AIML - CS 337

Lecture 7: Linear Regression- Regularization and Stability

August 29^{th}, 2022

Lecturer: Abir De *Scribe: Group 13, Group 14*

In the previous lecture, we briefly discussed the regression problem and explained the different types of loss functions we can use to train a regression model. This lecture will be a continuation of the linear regression problem and we will introduce a very important technique to improve loss functions called regularization. In the later part of this lecture, we will learn about a new feature of models called "stability".

1 Choosing between loss functions: MSE vs Absolute Error

In the previous class, we also discussed the comparison between MSE loss function and the absolute error function. We had discussed that minimizing the MSE loss is akin to obtaining the maximum likelihood estimate for the target function f, assuming that the noise in the y was sampled from a Gaussian distribution.

Similarly, minimizing the absolute error loss is the same as obtaining the MLE for the target function given that the noise was sampled from a Laplacian Distribution.

The user prefers the absolute error function when the user wants to maximize the number of correct predictions, but does not care much about the error when a prediction is wrong. On the other hand, the MSE loss function is preferred when the difference between the actual value and the predicted value is also of significance.

2 Regression

Continuing where we left in the last class, we have the following loss function which needs to be minimized to get optimal parameters for the regression problem:

$$L(\boldsymbol{w}, b) = \sum_{i \in D} (y_i - \boldsymbol{w}^T \boldsymbol{x_i} - b)^2$$

$$\text{s.t.} \quad y_i \in \mathbb{R}, \quad \boldsymbol{x_i} \in \mathbb{R}^d \quad i.e. \quad \boldsymbol{x_i} = \begin{bmatrix} x_i^1 & x_i^2 & \cdots & x_i^d \end{bmatrix}^T$$

Here, dataset $D = \{\boldsymbol{x_i}, y_i\}$ is the set of all points over which the analysis is done, $\boldsymbol{x_i}$ is the input vector for i^{th} sample and d is the number of features in each vector $\boldsymbol{x_i}$. y_i is the **label** associated with each **sample** $\boldsymbol{x_i}$ and **weight** $\boldsymbol{w} \in \mathbb{R}^d$ is the vector of weights assigned to each individual feature.

Observe that we can avoid explicitly mentioning b in the loss function, since it is a scalar and a modification of $\boldsymbol{x_i}$ can incorporate it. Thus our minimization problem reduces to:

$$\begin{aligned}
\min_{\boldsymbol{w}} L'(\boldsymbol{w}) = & \quad \min_{\boldsymbol{w}} \quad \sum_{i \in D} (y_i - \boldsymbol{w}^T \boldsymbol{x_i})^2 \\
= & \quad \min_{\boldsymbol{w}} \quad \sum_{i \in D} (y_i^2 - 2\boldsymbol{w}^T \boldsymbol{x_i} y_i + (\boldsymbol{w}^T \boldsymbol{x_i})^2) \\
= & \quad \min_{\boldsymbol{w}} \quad (\vec{\boldsymbol{y}} - \boldsymbol{X}\boldsymbol{w})^T (\vec{\boldsymbol{y}} - \boldsymbol{X}\boldsymbol{w}) \\
= & \quad \min_{\boldsymbol{w}} \quad (\vec{\boldsymbol{y}}^T \vec{\boldsymbol{y}} - 2\vec{\boldsymbol{y}}^T \boldsymbol{X}\boldsymbol{w} + \boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{w})
\end{aligned}$$

$$\text{where} \quad \vec{\boldsymbol{y}} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} \boldsymbol{x_1}^T \\ \boldsymbol{x_2}^T \\ \vdots \\ \boldsymbol{x_n}^T \end{bmatrix} \text{ and } \quad n = |D|$$

On differentiating w.r.t $\boldsymbol{w}$ and equating the *gradient* to 0, we get:

$$\begin{aligned}
\nabla_{\boldsymbol{w}} L'(\boldsymbol{w}) = 2\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \vec{\boldsymbol{y}}) &= 0 \\
\boldsymbol{X}^T \vec{\boldsymbol{y}} &= \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{w} \\
\boldsymbol{w} &= (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \vec{\boldsymbol{y}} \\
\boldsymbol{w} &= (\boldsymbol{X}^T \boldsymbol{X})^{-1} \sum_{i \in D} y_i \boldsymbol{x_i}
\end{aligned}$$

Thus, we have an expression for the optimal $\boldsymbol{w}$ ($\boldsymbol{w}^*$). Observe: we need $\boldsymbol{X}^T\boldsymbol{X}$ to be invertible for this model to work. In the following subsection, we will discuss this in detail.

