6.867
Problem Set 1

Dimitrios Koutentakis

September 2018

Classification Error of k-NN, k = 1

1.1 Bayes classifier and error

In the k-NN algorithm with $k = 1$, we compute the Bayes Classifier ($Y^*(x)$) and the Bayes error ($\delta^*(x)$). We have the feature random variable X which we want to classify with the label $Y \in \{0, 1\}$. The Bayes Classifier for this problem will be:

$$Y^*(x) = \begin{cases} 1, \text{ if } P(Y=1|X=x) > \frac{1}{2} \\ 0, \text{ otherwise} \end{cases}$$

The corresponding error($\delta^*(x)$) will be the probability that x gets misclassified. Thus:

$$\begin{aligned}
\delta^*(x) =& \mathbb{P}(Y*(x) \neq Y) \\
=& \mathbb{P}(Y^*(x) = 0|Y=1) \cdot \mathbb{P}(Y=1|X=x) \\
&+ \mathbb{P}(Y^*(x)=1|Y=0) \cdot \mathbb{P}(Y=0|X=x) \\
=& \mathbb{P}(Y^*(x)=0|Y=1)\eta_1(x) \\
&+ \mathbb{P}(Y^*(x)=1|Y=0)\eta_0(x) \\
=& \begin{cases} \eta_1, \text{ if } \eta_0(x) \geq \frac{1}{2} \\ \eta_0, \text{ if } \eta_1(x) \geq \frac{1}{2} \end{cases} \\
\Rightarrow \delta^*(x) =& min(\eta_1, \eta_2)
\end{aligned}$$

1.2 VC dimension

The VC dimension of the 1-NN algorithm will be infinity. That is because for any given number of points, we can have just as many classes, i.e. for any training set we can achieve an training error of zero. That is to say that the algorithm can assign each point to its own class. Then 1-NN will be able to shatter the whole set of the points up to infinity.

Since the VC-dimension is infinity, we are at a great risk of overfitting to the training data. Since the generalization error is the testing error minus the training error and the training error is zero, the generalization error is the testing error. It is easy to construct many adversarial test sets, all points of which will be misclassified. Thus the testing error (and hence the generalization error) is unbounded.

1.3 Error proof

$$\begin{aligned}
P(|X_j - x| \geq \beta) =& 1 - P(|X_j - x| < \beta) \\
=& 1 - P(x - \beta < X_j < x + \beta) \\
=& 1 - P(x-\beta < X_j < x+\beta|Y=0)P(Y=0) \\
&- P(x-\beta < X_j < x+\beta|Y=1)P(Y=1) \\
=& 1 - \int_{x-\beta}^{x+\beta} f_0(t)P(Y=0) + f_1(t)P(Y=1)dt \\
\leq& 1 - \int_{x-\beta}^{x+\beta} \gamma P(Y=0) + \gamma P(Y=1)dt \\
\leq& 1 - 2\beta\gamma
\end{aligned}$$

Thus:

$$\begin{aligned}
P(\min_{1\leq n} |X_j - x| \geq \beta) =& P(|X_1 - x| \geq \beta, ..., |X_n - x| \geq \beta) \\
=& \prod_{j=1}^{n} P(|X_j - x| \geq \beta) \\
\leq& \prod_{j=1}^{n} (1 - 2\beta\gamma) \\
\leq& (1-2\beta\gamma)^n
\end{aligned}$$

Since: $0 < 1 - 2\beta\gamma < 1$, the limit becomes:

$$\begin{aligned}
\lim_{n\to\infty} P(\min_{1\leq j} |X_j - x| \geq \beta) =& \lim_{n\to\infty} (1-2\beta\gamma)^n \\
=& 0
\end{aligned}$$

1.4 Misclassification limits

Here we have:

$$
\begin{aligned}
&P(Y \neq Y_{1NN}(x)|X = x) = \\
&= \int_a^b P(Y(x) \neq Y(\xi)|X = x) \\
&\times P(\min_{1\leq j\leq n} |X_j - x| = |x - \xi|)d\xi \\
&= \int_a^b \Big(P(Y(x) = 1, Y(\xi) = 0|X = x) \\
&+ P(Y(x) = 0, Y(\xi) = 1|X = x)\Big) \\
&\times P(\min_{1\leq j\leq n} |X_j - x| = x - \xi)d\xi \\
&= \int_a^b \Big(\eta_1(x)\eta_0(\xi) + \eta_0(x)\eta_1(\xi)\Big) \\
&\times P(\min_{1\leq j\leq n} |X_j - x| = |x - \xi|)d\xi
\end{aligned}
$$

