COMP0078
Supervised Learning

Coursework 1

by

Mamoune Chaoui - 14050661 - mamoune.chaoui.20@ucl.ac.uk
Alexandra Proca - 20047328 - alexandra.proca.20@ucl.ac.uk

November 2020

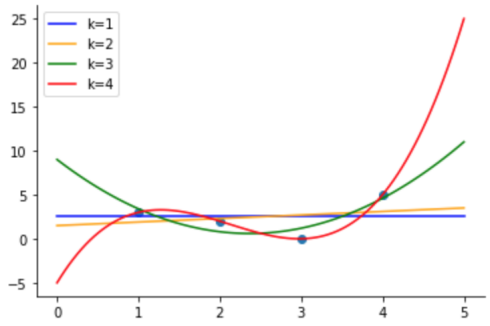
Department of Computer Science
University College London

Contents

1 PART I

1.1 Linear Regression

1. (a) The weights were calculated for the basis at each $k = 1, 2, 3, 4$ using $\boldsymbol{w} = (\boldsymbol{\phi}(\boldsymbol{x}))^T\boldsymbol{\phi}(\boldsymbol{x}))^{-1}\boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{y}$, where $\boldsymbol{\phi}(\boldsymbol{x})$ is the basis of $\boldsymbol{x}$, and the corresponding curves were plotted.

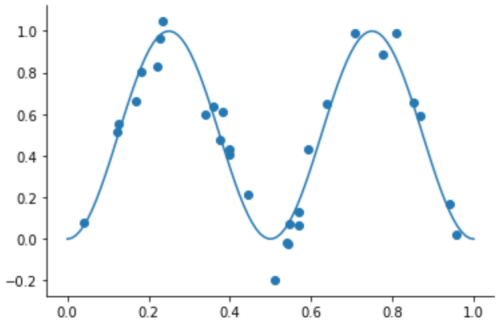


(b) The equations found for $k = 1, 2, 3, 4$ are:
$k = 1 : y = 2.5$
$k = 2 : y = 1.5 + 0.5x$
$k = 3 : y = 9 - 7.1x + 1.5x^2$
$k = 4 : y = -5 + 15.17x - 8.5x^2 + 1.33x^3$

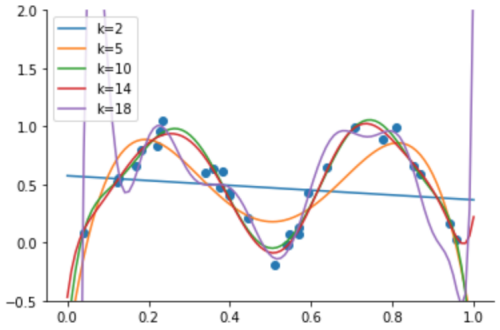
(c) The mean squared error (MSE) for each curve was calculated using $MSE = \frac{(\boldsymbol{y}-\boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{w})^T(\boldsymbol{y}-\boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{w})}{m}$

K	MSE
1	3.25
2	3.05
3	0.8
4	7.12e-25

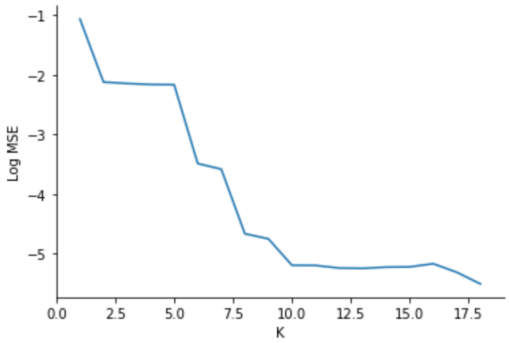
2. (a) The data was generated by sampling x_i uniformly at random from the interval [0,1] 30 times, sampling ϵ_i at random from $\mathcal{N}(0, 0.07)$ 30 times, and calculating $g_\sigma(x) = \sin^2(2\pi x) + \epsilon$. The corresponding points $S_{0.07,30} = (x_1, g_{0.07}(x_1)), \ldots, (x_{30}, g_{0.07}(x_{30}))$ were plotted, along with the plot of $y = \sin^2(2\pi x)$ in the range $0 \leq x \leq 1$. The data models the sin function with additional noise, as expected. **Note:** the following figures in **1.1** were found over a particular run and fluctuate slightly from run-to-run due to the randomization of initialization. Additionally, every following calculation of weights and MSE in **Part 1** utilize the aforementioned formulas implemented in corresponding functions.



