AIML - CS 337

Lecture 6: Loss Function and Models in Regression

25 August 2022

Lecturer: Abir De *Scribe: Group 11 and Group 12*

In this lecture, we lay the foundations of regression, as well as initiate a discussion on the various loss functions used to judge the quality of a model, and how to choose appropriate loss functions based on the noise present and other circumstances.

1 Regression : Housing Prices Scenario

We begin with the problem of determining the price of a house based on the various *features* of the house, such as it's area, location etc. Assuming that all the features of the house can be quantized, we represent the *feature vector* of the house by the column vector $\boldsymbol{x} = [\text{Area} \ \ \text{Location} \ \ \ldots]^T \in \mathbb{R}^{n\times 1}$, where we assume we have n features. The price of the house is represented by the scalar y, and thus our **regression problem** asks for a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ which *closely simulates* the prices $\{y_i\}_{i\in D}$ given the feature vectors $\{\boldsymbol{x_i}\}_{i\in D}$.

If we constrain the space of functions $f : \mathbb{R}^n \mapsto \mathbb{R}$ such that f is linear in the feature variables, the problem is then termed as **linear regression**. f is of the form

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$$

where $w \in \mathbb{R}^n, b \in \mathbb{R}$ are *parameters* of f.

Note that deciding which features are useful in determining the price of the house and how to quantize them is another problem in itself, called the *feature selection problem.* We ignore this problem for now and dive into what is meant by "closely simulates" by exploring a couple of loss functions.

Devising a Loss Metric

To say that a function f *closely simulates* given data, we must have some form of error/loss metric that we try to minimize, such that the predictions from f are close to the true values.

Let $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ and **D** be dataset $\{\boldsymbol{x}_i, y_i\}_{i=1}^{m}, \boldsymbol{x}_i \in \mathbb{R}^{n\times 1}, y_i \in \mathbb{R}$. The error/loss function can be denoted as $F(f, D)$. For the current problem of regression, we assume $\{y_i\}$ to be continuous valued. Few candidates for the function could be :

- $\sum_i (y_i - f(\boldsymbol{x}_i))$

  Note: There is a glaring problem with this function in that we can minimize this infinitely towards $-\infty$ without actual predicting values near y_i

- $\sum_i |y_i - f(\boldsymbol{x}_i)|$
- $\sum_i (y_i - f(\boldsymbol{x}_i))^2$
- $\sum_i e^{(y_i - f(\boldsymbol{x}_i))^2}$
- $\sum_i |y_i - f(\boldsymbol{x}_i)|^3$ *This is rarely used*

In general, decision of the loss function to choose depends on

- Distribution of errors
- Demands of the problem (For instance, extent of penalization for outliers)

We now explore some commonly used loss functions in detail -

Mean Squared Loss

$$F(f, D) := \sum_{\{\boldsymbol{x}_i, y_i\} \in D} (y_i - f(\boldsymbol{x_i}))^2$$

One may have seen this loss in many places and for good reason: **Minimizing this loss is equivalent to maximizing the likelihood $P(\{y_i\}_{i\in D} | \{\boldsymbol{x_i}\}_{i\in D})$ under the assumption there is a Gaussian noise between $f(\boldsymbol{x_i})$ and y_i.**
Let us prove this:
Assuming that y indeed varies over $\boldsymbol{x}$ according to the distribution f, but with an iid Gaussian (with mean 0 and variance σ^2) noise ε, one notes that $P(y|\boldsymbol{x})$ is equivalent to asking how likely is it that the noise in y was sampled from a Gaussian distribution, and thus

$$P(\{y_i\}_{i\in D} | \{\boldsymbol{x_i}\}_{i\in D}) = \prod_{i\in D} P(y_i|\boldsymbol{x_i}) \propto \prod_{i\in D} e^{-\frac{(y_i - f(\boldsymbol{x_i}))^2}{2\sigma^2}} = \exp\left(\frac{-1}{2\sigma^2}\sum_{i\in D}(y_i - f(\boldsymbol{x_i}))^2\right)$$

where the first equality follows from an independence assumption. Then one can clearly see that minimizing $F(f, D)$ maximizes the likelihood that the prices indeed are drawn from the distribution we have modelled. An extra merit of above loss is that the gradient of the function becomes zero when the loss becomes zero.
As for the fact that we assumed our noise to be Gaussian, this is a standard assumption throughout machine learning: In unknown scenarios where the nature of the noise is unclear, assuming Gaussian noise generally works well in practical cases. The belief in the Gaussian nature of the noise is so deep-rooted, that one of the common techniques in machine learning to simulate noise is as follows

$$\varepsilon_{\boldsymbol{x}} \sim \mathcal{N}(0, \xi(\boldsymbol{x})) = \mathcal{N}(0, \sigma_{\boldsymbol{x}}^2)$$

where $\xi : \mathbb{R}^n \mapsto \mathbb{R}$ represents a neural network which takes as input $\boldsymbol{x}$ and returns the variance of the noise (assumed to be a mean 0 Gaussian RV) between our predictions and the ground reality.
However, the pre-eminence of Gaussian noise models (and consequently squared-error functions) does by no means preclude the possibility of other types of noises, and thus we shall explore another common error function, one based on the ℓ_1-norm.

