aproximation for the output, then we can get the output as a combination of regressions, such as:

$$y = (x_1, x_2, .., x_d)(w_1, w_2, .., w_d)^t = \sum_{i=1}^{d} x_i w_i$$

But, if the variables are independent, then the linear combination of all of them is not the best thing to do, because the combination of all becomes a different model to predict y

(c) Solution

- Make a singular value decomposition of the matrix of weights $\mathbf{W} = \mathbf{W}\sum\mathbf{B}$
- Random projection fo the matrix of data $\mathbf{X}$ to reduce the number of features
- Predictions are done by $\mathbf{Y} = \mathbf{X}\mathbf{B}$

1 Practical homework

Considering that we have m classes and p features the derivative of the term

$$\frac{1}{2}\sum_{j'=1}^{m}\|\mathbf{w}^{j'}\|_2^2$$

can be expressed as

$$\frac{\partial}{\partial w_k^j}\sum_{j=1}^{m}\sum_{k=1}^{p} w_k^j w_k^j = w_k^j \tag{21}$$

2 Practical homework

The hinge loss term (H) is defined by

$$H = \frac{C}{n}\sum_{i=1}^{n}\sum_{j'=1}^{m} max\{0, 1-(\mathbf{w}^{j'}\cdot\mathbf{x}_i)1\{y_i = j'\}\}^2$$

Assuming that the marginal

$$1.0 - w_j \cdot x_i 1_i^j > 0$$

the hing term satisfies

$$\frac{C}{n}\sum_{i=1}^{n}\sum_{j'=1}^{m}\left(1-\sum_{k=1}^{p} w_k^{j'} x_{ik} 1_i^{j'}\right)^2 = \frac{C}{n}\sum_{i=1}^{n}\sum_{j'=1}^{m}\left[1 - 2\sum_{k=1}^{p} w_k^{j'} x_{ik} 1_i^{j'} + \left(\sum_{k=1}^{p} w_k^{j'} x_{ik} 1_i^{j'}\right)^2\right] \tag{22}$$

so the derivative as function of w_k^j is

$$Grad_{kj} = \frac{2C}{n}\left(\sum_{i=1}^{n}\sum_{k=1}^{p} w_k^j x_{ik} x_{ik} - \sum_{i=1}^{n} x_{ik} 1_i^j\right) \tag{23}$$

4 Practical homework

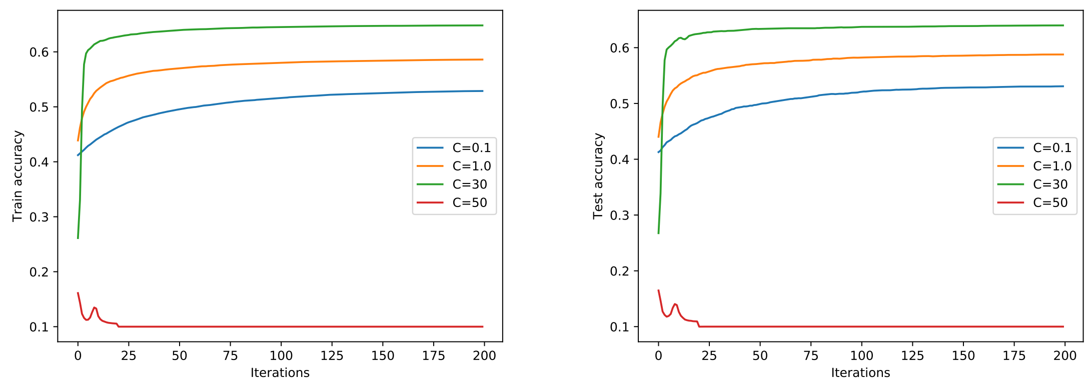

Figure 4: Accuracy for $C = 0.1, 1, 30, 50$, 200 epochs, learning rate of 0.001, and a minibatch size of 5000: Left: Train accuracy. Right: Test accuracy

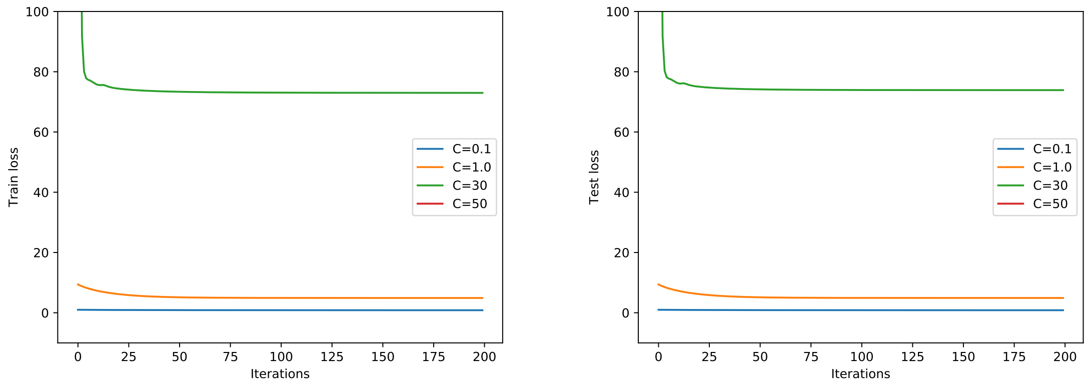

Figure 5: Loss for $C = 0.1, 1, 30, 50$, 200 epochs, learning rate of 0.001, and a minibatch size of 5000: Left: Train loss. Right: Test loss