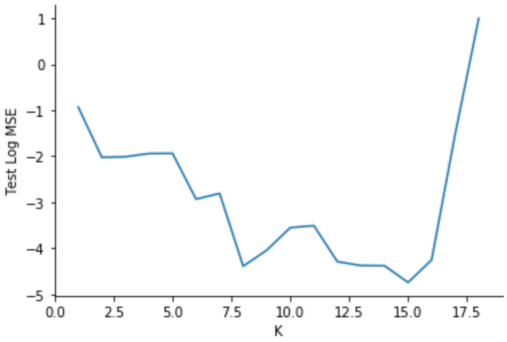
The weights were calculated for the basis $1, x, x^2, \ldots, x^k$ at each $k = 1, \ldots, 18$ and the corresponding curves were plotted, superimposed over a plot of the data points. As k increases, the weights adjust to fit the data with higher dimensionality of x, leading to overfitting with high k values.



(b) The weights were calculated for the basis at each $k = 1, \ldots, 18$ for the generated data and the corresponding MSE, $\text{tse}_k(S)$ were calculated. The natural log (ln) MSE at each k was plotted. The MSE decreases as k increases.

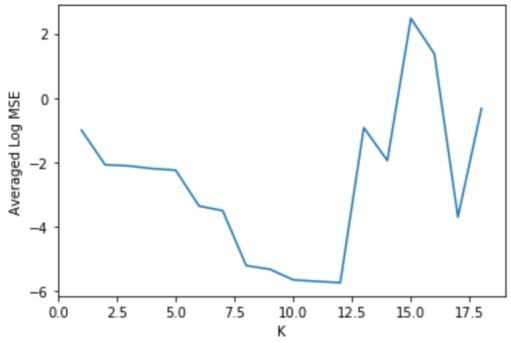


(c) A test set T: $T_{0.07,1000} = (x_1, g_{0.07}(x_1)), \ldots, (x_{1000}, g_{0.07}(x_{1000}))$ of a thousand points was generated. The test error $\text{tse}_k(S, T)$ was calculated by finding the MSE of the test set T on the polynomial dimension k fitted from training set S. The ln MSE at each $k = 1, \ldots, 18$ was plotted. The MSE decreases at first, but later increases as the model overfits the training data and performs worse on the test data with larger k.

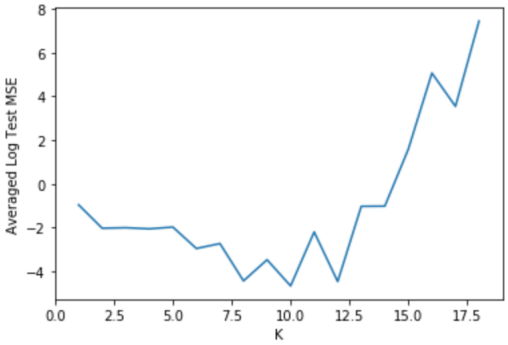


(d) Training data $S_{0.07,30}$ was generated, weights were calculated for the basis at each $k = 1, \ldots, 18$, and the corresponding MSE $\text{tse}_(S)$ were calculated 100 times. The average of $\text{tse}_(S)$ at each k was calculated and the ln MSE_{avg} at each k was plotted. Over multiple runs, it can be seen that

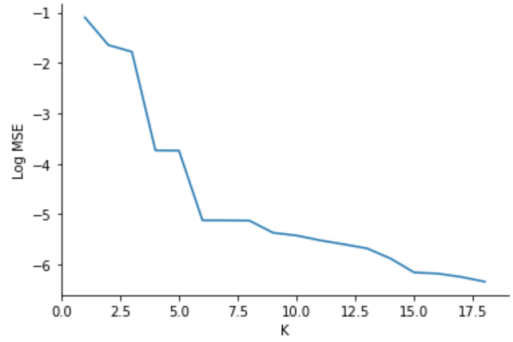
the MSE decreases, but has numerical errors for high values of k, causing the MSE to increase. This explains the fluctuation of the figure below around high k values.



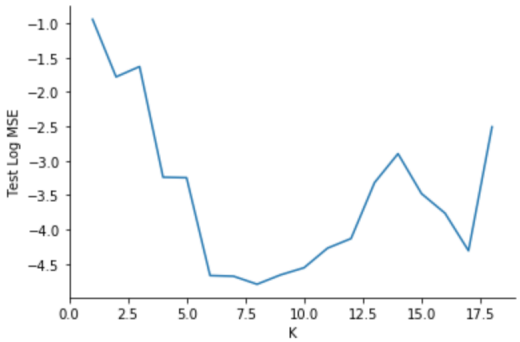
Test data $T_{0.07,1000}$ was generated, and the test error $\text{tse}_k(S,T)$ was calculated 100 times. The average of $\text{tse}_k(S,T)$ at each k was calculated and the ln MSE_{avg} was plotted. The MSE significantly increases as k increases because the model overfits and performs worse on the test data.