Additionally:

$$
\eta_0(X_{n'(x)}) = \int_a^b P(Y(x) = 0|X = \xi) \times P(\min_{1\leq j\leq n} |X_j - x| = |x - \xi|)d\xi
$$

In the case where $n \to \infty$, we showed that the probability of x not being seen tends to zero. Thus, for $\xi \neq x$, $\lim_{n\to\infty} P(\min_{1\leq j\leq n} |X_j - x| = |x - \xi|) = 0$. In the case where $\xi = x$: we have:

$$
\begin{aligned}
\eta_0(X_{n'(x)}) =& P(Y = 0|X = x)P(\min_{1\leq n} |X_j - x = 0) \\
=& \eta_0(x)
\end{aligned}
$$

Similarly, we have:

$$
\eta_1(X_{n'(x)}) = \eta_1(x)
$$

1.5 Limit of infinite samples

From (d), we want to find the asymptotic behavior of: $P(Y \neq Y_{1NN}(x)|X = x)$. When $\lim_{n\infty}$, the integral evaluates to 0 everywhere except where $\xi = x$. Then:

$$
\begin{aligned}
&\lim_{n\to\infty} P(\min_{1\leq j\leq n} |X_j - x| = |x - \xi|) = \\
&= 2\eta_1(x)\eta_0(x) \\
&= 2\min\{\eta_1(x), \eta_0(x)\}(1 - \min\{\eta_1(x), \eta_0(x)\}) \\
&\leq 2\min \eta_1(x), \eta_0(x) \\
&= \delta^*(x)
\end{aligned}
$$

Logistic regression

2.1 VC-Dimension

Here we calculate the VC-dimension of the logistic regression classifier. In particular, we prove that the VC-dimension of logistic regression is $\geq d$. Then, we will prove that the logistic regression has a VC-dimension of $< d$, where d is the dimension of our data ($x_1, x_2, ..., x_d \in \mathbb{R}^d$).

2.1.1

Let us start by taking d points in $\mathbb{R}^d$. Then we can prove that logistic regression can always shatter the set.

Let the points $x_1, x_2, ..., x_d$ be linearly independent. Also, let x be the matrix with dimensions $d \times d$ containing our points and Y is a row vector with the labels of our points. Then, a classifier can shatter them if we can find a w with dimensions $d \times 1$ for any of the possible labels of the points that can occur in Y such that:

$$
\begin{aligned}
w^T X = Y \\
\Rightarrow w^T = Y X^{-1}
\end{aligned}
$$

Since our points are linearly independent, the matrix X will have to be non-singular. In that case, X^{-1} exists and hence $\forall Y$ we can find such w^T (hyperplane) that shatters the points. Thus the logistic regression classifier will have a VC-dimension larger or equal to d, where d is the dimension of our data ($x_1, x_2, ..., x_d \in \mathbb{R}^d$).

2.1.2

Let us now choose $d + 1$ points in $\mathbb{R}^d$, such that none of them is the zero-vector. Then, since we have more points than dimensions, and a_i are real numbers, and $a_1 \times a_2 \times ... \times a_3 \neq 0$, we will have:

$$
a_1x_1 + a_2x_2 + a_3x_3 + ... + a_dx_d + a_{d+1}x_{d+1} = 0
$$

We then assign the labels of the point x_i based on the sign of its coefficient a_i (if it is non-zero). Let us also assume that there exists a w matrix of dimensions dx1 such that $w^T x_i = sign(a_i)$, if $a_i \neq 0$ and random if $a_i = 0$. Then, we have:

$$
a_1w^Tx_1 + a_2w^Tx_2 + a_3w^Tx_3 + ... + a_dw^Tx_d + a_{d+1}w^Tx_{d+1} = 0
$$

Every term above is either a zero or positive, since $w^T x_i = sign(a_i) \Rightarrow a_iw^Tx_i \geq 0$, if $a_i \neq 0$, or the term is zero by $a_i = 0$. Then, in order for the equality to hold, all the terms will have to be equal to zero. However it is the case that not all a_i can be zero. If the classifier shatters the points, then $w^T x_i \neq 0i$. This cannot be true and we hence have that our linear classifier cannot successfully shatter these $d + 1$ points.

