Homework 1

Tiago Antunes

October 21, 2020

1 Mathematics Basics

1.1 Optimization

$$\begin{aligned} \min_{x_1,x_2} \quad & x_1^2 + x_2^2 - 1 = f(x_1, x_2) \\ s.t. \quad & x_1 + x_2 - 1 = h(x_1, x_2) = 0 \\ & x_1 - 2x_2 \geq 0 \iff -x_1 + 2x_2 = g(x_1, x_2) \leq 0 \end{aligned}$$

First we need to get the Lagrange function:

$$\mathcal{L}(x_1, x_2, \lambda_1, \lambda_2) = x_2^2 + x_2^2 - 1 + \lambda_1(x_1 + x_2 - 1) + \lambda_2(-x_1 + 2x_2)$$

Now calculate

$$\nabla \mathcal{L} = \mathbf{0}$$

$$\nabla_{x_1,x_2,\lambda_1,\lambda_2} \mathcal{L}$$

$$\begin{bmatrix} 2x_1 + \lambda_1 - \lambda_2 \\ 2x_2 + \lambda_1 + 2\lambda_2 \\ x_1 + x_2 - 1 \\ -x_1 + 2x_2 \end{bmatrix} = \mathbf{0} \iff \begin{cases} x_1 = \frac{-\lambda_1 + \lambda_2}{2} \\ x_2 = \frac{-\lambda_1 - 2\lambda_2}{2} \\ \frac{-\lambda_1 + \lambda_2}{2} + \frac{-\lambda_1 - 2\lambda_2}{2} - 1 = 0 \\ \frac{\lambda_1 - \lambda_2}{2} - \lambda_1 - 2\lambda_2 = 0 \end{cases}$$

Solving we obtain the values

$$\lambda_1 = \frac{-10}{9}, \lambda_2 = \frac{2}{9}, x_1 = \frac{2}{3}, x_2 = \frac{1}{3}$$

Now we need to verify that the conditions hold in this case.

$$\begin{cases} h(\frac{2}{3}, \frac{1}{3}) = \frac{2}{3} + \frac{1}{3} = 0 \\ g(\frac{2}{3}, \frac{1}{3}) = \frac{-2}{3} + 2\frac{1}{3} = 0 \end{cases}$$

And we get a minimum value of $f(\frac{2}{3}, \frac{1}{3}) = -\frac{4}{9}$

1.2 Stochastic Process

The given coin is fair, and so we have $p = 1 - p = 0.5 = P(H) = P(T)$.

First we must consider the event of getting heads, H. It either happens on the first trial with chance p or with chance $1 - p$ it will expect $1 + \psi(H)$ We then have $\psi(H) = p \cdot 1 + (1 - p)(1 + \psi(H)) = 1 \iff \psi(H) = \frac{1}{p}$ Because $p = \frac{1}{2}$ (because it's a fair coin) we then have:

$$\psi(H) = 2$$

Since it's a fair coin, the same logic can be applied to T and so we have

$$\psi(T) = 2$$

To get k tails in a row, we first need to obtain -1 (recursion) followed by p chance of tails or, in the other case, followed by p chance of heads, and by the chance of K tails. kT is the event of getting k consecutive tails, $kT - 1$ of getting $k - 1$ tails in a row.

$$\psi(kT) = \frac{1}{2}(\psi(kT - 1) + 1) + \frac{1}{2}(\psi(kT - 1) + 1 + \psi(kT))$$

Which simplifies to:

$$\psi(kT) = 2\psi(kT - 1) + 2$$

We can show that $\psi(kT) = 2^{k+1} - 2$ is the closed form of the previous recurrence relation.

$$\psi(1) = 2^{1+1} - 2 = 2$$

which is correct, since $\psi(T) = 2$. Now for the induction step:

$$\psi(kT) = 2^{k+1} - 2$$

$$\psi(kT + 1) = 2\psi(kT) + 2 = 2(2^{k+1} - 2) + 2 = 2^{(k+1)+1} - 4 + 2 = 2^{(k+1)+1} - 2$$

To obtain the solution, we need to obtain $\psi(HkT)$, getting a heads followed by K tails. That can be easily obtained by:

$$\psi(HkT) = \psi(H) + \psi(kT) = 2 + 2^{k+1} - 2 = 2^{k+1}$$

And so the expected number of tosses is 2^{k+1}.