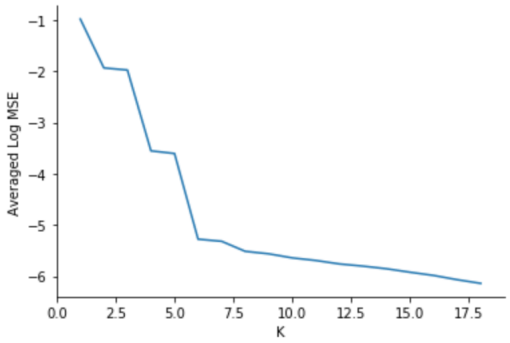
3. (a) Training data $S_{0.07,30}$ was generated using $g_\sigma(x) = \sin^2(2\pi x) + \epsilon$. The weights were calculated for the basis $\sin(1\pi x), \sin(2\pi x), \sin(3\pi x), \ldots, \sin(k\pi x)$ for $k = 1, \ldots, 18$ and the corresponding MSE, $\text{tse}_k(S)$ were calculated. The ln MSE at each k was plotted. The MSE steadily decreases with larger values of k.



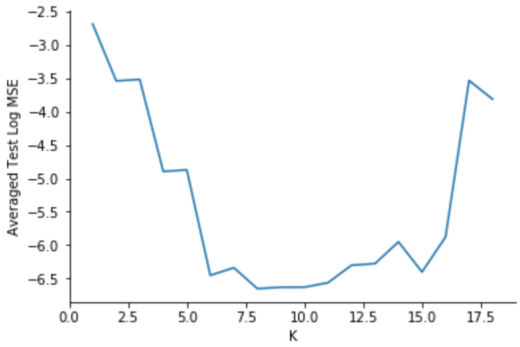
(b) A test set T: $T_{0.07,1000} = (x_1, g_{0.07}(x_1)), \ldots, (x_{1000}, g_{0.07}(x_{1000}))$ of a thousand points was generated. The test error $\text{tse}_k(S,T)$ was calculated by finding the MSE of the test set T on the basis $\sin(1\pi x), \sin(2\pi x), \sin(3\pi x), \ldots, \sin(k\pi x)$ and weights at k fitted from the training set S. The ln MSE at each $k = 1, \ldots, 18$ was plotted. The model begins to overfit and perform worse on the test data as k increases to higher values.



(c) Training data $S_{0.07,30}$ was generated, weights were calculated for the basis $\sin(1\pi x), \sin(2\pi x), \sin(3\pi x), \ldots, \sin(k\pi x)$ at each $k = 1, \ldots, 18$, and the corresponding MSE $\text{tse}_k(S)$ were calculated 100 times. The average of $\text{tse}_k(S)$ at each k was calculated and the ln MSE_{avg} was plotted. The MSE consistently decreases on the training data.



Test data $T_{0.07,1000}$ was generated, and the test error $\text{tse}_k(S, T)$ was calculated 100 times. The average of $\text{tse}_k(S, T)$ at each k was calculated and the ln MSE_{avg} was plotted. The test MSE begins to greatly increase for large values of k, as the model overfits the data.



1.2 Filtered Boston Housing and Kernels

4. (a) For 20 runs, the Boston Housing dataset was split into 2/3 training set and 1/3 test set, a vector of ones the length as the training set was used as a basis, and the weights were calculated using the training set. The MSE of the training set and the MSE of the test set using the weights from the training set were calculated. The naive regression does not perform very well, with a test MSE around 84, as it fits to the mean of y without using a basis dependent on x.

(b) A simple interpretation of the function 'a' above is that it is the equivalent to having a basis of 1 at $k = 1$ polynomial regression. It linearly fits the data with the line $y = c$ resulting in the least error (the mean of y). It can also be interpreted as using a bias to fit to y.