2.2 Linearly separable Logistic Regression

In order to classify data points, the logistic regression uses the sigmoid function:

$$\sigma(w^T x - w_0) = \frac{1}{1 + e^{-(w^T x + w_0)}}$$

Using the sigmoid function, a point is correctly classified if the sign of $w^T x$ is the correct sign (meaning positive sign for $y_i = 1$, negative otherwise. The magnitude of this result changes the "sharpness" of the function (how steep it is) and it expresses the confidence of the classifier in the prediction. We assumed that our training data is linearly separable, ie $\forall x_i \exists w$, such that $w^T x_i$ has the correct sign. Now, if we scale w to αw, with $\alpha > 1$, we will still obtain the same signs, but with a greater magnitude and hence a greater confidence.

If we scale w by α, and use αw in our $L(w)$ equation, given that w correctly classifies our data, we can see that $L(\alpha w) < L(w)$, if $\alpha < 1$. It is thus obvious that we want a values of w as large as possible.

In Figure 1, we can see how the sigmoid function changes for different values of w in the 1D case. Here we have used a values of: $w = \{1, 5, 10, 50\}$ and an offset $w_0 = -2$. Our sigmoid function thus is: $\sigma(w^T x - 2) = \frac{1}{1+e^{-w(x-2)}}$.

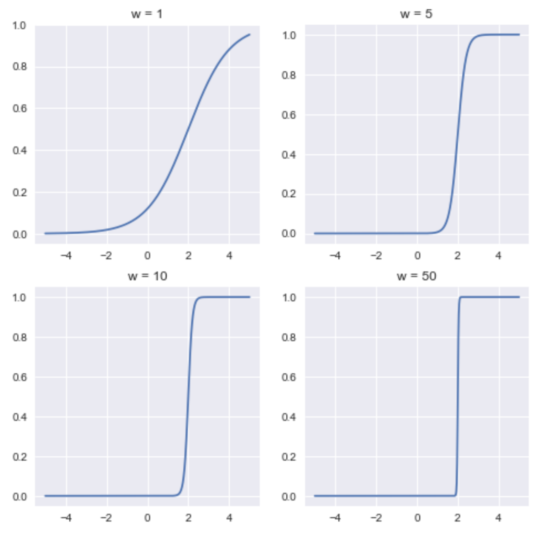


Figure 1: Sigmoid functions for varying w

2.3 Least Squared "Logistic Regression"

2.3.1

The squared loss function for the Logistic Regression is:

$$L(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - \sigma(w^T x_i))^2$$

In order to find the gradient descent step for minimizing the squared loss function, we need to differentiate in order to get the gradient of the function $(\nabla L(w))$. Thus we get:

$$\nabla L(w) = -\sum_{i=i}^{m} (y_i - \sigma(w^T x_i)(\sigma(w^T x_i)(1 - \sigma(w^T x_i))x_i$$

Then the gradient descent step for minimizing squared loss is:

$$w^{t+1} = w^t + \alpha \sum_{i=i}^{m} (y_i - \sigma(w^T x_i)(\sigma(w^T x_i)(1 - \sigma(w^T x_i))x_i$$

2.3.2

John's overall approach to find the parameters w by performing gradient descent on the quared loss function of the Logistic Regression is not a good one. In particular, it is not guaranteed that this loss function will be globally non-increasing. That means that the function might have one (or several) local minima which cause the gradient descent optimizer to get stuck.

One such example is constructed with the points: $X = \{2, -10, 20, -4, 0, 3, -7, -100\}$ and $Y = \{1, 1, 0, 1, 1, 1, 1, 0\}$. These numbers result in a Logistic Regression model with the squared loss function shown in Figure 2:

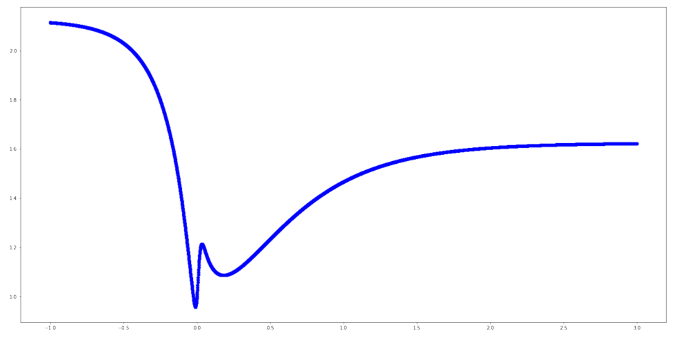


Figure 2: Loss function of our Logistic Regression model

The squared loss function in Figure 2 is not easy to minimize with normal gradient descent, because there are multiple local optima, different than the global minimum. Thus the optimizer might converge somewhere that is not the actual solution.