2 SVM

$$
\begin{array}{ll}
\min\limits_{w,b,\xi,\hat{\xi}} & \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}(\xi_i + \hat{\xi}_i) = f(w,b,\xi,\hat{\xi}) \\
s.t. & yi \leq w^T \cdot x_i + b + \epsilon + \xi_i \iff y_i - w^T \cdot x_i - b - \epsilon - \xi_i \leq 0 \\
& yi \geq w^T \cdot x_i + b - \epsilon - \xi_i \iff -y_i + w^T \cdot x_i + b - \epsilon - \xi_i \leq 0 \\
& \xi_i \geq 0 \iff -\xi_i \leq 0 \\
& \hat{\xi}_i \geq 0 \iff -\hat{\xi}_i \leq 0 \\
& i \in \{1,\dots,N\}
\end{array}
$$

We have the Lagrangian function $\mathcal{L}$ which satisfies strong duality and so the solutions to the primal and the dual must satisfy the KKT conditions.

$$
\mathcal{L}(w,b,\xi,\hat{\xi},\alpha,\beta,\gamma,\delta) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}(\xi_i + \hat{\xi}_i) + \sum_{i=1}^{N}\alpha_i(y_i - w^T \cdot x_i - b - \epsilon - \xi_i)
$$

$$
+ \sum_{i=1}^{N}\beta(-y_i + w^T \cdot x_i + b - \epsilon - \xi_i) + \sum_{i=1}^{N}\gamma(-\xi_i) + \sum_{i=1}^{N}\delta(-\hat{\xi}_i)
$$

We first start by calculating $\nabla\mathcal{L} = 0$.

$$
\nabla\mathcal{L} = \begin{bmatrix} \frac{d\mathcal{L}}{dw} \\ \frac{d\mathcal{L}}{db} \\ \frac{d\mathcal{L}}{d\xi} \\ \frac{d\mathcal{L}}{d\hat{\xi}} \end{bmatrix} = \begin{bmatrix} w - \sum_{i=1}^{N}\alpha_i \cdot x_i + \sum_{i=1}^{N}\beta_i \cdot x_i \\ -\sum_{i=1}^{N}\alpha_i + \sum_{i=1}^{N}\beta_i \\ \mathbf{C} - \alpha - \gamma \\ \mathbf{C} - \beta - \delta \end{bmatrix} = \mathbf{0}
$$

$$
\iff
$$

$$
\begin{cases}
w = \sum_{i=1}^{N}(\alpha_i - \beta_i)x_i \\
\sum_{i=1}^{N}\beta_i = \sum_{i=1}^{N}\alpha_i \\
\mathbf{C} = \alpha + \gamma \\
\mathbf{C} = \beta + \delta
\end{cases}
$$

Now we can substitute these expressions in the original function.

$$
\mathcal{L}(w,b,\xi,\hat{\xi},\alpha,\beta,\gamma,\delta) =
$$

$$
\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \beta_i)(\alpha_j - \beta_j)\mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{N}yi(\alpha_i - \beta_i) - \sum_{i=1}^{N}\epsilon(\alpha_i + \beta_i)
$$

$$
-\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \beta_i)(\alpha_j - \beta_j)\mathbf{x}_i^T\mathbf{x}_j + b(\sum_{i=1}^{N}\beta_i - \sum_{i=1}^{N}\alpha_i) + \xi(\mathbf{C} - \mathbf{C}) + \hat{\xi}(\mathbf{C} - \mathbf{C})
$$

$$
= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \beta_i)(\alpha_j - \beta_j)\mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{N}y_i(\alpha_i - \beta_i) - \sum_{i=1}^{N}\epsilon(\alpha_i + \beta_i)
$$

Then the dual problem formulation is given by:

$$\begin{aligned} \max_{\alpha,\beta} \quad & -\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i-\beta_i)(\alpha_j-\beta_j)\mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{N} y_i(\alpha_i-\beta_i) - \sum_{i=1}^{N}\epsilon(\alpha_i+\beta_i) \\ s.t. \quad & 0 \le \alpha \le C \\ & 0 \le \beta \le C \\ & \textstyle\sum_{i=1}^{N}\beta_i = \sum_{i=1}^{N}\alpha_i \end{aligned}$$