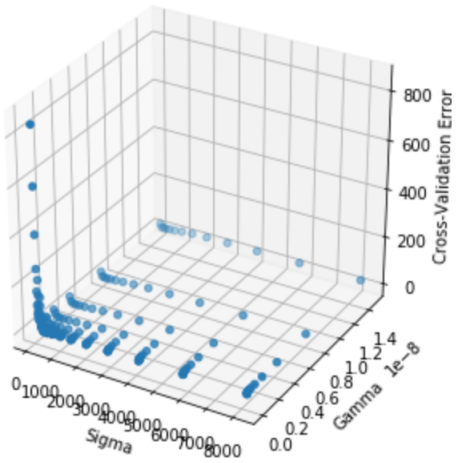
(c) For each of the 12 attributes, the data was run 20 times. For each run, the data was split into 2/3 training set and 1/3 test set and the basis of $\boldsymbol{x}_i, 1$ ($\boldsymbol{x}_i$ being x at the ith attribute) with dimension len$(X_{train}) \times 2$ was used to calculate the weights, and the MSE of the training set was calculated. A basis of $\boldsymbol{x}_i, 1$ with dimension len$(X_{test}) \times 2$ for the test set was then used with the weights found from the training set to calculate the MSE of the test set. The linear regression with single attributes performs better than the naive regression, but still has a high test MSE ranging from 36-85. By observing the MSE for each respective attribute, each attribute's probability of predicting y can be compared. Notably, attribute 12 has the lowest MSE around 36, indicating that it may be a more significant factor in predicting y. Attribute 4 has the highest MSE around 85, indicating that it may not be very significant in predicting y.

(d) The data was run 20 times using all attributes of the data. For each run, the data was split into 2/3 training set and 1/3 test set and the basis of $\boldsymbol{x}, 1$ with dimension len$(X_{train}) \times 13$ was used to calculate the weights, and the MSE of the training set was calculated. A basis of $\boldsymbol{x}, 1$ with dimension len$(X_{test}) \times 13$ for the test set was then used with the weights found from the training set to calculate the MSE of the test set. The linear regression using all attributes performs significantly better than that using individual attributes, with a test MSE around 23, as it fits the data to y using all of the contributing attributes.

1.3 Kernelised Ridge Regression

5. (a) A vector of γ values $[2^{-40}, 2^{-39}, \ldots, 2^{-26}]$ and a vector of σ values $[2^{7}, 2^{7.5}, \ldots, 2^{12.5}, 2^{13}]$ were created. A hyperparameter search was conducted by iterating through every combination of γ and σ. For each combination, the training split was separated into five folds, iterating through each 4 train fold and 1 test fold combination for cross-validation. In each fold iteration, the Gaussian kernel was calculated using $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$ and $\boldsymbol{\alpha}$ was calculated using $\boldsymbol{\alpha}^* = (\boldsymbol{K} + \gamma\ell\boldsymbol{I}_\ell)^{-1}\boldsymbol{y}$ with the train fold. The MSE of the train fold was calculated using the values found for $\boldsymbol{\alpha}$ and $\boldsymbol{K}$. The Gaussian kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_{test})$ was then calculated for the test fold and the MSE of the test fold was found using the test kernel and $\boldsymbol{\alpha}$ computed from the train fold. The train and test MSE were then averaged over the 5 folds. After all iterations the minimum train and test MSE of all γ and σ combinations were found. The optimal hyperparameter values were found to be $\gamma = 2^{-40}$ and $\sigma = 2^{9}$ on a single run, yielding a train MSE of 5.24 and test MSE of 12.08.

(b) The cross-validation error was plotted as a function of γ and σ. The cross validation error increases significantly with values of σ and γ near 0.



(c) Using the optimal values found for γ and σ, $\boldsymbol{K}$ and $\boldsymbol{\alpha}$ were calculated for the training set and the MSE was found. The test MSE was then found using $\boldsymbol{\alpha}$ from the training set and $\boldsymbol{K}$ calculated for the test set. Kernel ridge regression using the optimal (γ, σ) pair found performed the best out of all the methods implemented, with an average test MSE around 18.76, potentially due to its optimization with a nonlinear method. Using kernelization, the model is able to fit to nonlinear data more optimally than linear methods.

(d) The various methods performed in question 4 and question 5 were done over 20 random (2/3,1/3) splits of the data and the results are summarized below. The reported values were found over a particular run and fluctuate slightly from run to run due to the randomization of initialization.

Method	**MSE Train**	**MSE Test**
Naive Regression	85.98±4.28	81.38±8.31
Linear Regression (attribute 1)	72.99±4.33	69.99±8.66
Linear Regression (attribute 2)	72.57±4.99	75.61±9.64
Linear Regression (attribute 3)	63.07±5.80	68.03±11.52
Linear Regression (attribute 4)	80.25±2.36	85.49±4.58
Linear Regression (attribute 5)	65.63±4.95	75.72±9.69
Linear Regression (attribute 6)	43.10±3.80	44.86±7.28
Linear Regression (attribute 7)	71.19±5.33	75.07±10.35
Linear Regression (attribute 8)	78.28±4.49	81.09±8.60
Linear Regression (attribute 9)	72.44±4.41	71.12±8.61
Linear Regression (attribute 10)	65.50±4.75	66.18±9.25
Linear Regression (attribute 11)	61.48±4.84	65.65±9.61
Linear Regression (attribute 12)	39.28±2.63	36.13±5.01
Linear Regression (all attributes)	22.15±2.06	23.89±5.19
Kernel Ridge Regression	4.14±1.80	18.76±6.35

2 PART II

Notation: We overload [.] as follows: $[n] := \{1, 2, ..., n\}$ if n is a positive integer and $[pred] = 1$ if $pred$ is a logical predicate which is true and $[pred] = 0$ otherwise.