Support Vector Machine (SVM)

3.1 Implementing the dual form of linear SVMs with slack variable

The dual form of the linear SVM consists of the objective:

$$\min \frac{1}{2}(x^T qx) + Px$$

And the constraints:

$$GA \leq h$$
$$xA = b$$

When applying our dual form of a linear SVM to the data set: X, Y, with:

$$X = \begin{bmatrix} 2 & 2 \\ 2 & 3 \\ 0 & -1 \\ -3 & -2 \end{bmatrix} \qquad Y = \begin{bmatrix} +1 \\ +1 \\ -1 \\ -1 \end{bmatrix}$$

We get the following matrices:

$$P = \begin{pmatrix} 8 & 10 & 2 & 10 \\ 10 & 13 & 3 & 12 \\ 2 & 3 & 1 & 2 \\ 10 & 12 & 2 & 13 \end{pmatrix} \qquad q = \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$$

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{pmatrix} \qquad h = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 1 & -1 & -1 \end{pmatrix} \qquad b = \begin{pmatrix} 0 \end{pmatrix}$$

The support vectors are $(2, 2)$ and $(0, -1)$.

3.2 Testing SVm on 2D dataset

Testing the implementation of the linear SVM on the training and validation data of the four datasets, we get the following error rates, summarized in Table 1:

Data Set	Train Error Rate	Validation Error Rate
Data 1	0.0	0.0
Data 2	0.1775	0.18
Data 3	0.02	0.03

Table 1: Error Rates for Linear SVM

The classifications, along with the boundary and the margins (as well as the error rates) are shown in Figure 3.

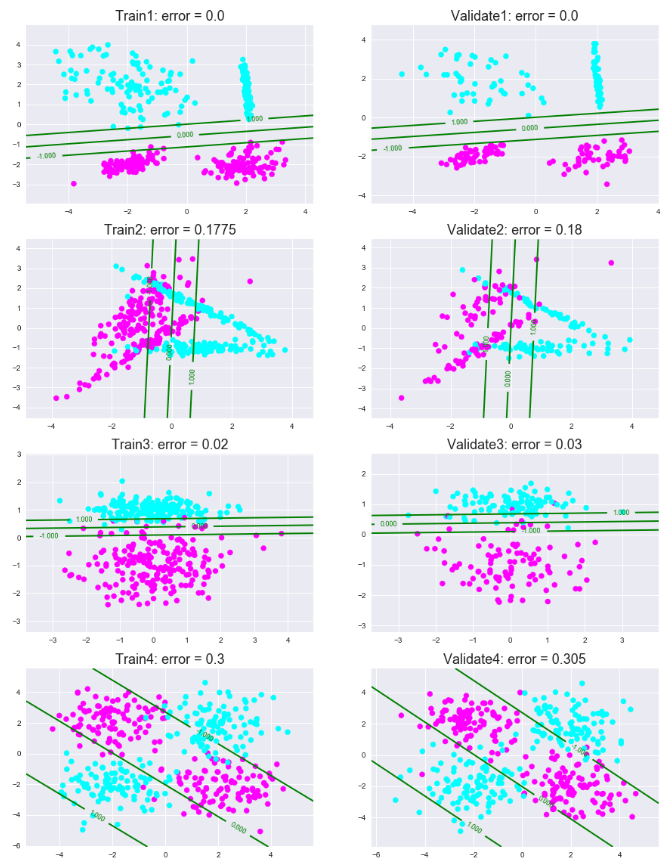


Figure 3: Classifications of each of the data sets with accuracy

3.3 Extending SVM to kernels

3.3.1

When extending the dual form SVM to work with kernels, we can see how the output of the algorithm changes as we vary C over $\{0.01, 0.1, 1.0, 10.0, 100.0\}$.