3 Deep Neural Networks

The best configuration found for the spiral data set is the following:

Features	$X_1, X_2, sin(X_1), sin(X_2)$
Number Hidden Layers	2
Hidden layers sizes	6, 2
Learning rate	0.03/0.1
Activation function	$Tanh$
Regularization	None

First of all, the features layer. At first no feature was added, having only X_1, X_2 as the features which, obviously didn't work, as the NN wasn't getting enough information to process the nodes. A quick analysis allowed to understand how data was distributed, and, by being a spiral, that it would be slowly getting away from the center, requiring to change along the axis, but at the same time it would be having a circular motion. Therefore, each axis would have a $X_n \cdot sin(X_n)$ type of movement, concluding that it was required to use the $X_1, X_2, sin(X_1), sin(X_2)$ data as the input. *cossine*, although not present, wouldn't be a good choice since points start at (0,0).

Second, the hidden layers structure. The structure that allowed for more precision and for stability was having 5 neurons in one layer. Less than 5 didn't reduce the error under 0.1, and the decrease was slowly with more in just 1 layer. After adding an extra layer with a low amount of neurons, 2 (using just 1 resulted in bad accuracy), the convergence speed increased. The more the first layer increased, the more overfit it got. 6 was the best one as sometimes 5 layers didn't converge and resulted in some weird results more often than with 6.

Third, the learning rate, which affected not only the speed with which the error rate decreased to small values, and also dictated if the convergence would happen. Either 0.03 or 0.1 worked as reasonable values, with 0.1 being preferred for quickly reaching better values, but 0.03 was able to reach stable convergence more often. A tradeoff is required.

For the activation function, two are good: Tanh and sigmoid. Linear and ReLU both were unable to reach acceptable error rates. Between Tanh and Sigmoid, Tanh converged more quickly and is therefore preferred.

For last, the regularization, not using any was the best result. Both L1 and L2 in all values stopped the NN from converging to a good error rate and only got worse results. No regularization gave the best results.

4 IRLS for Logistic Regression

The IRLS formula for weight update is $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - H^{-1}\nabla_w \mathcal{L}(\boldsymbol{w}_t)$, given that we're required to maximize $\mathcal{L}$. When using regularisation it is needed to add a new factor. We then have a new formula to maximize:

$$\max_w -\frac{\lambda}{2}||\boldsymbol{w}||_2^2 + \mathcal{L} = \max_w l(w)$$

And so it is needed to derive both its gradient and Hessian matrix:

$$\nabla_w l(w) = -\lambda \boldsymbol{w}_t + \nabla\mathcal{L} = -\lambda \boldsymbol{w}_t + \boldsymbol{X}(\mathbf{y} - \boldsymbol{\mu})$$
$$H = \nabla_w \nabla_w l = -\lambda \boldsymbol{I} - \boldsymbol{X}\boldsymbol{R}\boldsymbol{X}^T$$

We now get the following expression to update the weights:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + (\lambda \boldsymbol{I} + \boldsymbol{X}\boldsymbol{R}\boldsymbol{X}^T)^{-1}(-\lambda \boldsymbol{w}_t + \boldsymbol{X}(\boldsymbol{y} - \boldsymbol{\mu}))$$

Some things worth mentioning regarding the implementation of the code:

1. Sometimes it was not possible to do the inversion as the matrix was invertible. It was required to use a specific numpy method, **np.linalg.pinv**, which allowed to successfully compute the matrix.

2. The data was given with labels $\forall i \in \{1, ..., N\}, y_i \in \{-1, 1\}$ and so it was necessary to convert the labels into $\{0, 1\}$.

3. It was avoided the creation of matrix R due to its size of 32k by 32k. A property of diagonal matrixes was used, and so it is only necessary to multiply each row of matrix X by the element of that line of R.