2.1 Bayes Estimator

In both of the following subquestions, we will find the Bayes estimator with respect to the probability mass function $p(x, y)$ over (X, Y) where X and Y are finite thus $\sum_{x \in X} \sum_{y \in Y} p(x, y) = 1$.

6. (a) For this subquestion $Y = [k]$ and let $c \in [0, \infty)^k$ be a vector of k costs. Let us define $L_c : [k] \times [k] \longrightarrow [0, \infty)$ such that

$$L_c(y, \hat{y}) := [y \neq \hat{y}]c_y$$

as the imbalanced classification loss function, i.e., if we don't predict the correct outcome y we suffer c_y loss.

Setting $\hat{y} = f(x)$ for an arbitrary predictor f, the expected error of f is:

$$\mathcal{E}(f) := \sum_{x \in X} \sum_{y \in Y} [y \neq f(x)]c_y p(x, y)$$

$$= \sum_{x \in X} \sum_{y \in Y} [y \neq f(x)]c_y p(y|x)p(x)$$

Now let us find the value of $\mathcal{E}(f)$ at a specific point.
Setting $x = x'$:

$$\mathcal{E}(f(x')) = \sum_{y \in Y} [y \neq f(x')]c_y p(y|x')p(x')$$

Using the notation $e = \mathcal{E}(f(x'))$ and replacing '=' by '$\propto$' (proportional), we obtain:

$$e \propto \sum_{y \in Y} [y \neq f(x')]c_y p(y|x')$$

$$= \sum_{y \in Y} (1 - [y = f(x')])c_y p(y|x')$$

$$= \sum_{y \in Y} c_y p(y|x') - \sum_{y \in Y} [y = f(x')]c_y p(y|x')$$

We know that the Bayes estimator is:

$$f^*(x) := argmin_f \mathcal{E}\left(f(x)\right)$$

so,

$$f^*(x') = argmin_f \left(\sum_{y \in Y} c_y p(y|x') - \sum_{y \in Y} [y = f(x')]c_y p(y|x') \right)$$

$$= argmin_f \left(- \sum_{y \in Y} [y = f(x')]c_y p(y|x') \right)$$

$$= argmax_f \sum_{y \in Y} [y = f(x')]c_y p(y|x')$$

Therefore, the Bayes estimator is:

$$f^*(x) = argmax_f \sum_{y=f(x)} c_y p(y|x)$$

(b) For this subquestion $Y \subset \Re$. Let $F : \Re \longrightarrow \Re$ be a strictly convex differentiable function. So $F' : \Re \longrightarrow \Re$ is a strictly increasing function. Define the F-loss as

$$L_F(y, \hat{y}) := F(\hat{y}) - F(y) + (y - \hat{y})F'(y).$$

i. Set $F(x) = x^2$, then $F'(x) = 2x$, and

$$L_F(y, \hat{y}) = y^2 - \hat{y}^2 + (y - \hat{y})2y = y^2 - \hat{y}^2 + 2y^2 - 2y\hat{y} = (y - \hat{y})^2$$

Therefore, in the case where $F(x) = x^2$, the F-loss is the square loss function.

ii. ($\Rightarrow$) Suppose $y = \hat{y}$:

$$L_F(y, \hat{y}) = F(y) - F(y) + (y - y)F'(y) = 0$$

Thus

$$y = \hat{y} \Longrightarrow L_F(y, \hat{y}) = 0$$

($\Leftarrow$) Suppose $L_F(y, \hat{y}) = 0$ and $y < \hat{y}$, then:

$$\frac{F(\hat{y}) - F(y)}{\hat{y} - y} = F'(y)$$

Using the Mean Value Theorem:

$$\exists\ c \in]y, \hat{y}[\text{ s.t } \frac{F(\hat{y}) - F(y)}{\hat{y} - y} = F'(c)$$

Thus:

$$F'(y) = F'(c)$$

This is a contradiction since F' is strictly increasing and continuous, and $c < \hat{y}$. Therefore $y \geq \hat{y}$.
Similarly, supposing $L_F(y, \hat{y}) = 0$ and $y > \hat{y}$, we can show by proof of contradiction that $\hat{y} \geq y$.