2.1 Invertibility of $\boldsymbol{X}^T\boldsymbol{X}$

Note that, $\boldsymbol{X}^T\boldsymbol{X}$ will be invertible if and only if the determinant of the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is non-zero, $i.e.$, the matrix is non singular.

We know that each $\boldsymbol{x_i}$ is an i.i.d (identical, independent draw) from a continuous distribution. Thus for $n > d$ where $\boldsymbol{X}$ is of dimensions $n \times d$, we can claim that $|\boldsymbol{X}^T\boldsymbol{X}|$ has a continuous distribution. Hence, the probability that determinant of $\boldsymbol{X}^T\boldsymbol{X}$ will take exactly 0 value is 0, $i.e.$

$$P(|\boldsymbol{X}^T\boldsymbol{X}| = 0) = 0$$

Intuitively, we can also see that the matrix X is highly likely to be full rank. A dxN matrix (where $N \geq d$) is full rank if atleast d columns out of the N columns are linearly independent.

At the same time, note that the probability of $|\boldsymbol{X}^T\boldsymbol{X}|$ lying within some ϵ near 0 is not zero $i.e.$ $P(|\boldsymbol{X}^T\boldsymbol{X}| < \epsilon) \neq 0$

In mathematics, a condition number is a number representative of the change of an output proportionate to a change in the input of a function. For example, if a small change in the input results in a small change in the output, the function produces a small condition number and is said to be well-conditioned. Alternatively, if a small change in the input results in a large change in the output, the function produces a large condition number and is defined as ill-conditioned. The condition number is mathematically given as the ratio of the maximum and minimum eigenvalues.

Poorly conditioned matrix A is a matrix with a high condition number. A^{-1} amplifies input errors. Small errors in x can change the output of $A^{-1}x$ rapidly.

Definition 2.1. Condition number of a matrix A is the ratio of its minimum eigenvalue to maximum eigenvalue

$$Cond(\boldsymbol{A}) = \frac{min(eigen(\boldsymbol{A}))}{max(eigen(\boldsymbol{A}))}$$

Condition number of a matrix **A** is a good indicator of the invertibility of the matrix. A high condition number results in a well conditioned matrix which increases the chances of **A** being invertible, whereas, a low condition number results in an ill conditioned matrix, $i.e.$ reduced chances of invertiblilty.

We want to ensure that the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is well-conditioned. One possible approach for this is replacing $\boldsymbol{X}^T\boldsymbol{X}$ with $\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}$, $i.e.$

$$\boldsymbol{w}* = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\vec{\boldsymbol{y}}$$

Note here that such a transformation changes the condition number as follows:

$$\begin{aligned}
&Cond(\boldsymbol{X}^T\boldsymbol{X}) \rightarrow Cond(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}) \\
&\frac{min(eigen(\boldsymbol{X}^T\boldsymbol{X}))}{max(eigen(\boldsymbol{X}^T\boldsymbol{X}))} \rightarrow \frac{min(eigen(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}))}{max(eigen(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}))} \\
&\frac{a}{b} \rightarrow \frac{a+\lambda}{b+\lambda} \quad \text{(assuming} \quad Cond(\boldsymbol{X}^T\boldsymbol{X}) = \frac{a}{b})
\end{aligned}$$

Thus, the $\lambda\boldsymbol{I}$ term adds a lower bound to the value of condition number improving the chances of invertibility of desired matrix.

Also, we find that this type of $\boldsymbol{w}*$ is the solution to the following loss minimization problem:

$$\min_{\boldsymbol{w}} \quad \sum_{i \in D}(y_i - \boldsymbol{w}^T\boldsymbol{x_i})^2 + \lambda\|\boldsymbol{w}\|^2$$

More about this in the next section.