4. Initially, to find the convergence, it was used the following stop condition:

   $$||w_t - \nabla l|| < k$$

   Where k is an arbitrary constant that works as a threshold. Unfortunately, this method proved very strict and resulted in many iterations (if $\lambda = 0$ then it would never converge unless the threshold was 2 which is a big value for a threshold), which increased the time to train the model by a lot ($\approx$ 3x). As a new stopping condition, it is now stopping after an arbitrary number of iterations without improving the accuracy (default value used was 5).

5. The multiplication $\boldsymbol{X}\boldsymbol{R}\boldsymbol{X}^T$ is heavy due to the size of the matrixes, and so it was decided to use a sparse matrix in CSR format to accelerate the computation speed. Further work in this area can be done.

Starting by the prediction accuracy, figure 1 proves useful for this and it is noticeable that the accuracy changes very little even when using a wide range of values of λ.

	0	0.001	0.003	0.01	0.03	0.1	0.3	1	2	3	4	5	6	7	8	9	10	11	12	13
Test	0.8499	0.8499	0.8499	0.8499	0.8500	0.8499	0.8498	0.8499	0.8500	0.8499	0.8498	0.8503	0.8503	0.8504	0.8504	0.8501	0.8503	0.8500	0.8501	0.8501
Training	0.8491	0.8491	0.8491	0.8491	0.8491	0.8491	0.8492	0.8491	0.8491	0.8488	0.8488	0.8490	0.8488	0.8487	0.8486	0.8487	0.8487	0.8485	0.8486	0.8485

Figure 1: Accuracy of a test set given a regularization value

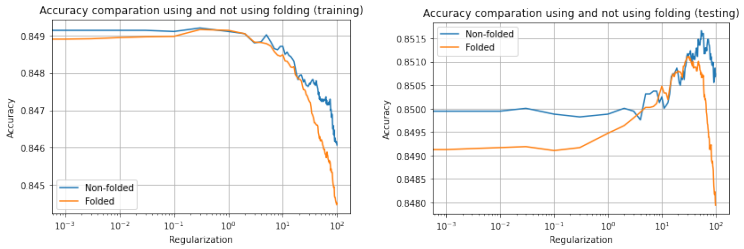

Figure 2: Accuracy measuring of the usage of folding technique

To further investigate the problem, it was used folding to check the accuracy. Figure 2 allows us to further analyse this. Because the values of λ weren't giving enough information, I decided to upgrade the values up to 100 to see it decreasing and make sure that the program was correctly implemented, although they don't make any difference. The change in accuracy is so low, even for such high values of lambda, that the conclusion one can take from this is that regularization doesn't affect this model in terms of accuracy. Nevertheless, the values to which the model predicts the best is $\lambda = 60$ with a test accuracy of 0.8517. Due to the strange values obtained with the generalization, the test set of folding wasn't used, and it was directly tested with the full training set and the test set. All k-folds assumed $k = 10$.

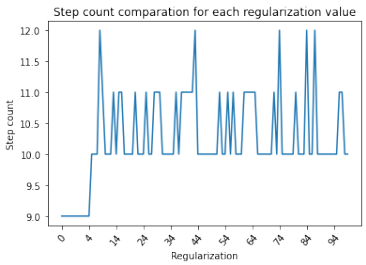

Figure 3: Step count until convergence

Regarding now the step size (Figure 3) and norm size (Figure 4) it can be seen that increasing the regularization factor resulted in a higher step counter for $\lambda > 4$ and that the norm size also decreased greatly up to $\lambda = 4$. After this value, the step count ranges between [10,12] and the weights' norm decreases very slowly.

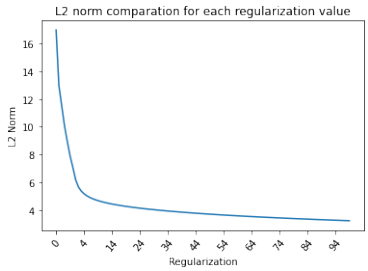

Figure 4: Norm size of weights vector

To conclude, I found that the model doesn't benefit much from regularization. A standard implementation of IRLS without the regularization factor behaves the same and converges as fast or faster depending on the regularization used (9 steps only). The accuracy is very similar and the folding also shows that the model generalizes well for a generalization factor of $\lambda = 0$.