Hence $y = \hat{y}$, and therefore,

$$y = \hat{y} \ \Leftrightarrow L_F(y,\ \hat{y}) = 0$$

iii. Let $y, \hat{y} \in \Re$.
We know that F is strictly convex and F' is strictly increasing.
Differentiating the F-loss function w.r.t. $\hat{y}$:

$$\frac{\partial L_F(y, \hat{y})}{\partial \hat{y}} = F'(\hat{y}) - F'(y)$$

We now find the minimum of the F-loss function:

$$\frac{\partial L_F(y, \hat{y})}{\partial \hat{y}} = 0 \Rightarrow F'(\hat{y}) = F'(y)$$
$$\Rightarrow\ \hat{y} = y$$

The second result follows from the fact that F' is a one-to-one function.
So the minimum is found at $\hat{y} = y$. We know from part ii. that

$$y = \hat{y} \ \Rightarrow L_F(y, \hat{y}) = 0$$

Hence the minimum of the F-loss function is 0.
Therefore, for all $y, \hat{y} \in \Re$, we have

$$L_F(y, \hat{y}) \geq 0.$$

iv. We will now derive the Bayes estimator for the F-loss function. Setting $\hat{y} = f(x)$ for an arbitrary predictor f, the expected error of f is:

$$\begin{aligned}\mathcal{E}(f) &:= \sum_{x \in X} \sum_{y \in Y} L_F(y, f(x)) p(x, y) \\ &= \sum_{x \in X} \sum_{y \in Y} L_F(y, f(x)) p(y|x) p(x)\end{aligned}$$

Now let us find the value of $f^*(x)$ at a specific point.
Setting $x = x'$:

$$\mathcal{E}(f(x')) = \sum_{y \in Y} L_F(y, f(x')) p(y|x') p(x')$$

Using the notation $e = \mathcal{E}(f(x')),\ z = f(x')$ and replacing '=' by '$\propto$' (proportional), we obtain:

$$\begin{aligned}e &\propto \sum_{y \in Y} L_F(y, f(x')) p(y|x') \\ \frac{\partial e}{\partial z} &= \sum_{y \in Y} (F'(z) - F(y)) p(y|x')\end{aligned}$$

Setting the derivative to 0, we obtain:

$$\frac{\partial \mathcal{E}}{\partial z} = 0 \Longrightarrow \sum_{y \in Y} F'(z) p(y|x') = \sum_{y \in Y} F'(y) p(y|x')$$

We know that:

$$\sum_{y \in Y} p(y|x') = 1$$

Thus

$$\frac{\partial \mathcal{E}}{\partial z} = 0 \Longrightarrow F'(z) = \sum_{y \in Y} F'(y) p(y|x')$$

Since F' is strictly increasing and continuous, then it is invertible. Therefore the Bayes estimator is:

$$F^*(x) = (F')^{-1} \left(\sum_{y \in Y} F'(y) p(y|x) \right) = (F')^{-1} \left(\mathbb{E} \left[F'(y) | x \right] \right)$$

Now that we found the Bayes estimator for the F-loss, let us apply it to the case where $F(x) = |x|^p$ for $p > 1$.

First of all, we calculate the derivative of $F(x) = |x|^p$:

$$F'(x) = p.sign(x).x^{p-1} \ .$$

Now, we need to calculate the inverse of the function F'.

Let $y \in \Re$ and $p > 1$,

$$F'(y) = x \Longrightarrow p.sign(y).y^{p-1} = x$$

$$\Longrightarrow sign(y).y^{p-1} = \frac{x}{p}$$

We can see from the following plots of the inverses of $g(x) = |x|^5$ and $h(x) = |x|^6$ and their inverses' derivatives, that $sign(y) = sign(x)$:

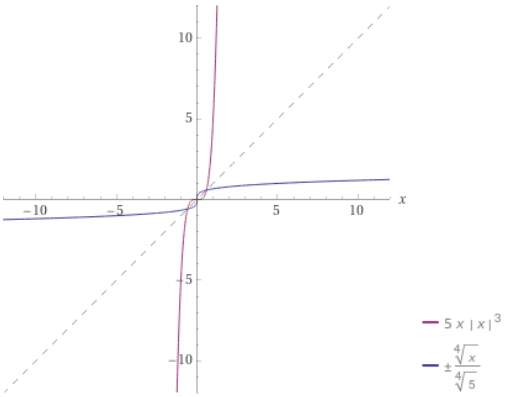


Figure 1: Inverse and inverse's derivative of $g(x) = |x|^5$

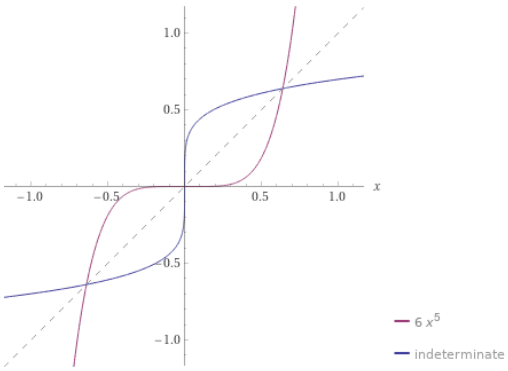


Figure 2: Inverse and inverse's derivative of $h(x) = |x|^6$

Thus,

$$\implies y^{p-1} = \begin{cases} \frac{x}{p}, & \text{if } x \geq 0, \\ -\frac{x}{p}, & \text{else} \end{cases}$$

$$\implies y = \begin{cases} \left(\frac{x}{p}\right)^{\frac{1}{p-1}}, & \text{if } x \geq 0, \\ \left(-\frac{x}{p}\right)^{\frac{1}{p-1}}, & \text{else} \end{cases}$$

Hence,

$$(F')^{-1}(x) = \begin{cases} \left(\frac{x}{p}\right)^{\frac{1}{p-1}}, & \text{if } x \geq 0, \\ \left(-\frac{x}{p}\right)^{\frac{1}{p-1}}, & \text{else} \end{cases}$$

Therefore, the Bayes estimator for the F-loss for $F(x) = |x|^p$ with $p > 1$ is:

$$F^*(x) = (F')^{-1}\left(\mathbb{E}\left[p.sign(y).y^{p-1}|x\right]\right) = \left(\mathbb{E}\left[sign(y).y^{p-1}|x\right]\right)^{\frac{1}{p-1}}.$$

2.2 Kernel Modification

Consider the function $K_c(\mathbf{x}, \mathbf{z}) = c + \sum_{i=1}^{n} x_i z_i$ where $\mathbf{x}, \mathbf{z} \in \Re^n$.

7. (a) Let us find the values of $c \in \Re$ for which K_c is a positive semi-definite kernel.
Assume K_c is positive semi-definite, the following needs to be true:

$$K_c(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

for some feature map $\phi : \Re^n \longrightarrow W$ and Hilbert space W.

By defining the feature map ϕ as:

$$\phi(\mathbf{x}) = \left(x_1, x_2, ..., x_n, \sqrt{c}\right)^T, \text{ for } \mathbf{x} \in \Re^{n+1},$$

we require $c \geq 0$, and we obtain:

$$K_c(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

$$= c + \sum_{i=1}^{n} x_i z_i$$

Let K be a positive semi-definite kernel such that:

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} x_i z_i = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle, \text{ for } \mathbf{x}, \mathbf{z} \in \Re^n.$$

For m data points, $x_i \in \Re^n$ and $a_i \in \Re, i = 1, ..., n$, we have:

$$\sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j K_c(x_i, x_j) \geq 0 \Longrightarrow \sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j (c + K(x_i, x_j)) \geq 0$$

$$\Longrightarrow \sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j c + \sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j K(x_i, x_j) \geq 0$$

$$\Longrightarrow c \left(\sum_{i=1}^{m} a_i \right)^2 + \left\| \sum_{i=1}^{m} a_i \phi(x_i) \right\|^2 \geq 0$$

Therefore, for $c \geq 0$, K_c is a positive semi-definite kernel.

(b) If we use K_c as a kernel function with linear regression (least squares), then c plays the role of the bias. If $c = 0$, then the regression line (solution) has to go through 0, leaving no room to improve our fitted model. For example, if all $x_i's = 0$, then the predicted value is also 0, which may be different than what the data would suggest. On the other hand, if $c > 0$, then our model becomes more flexible and allows the regression line to fit the data more easily.

2.3 Simulating 1-NN

8. Suppose we perform linear regression with a Gaussian kernel $\mathbf{K}_\beta(\mathbf{x}, \mathbf{t}) = \exp(-\beta ||\mathbf{x} - \mathbf{t}||^2)$ to train a classifier on a dataset $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m) \in \Re^n \times \{-1, 1\}$. We then obtain a function $f : \Re^n \longrightarrow \Re$ which is of the form $f(\mathbf{t}) = \sum_{i=1}^{m} \alpha_i K_\beta(\mathbf{x}_i, \mathbf{t})$, whose classifier is $sign(f(\mathbf{t}))$. This classifier depends on the parameter β selected for the kernel.

$$f(\mathbf{t}) = \sum_{i=1}^{m} \alpha_i \exp(-\beta ||\mathbf{x} - \mathbf{t}||^2)$$

We can see from the equation above that the parameter β represents the width of the Gaussian kernel. It controls how the distances between the test point and the sample points affect the neighbors' influence on the prediction.