3 Regularization

Theoretically, **regularization** is a popular technique used to reduce errors by fitting the function appropriately on the given training set and to avoid overfitting

The most common regularization techniques are:

1. L1 regularization
2. L2 regularization
3. Dropout regularization

Here we will discuss L2 regularization.

We saw an example of L2 regularization at the end of the last section trying to ensure that the matrix whose inverse we are calculating is well conditioned. Following was the loss minimization expression for it:

$$\min_{\boldsymbol{w}} \quad \sum_{i \in D} (y_i - \boldsymbol{w}^T \boldsymbol{x_i})^2 + \lambda \|\boldsymbol{w}\|^2$$

Here, $\lambda \|\boldsymbol{w}\|^2$ is called the **regularizer** term. In the following sections, we discuss some **advantages** of regularization.

3.1 Ensures Well-Conditioned Matrix

As we saw in **Section 1.1**, if we add the L2 regularizer term to our Loss function then the solution for the optimal $\boldsymbol{w}^*$ will be:

$$\boldsymbol{w}* = (\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^T \vec{\boldsymbol{y}}$$

Instead of:

$$\boldsymbol{w}* = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \vec{\boldsymbol{y}}$$

The regularizer adds a lower bound to the condition number of the desired matrix matrix, thus increasing the chances of invertibility of $\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I}$ compared to $\boldsymbol{X}^T \boldsymbol{X}$.

3.2 Prevents Overfitting

Overfitting occurs when the model is constrained to the training set and not able to perform well on the test set, here the gap between the training error and testing error is large.

Example:
Let us look at a regularization problem with the following loss function:

$$\min_{\boldsymbol{w}} \quad \sum_{i \in D} (y_i - \boldsymbol{w}^T \boldsymbol{x_i})^2$$

Suppose the number of features in each data point $\boldsymbol{x_i}$ in the dataset increases significantly say from d_1 to d_2 such that $d_2 > d_1$. In response to this, the dimensions of our learnable parameter $\boldsymbol{w}$ also change from $d_1 \times 1$ to $d_2 \times 1$. Thus, the overall model will become more **complicated**.
Also, more features will help bring down the training loss and will increase the training accuracy, but since the model has been trained on the same training set with more features, it loses **generalization**. The **test accuracy** will become significantly **low**er due to this overfitting of model on training data.

3.2.1 How to avoid overfitting?

The solution is again regularization. On adding regularizer term to our original L2-loss, we obtain the new loss function as follows:

$$\min_{\boldsymbol{w}} \quad \sum_{i \in D} (y_i - \boldsymbol{w}^T \boldsymbol{x_i})^2 + \lambda \|\boldsymbol{w}\|^2$$

If the number of features of $\boldsymbol{x_i}$ increases then the dimension of $\boldsymbol{w}$ will also increase, and hence the dimension of $\boldsymbol{w}*$ will increase. Due to this increase in dimension the value of the regularizer term $\lambda \|\boldsymbol{w}\|^2$ rises, increasing the overall loss value.

So, the training loss can be greater with a higher number of features compared to one obtained with less number of features $i.e.$ with the regularized loss function, a model with higher number of features is not necessarily a better model for the training set. This helps prevent overfitting of the model.

3.2.2 Alternate technique to avoid overfitting

If regularization is not allowed, how to decide the optimal number of features to train the model?

We can use the method discussed here to find the optimal subset of features to be used for a training set with a large number of extractable information/features. Let the total number of features be d, the algorithm follows below:

- Separate a portion of the training data as **validation set**.
- Iterate over all subsets $\boldsymbol{s}$ of features with cardinality $|\boldsymbol{s}| = d' \leq d$

- Compare across all these subsets by finding their loss on the validation set. Choose the set of features that gives minimum loss on the validation set.

The above steps can be summarized by the following loss function:

$$\min_{\boldsymbol{w_s}, \boldsymbol{s}} \quad \sum_{i \in D} (y_i - \boldsymbol{w}_s^T \boldsymbol{x_{i,s}})^2$$

Here, $|\boldsymbol{s}| \leq d$ and $\boldsymbol{x_{i,s}}$ refers to the subset of $\boldsymbol{x_i}$ which has features corresponding to the set $\boldsymbol{s}$.