From Table 2, we observe that as C increases, the geometric margin $\frac{1}{||w||}$ decreases. This also makes intuitive sense because as C gets bigger, we penalize more and more the points that are missclassified. Thus the classifier brings the margins closer to the boundary in order to have a lesser training error.

3.3.2

As seen in 2, the number of support vectors decreases as C increases. This happens because as C increases, the margin becomes tighter and thus a smaller amount of points fall within the margin. (All the points within the margin are support vectors).

C	Linear Kernel		Gaussian	
	Support Vectors	$\|\|w\|\|^2$	Support Vectors	$\|\|w\|\|^2$
0.01	75	0.54	400	2.34
0.1	20	1.5	109	8.88
1.0	4	3.14	36	14. 78
10.0	3	4.0	31	15.86
100.0	3	4.0	30	15.86

Table 2: $||w||^2$ and of support vectors for varying C

3.3.3

Maximizing the geometric margin $1/||w||$ would not be an appropriate criterion for selecting C. In order to maximize the margin, we would get a C of 0, and our $||w||$ would be approaching 0, meaning that we don't gain much information about our data. This would result into a very weak classifier.

An alternative criterion we could use in order to select the value C would be cross-validation. By splitting our data into a training and a validation test, we can try out several values of C. Then, based on those empirical observations, we can choose the one that results in the best classification of the validation data (i.e. lowest validation error).

Soft-Margin SVM with Pegasos

4.1 Implementing Pegasos

In the following problems, we implement the Pegasos algorithm and answer the questions.

4.2 Testing regularization values

In Table 3, we observe that the margin increases as λ increases. This makes sense because as λ increases we have higher regularization and thus less overfitting. The regularization parameter λ penalizes $||w||$, so as λ increases, $||w||$ should decrease. Thus, $\frac{1}{|w|}$ increases, so the margin gets wider.

λ	$1/\|\|w\|\|^2$		λ	$1/\|\|w\|\|^2$
2^{-10}	0.23		2^{-4}	0.67
2^{-9}	0.27		2^{-3}	0.78
2^{-8}	0.34		2^{-2}	0.93
2^{-7}	0.42		2^{-1}	1.14
2^{-6}	0.47		2^{0}	1.43
2^{-5}	0.56		2^{1}	2.17

Table 3: $\frac{1}{||w||^2}$ for varying λ

4.3 Kernelizing Pegasos algorithm

As in the previous question, we have

$$h(x) = \langle \omega, \phi(x) \rangle = \sum_i a_i y_i k(x_i, x).$$

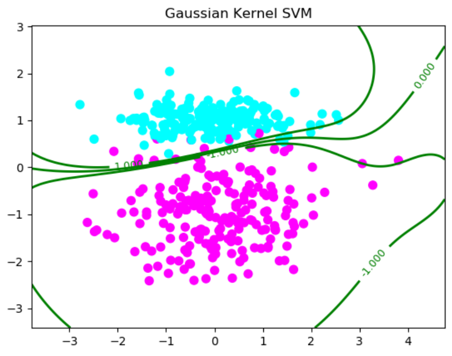
Hence we would expect that the solution to be sparse, but most likely not as sparse as the linear. This happens because the gaussian model has more complex features than the linear, so it is more likely to overfit.

4.4 Gaussian kernel classification

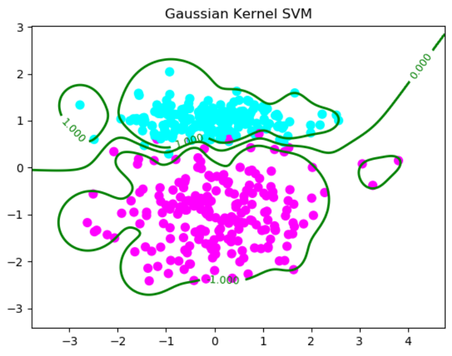
In Table 4.4, as well as in Figures 4a and 4b, we can observe that the larger the γ parameter, the tighter the decision boundary. Thus, fewer and fewer points are misclassified and the closer we get to overfitting.

Additionally, we can also see that the number of support vectors increases as γ increases.

γ	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}
#sv	30	30	29	38	58



(a) Small γ value



(b) Large γ value

Figure 4: RBF for different values of γ

Here we can tune the paramater γ in order to reduce the training error arbitrarily. Thus we can get better accuracy than in the previous section, but we would also have a lot more overfitting.