Intuitively, we do not want β to be very small, as it would give more influence to farther points. Hence, we want β to be large, so that the farther points become less relevant.

Now we find the scenario for which chosen β enables the trained linear classifier to simulate a 1-Nearest Neighbor algorithm trained on the same dataset.

Without loss of generality, we assume that the nearest neighbor to our test point $\mathbf{t}$ is $\mathbf{x}_1$, i.e. $\mathbf{x}_1 = argmin_{\mathbf{x}}\{||\mathbf{x} - \mathbf{t}||^2\}$.

So we want to choose β such that:

$$sign(f(\mathbf{t})) = sign(\alpha_1 \mathbf{K}_\beta(\mathbf{x}_1, \mathbf{t})) = sign(\alpha_1)$$

The last equation holds because $\mathbf{K}_\beta(\mathbf{x}, \mathbf{t}) > 0$ for all $\mathbf{x}, \mathbf{t} \in \Re^n$.

As β gets larger and larger, the α's and y's become equal, because the kernel becomes the identity matrix (from the expression $\boldsymbol{\alpha} = (\boldsymbol{K} + \gamma \boldsymbol{I})^{-1}\boldsymbol{y}$, for some hyperparameter γ).

This means that we want $\beta = \hat{\beta}(\mathbf{x}_1, ..., \mathbf{x}_m, \mathbf{t})$ to be large enough but finite so that the value of the Gaussian kernel applied at each data point tends to 0 except for the nearest point to our test point.

This way, the classifier outputs the sign of the nearest neighbor, and therefore, the linear classifier simulates a 1-Nearest Neighbor algorithm.

2.4 Whack-A-Mole

9. Let B_n be the set of $n \times n$ matrices with values in {0,1}, and $W_{i,j}$: $B_n \longrightarrow \mathrm{B}_n$ be the function that transforms a board by whacking the position (i, j).

   We assume that it is possible to whack empty holes.

   We can see (through simple maths) that $\mathbf{W} = \{W_{i,j} | i, j \in [1, n]\}$ is a $\mathbb{Z}/2\mathbb{Z}$ vector space.

   Denote by E_W the vector space generated by $\mathbf{W}$ and $\mathbb{Z}/2\mathbb{Z}$, with $\mathbf{W}$ as a basis.

   We are trying to find a sequence $f \in E_{\mathbf{W}}$, such that:

   $$f^{-1}(\mathbf{0}_n) = \mathbf{X}, \text{ where } X \in B_n.$$

   This means that we want to start from an empty board, and by whacking a number of times, we obtain the initial board.

   Let

   $$V_{i,j} = \sum_{i'=i-1}^{i+1} \sum_{j'=j-1}^{j+1} \mathbb{1}_{i',j' \in [1,n]} \ \mathrm{W}_{i,j}$$

It is easy to demonstrate that $\mathbf{V} = \{V_{i,j} | i, j \in [1, n]\}$ is another basis for $E_{\mathbf{W}}$.

Denote by '.' the scalar product in $E_{\mathbf{W}}$. So

$$f^{-1}.V_{i,j} = X_{i,j}$$

Let $P_{\mathbf{V}\to\mathbf{W}}$ be the transition matrix between the bases $\mathbf{V}$ and $\mathbf{W}$.

We have

$$f^{-1} = \sum_{i,j=1}^{n} X_{i,j} V_{i,j}$$

$$f = \sum_{i,j=1}^{n} X_{i,j} V_{i,j}$$

The last equality follows because we are working in $\mathbb{Z}/2\mathbb{Z}$.

So the representation of f in $\mathbf{W}$ is:

$$P_{\mathbf{V}\to\mathbf{W}}\ \mathbf{X} = \mathrm{P}^{-1}_{\mathbf{W}\to\mathbf{V}}\ \mathbf{X}$$

We can see that the inverse of $P_{\mathbf{V}\to\mathbf{W}}$ is calculated in polynomial time.

We can then determine if f is actually possible by whacking the possible coordinates. If at some point there is no available mole and we have not yet gone through all components of f, then we can consider that the given matrix has no solution. Otherwise, we return the components of f in the order of execution.