But this optimization problem is **NP Hard** due to the exponential search space involving iteration over all subsets.

4 Stability

A model is called stable if on addition of a new data point to training dataset, the learnable parameter $\boldsymbol{w}$ does not change very much.

Mathematically speaking, let D be our initial training dataset and $D \cup k$ be the new training dataset with the addition of a new data point k. Then we define the following quantity:

$$l = \|\boldsymbol{w}^*_{(D)} - \boldsymbol{w}^*_{(D \cup k)}\|$$

Stability is ensured when this l is small. This means that on addition/modification/removal of a data point, the learnable parameter $\boldsymbol{w}$ did not $change$ very much.

Mathematically, we call a model stable if l follows this condition:

$$l = \|\boldsymbol{w}^*_{(D)} - \boldsymbol{w}^*_{(D \cup k)}\| < \epsilon, \quad \text{where} \quad \epsilon \sim O(\frac{1}{n}) = O(\frac{1}{|D|})$$

where $|D| = n$ is the size of the complete dataset.

We can prove that if the model follows the stability condition stated above then,

$$\text{test-case error} \leq \alpha\epsilon + \beta$$

where, α and β are constants that are not in our control but ϵ is decided by model predictions.

The constant β is decided by the quality of dataset provided. For example if, $\boldsymbol{x_i}$ is sampled from a normal distribution $\mathcal{N}(0, 1)$ and corresponding y_i are sampled from uniform distribution $U(0, 1)$, then the value of constant β will be higher since dataset is randomly created.

4.1 Advantages of Stability

4.1.1 Smaller dataset needed for training

When a model is trained on a small dataset D but is stable, then on addition of a new data point **k** the optimal learnable parameter $\boldsymbol{w}*$ does not change a lot $i.e.$, the previously learned $w^*(D)$ is similar to $w^*(D \cup k)$ even on addition of a new unseen point.

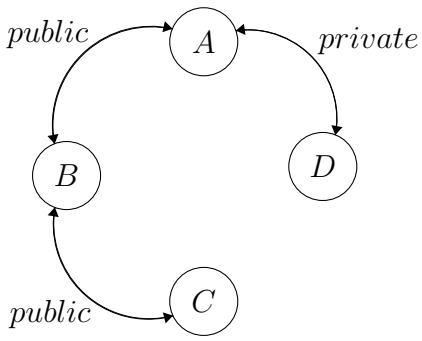
Thus, our model behaves as if it has been trained on the larger set and will also work well on unseen test samples.

4.1.2 Prevents leak of information to adversary

This is one of the information security concerns of the model. Suppose on addition of a data point k, the model's learnable parameter $\boldsymbol{w}^*$ changes a lot. This can reveal important characteristics of the model to the adversary which can be used to predict label y_i of the model on input x_i with some confidence and thus an adversary can exploit the model.

To understand this point better, let us present a small example:

Suppose A, B, C, D are 4 people on Facebook. As shown in the diagram below, Person A has a private Facebook connection with D which is not visible to others. All other connections are publicly visible.



Now suppose that the Facebook algorithm recommends Person D's profile to Person C, then C can estimate with certain confidence that either of B or A has a private connection with D. If Person C get to know that Person B has no private connection then Person C can say with certainty that Person A has a private connection with Person D.

Thus, the recommendation algorithm of Facebook's model could have been designed better to ensure data privacy in such a case.

4.2 Individual and Overall Loss

Definition 4.1. Individual loss: $l(\boldsymbol{w}^T\boldsymbol{x_i}, y_i) = (y_i - \boldsymbol{w}^T\boldsymbol{x_i})^2 + \lambda\|\boldsymbol{w}\|^2$

Definition 4.2. Overall loss: $\sum_{i\in D} l(\boldsymbol{w}^T\boldsymbol{x_i}, y_i)$

Notice the subtle difference between the regularized l2-loss introduced in section 2 and the overall loss function described here. In the overall loss definition, $\lambda\|w\|^2$ will be repeated $|D|$ times where $|D|$ is the size of the dataset while in the regularized l2-loss, it is included exactly once.

The relation between λ (for overall loss function of D) and λ' (for overall loss function of $D \cup k$):

$$\lambda' = \lambda\frac{|D \cup k|}{|D|}$$

4.3 Bounding the *change* in Overall Loss

We will attempt to obtain upper and lower bounds for the quantity $\|L(D\cup k, \boldsymbol{w}^*_{(D\cup k)}) - L(D, \boldsymbol{w}^*_{(D)})\|$. Notice that this expression seems to model the $change$ in overall loss on addition of a new datapoint k to D, $i.e.$:

$$L(D, \boldsymbol{w}^*_{(D)}) = \sum_{i\in D} l(\boldsymbol{w}^*_{(D)}\boldsymbol{x_i}, y_i) \tag{1}$$

To upper bound this quantity we will use Lipchitz Continuity or the 'Lipchitzness' of overall loss function.

A function is Lipschitz continuous if there exists a constant L such that:

$$\|f_x - f_y\| \leqslant L\|x - y\| \quad \forall x, y \tag{2}$$

Thus, we get:

$$\begin{aligned}
\|L(D \cup k, \boldsymbol{w}^*_{(D\cup k)}) - L(D, \boldsymbol{w}^*_{(D)})\| &= \|L(D, \boldsymbol{w}^*_{(D\cup k)}) + L(k, \boldsymbol{w}^*_{(D\cup k)}) - L(D, \boldsymbol{w}^*_{(D)})\| \\
&= \|L(D, \boldsymbol{w}^*_{(D\cup k)}) - L(D, \boldsymbol{w}^*_{(D)}) + L(k, \boldsymbol{w}^*_{(D\cup k)})\| \\
&\leq \|L(D, \boldsymbol{w}^*_{(D\cup k)}) - L(D, \boldsymbol{w}^*_{(D)})\| + \|L(k, \boldsymbol{w}^*_{(D\cup k)})\|
\end{aligned}$$

Which implies:

$$\begin{aligned}
&\|L(D, \boldsymbol{w}^*_{(D\cup k)}) - L(D, \boldsymbol{w}^*_{(D)}) + L(k, \boldsymbol{w}^*_{(D\cup k)})\| \\
&\quad\leq \|\sum_{i\in D} l(\boldsymbol{w}^*_{(D\cup k)}\boldsymbol{x_i}, y_i) - \sum_{i\in D} l(\boldsymbol{w}^*_{(D)}\boldsymbol{x_i}, y_i)\| + \|L(k, \boldsymbol{w}^*_{(D\cup k)})\| \quad \text{from (1)} \\
&\qquad\leq \sum_{i\in D} \|l(\boldsymbol{w}^*_{(D\cup k)}\boldsymbol{x_i}, y_i) - l(\boldsymbol{w}^*_{(D)}\boldsymbol{x_i}, y_i)\| + \|L(k, \boldsymbol{w}^*_{(D\cup k)})\| \\
&\qquad\leq \sum_{i\in D} \alpha\|\boldsymbol{w}^*_{(D\cup k)} - \boldsymbol{w}^*_{(D)}\| + \|L(k, \boldsymbol{w}^*_{(D\cup k)})\| \quad \text{from (2)} \\
&\qquad= \alpha|D|\|\boldsymbol{w}^*_{(D\cup k)} - \boldsymbol{w}^*_{(D)}\| + \|L(k, \boldsymbol{w}^*_{(D\cup k)})\|
\end{aligned}$$

Hence we have finally:

$$\|L(D \cup k, \boldsymbol{w}^*_{(D\cup k)}) - L(D, \boldsymbol{w}^*_{(D)})\| \leq \alpha|D|\|\boldsymbol{w}^*_{(D\cup k)} - \boldsymbol{w}^*_{(D)}\| + \|L(k, \boldsymbol{w}^*_{(D\cup k)})\|$$

In the above proof we have used **Triangle inequality** repeatedly in the initial steps and finally used Lipschitz continuity condition on the loss function with **Lipchitz constant $\boldsymbol{\alpha}$**. We can use this condition on our loss function since it is differentiable and differentiability implies Lipchitz continuity.

The next lecture will carry forward this discussion of Lipchitzness for proving an upper bound and will introduce a lower bound for the $change$ in overall loss using convexity of the loss function.