The ℓ_1-norm loss

The ℓ_1-norm loss is formally defined as

$$F(f, D) := \sum_{i \in D} |y_i - f(\boldsymbol{x_i})|$$

One of the most apparent scenarios which comes to mind is that if we know that the noise in our model follows a **Laplacian distribution**, ie:- $\propto \exp\left(\frac{-|x-\mu|}{b}\right)$. Then similar to the proof given above, a ℓ_1-norm loss would maximize the likelihood of our model simulating the ground reality.

However, it isn't the case that the ℓ_1-norm loss arises only in such esoteric cases: Indeed, imagine a scenario where our loss should encourage sparsity in the final model; such scenarios commonly occur in the fields of image compression, compressed sensing, sparse graphs (in the context of social media relations), etc. Note that the mean squared error forces low errors on every point in our domain D, since any error anywhere is amplified by the square function and increases the overall loss. However, as mentioned above, in applications where sparsity is more of a priority, large errors in some parts of our domain can be forgiven as long as the error on most of the domain is very low, ie:- if one takes the difference of the ground reality and our predictions, then the difference **forms a sparse vector/matrix, ie:- a vector most of whose entries are 0**.

Given this background, one might wonder why don't we minimize the ℓ_0-norm instead, where the ℓ_0-norm of a vector is defined to be the *number of non-zero elements of the vector*. Indeed, a ℓ_0-loss exactly satisfies the sparsity requirement. Unfortunately, minimizing the ℓ_0-norm is a **NP-hard** problem in most scenarios, and thus is an intractable loss function. Interestingly, the ℓ_1-norm, under *reasonable regularity conditions*, yields almost the same optimal solution as the ℓ_0-norm would have yielded. This is a fact widely employed in the field of compressed sensing where ℓ_1-norms are routinely deployed as proxies for the ℓ_0-norms to induce sparsity in the model.

How to choose the correct loss function? The distribution of errors is not generally given and neither can be guessed directly, We can test out a loss function by training over a *training dataset* and testing the model performance over a *validation dataset* disjoint from the training dataset. We can then compare performance metrics to decide the best loss function.

The decision of choosing loss function depends on the user requirement too. For cases when we want a larger penalty for larger errors i.e. prevent blatant outliers, **Mean Squared Loss** function is better. Whereas, when we do not care about outliers as much, then $\boldsymbol{\ell_1}$ **Norm** is better.

2 Conclusion

In short, what noise model is to be deployed depends on the context and the requirements: If very little is known about the noise, and the requirements are generic, then the mean squared error usually works well. However, if the requirements are such that sparsity is required, ie:- almost perfect accuracy on most predictions at the cost of some big misses for a few, then the ℓ_1-norm loss is preferred.

Analytic Derivation of Linear Regression under MSE loss

Consider the linear regression function $f(\boldsymbol{x_i}) = \boldsymbol{w^T x_i} + b$, where $\boldsymbol{w}, \boldsymbol{x_i} \in \mathbb{R}^n$ and $b \in \mathbb{R}$.
Under a MSE loss, if we're to find the best model, then the problem can be stated mathematically as

$$(\boldsymbol{w}_*, b_*) := \arg\min \sum_{i \in D} (y_i - \boldsymbol{w^T x_i} - b)^2$$

It's not hard to show that if we assume that the $\boldsymbol{x_i}$'s are sampled from some mean zero distribution, then $\mathbb{E}[y] = b$, and thus one can set $b_* = \mathbb{E}[y]$.
Thus henceforth we'll assume $b = 0$. Thus

$$\boldsymbol{w}_* := \arg \frac{d}{d\boldsymbol{w}} \left(\sum_{i \in D} (y_i - \boldsymbol{w^T x_i})^2 \right) = 0$$

Side Note:
We can vectorize this summation by stacking the y_is and $\boldsymbol{x_i}$s, and obtain analytical solution using matrix calculus.
Suppose we have n features and d data points so that $\mathbf{X} \in \mathbb{R}^{n \times d}, \mathbf{y} \in \mathbb{R}^{1 \times d}$, then we'll have the solution as $\mathrm{w} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y}^\top$. To get to this solution, we make the assumption that $\mathbf{X}\mathbf{X}^\top$ is invertible. We can't proceed further without that assumption.

Next, let's focus on the case with just one data point. Here, $\boldsymbol{w}_* := \arg \dfrac{d}{d\boldsymbol{w}} \left((y - \boldsymbol{w^T x})^2\right) = 0$

Simplifying the equation $\frac{d((y - \boldsymbol{w^T x})^2)}{dw} = 0$ yields $\boldsymbol{x x^T w}_* = y\boldsymbol{x}$. However, without further assumptions, this is analytically intractable.

To circumvent this issue, we employ a neat trick: We minimize $(y - \boldsymbol{w^T x})^2 + \lambda\|\boldsymbol{w}\|_2^2$ instead, and hope that we'll get the right answer as $\lambda \to 0$. An interesting observation about this formulation is that if we assume the probability distribution of $\boldsymbol{w}$ to be **Gaussian** and maximize the posterior,

$$\max_{\boldsymbol{w_*}} P(y|\boldsymbol{x})P(\boldsymbol{w}) \propto \max_{\boldsymbol{w_*}} \prod_{i\in D} \exp\left(-\frac{(y_i - \boldsymbol{w^T x_i})^2}{2\sigma^2}\right) e^{-\frac{||\boldsymbol{w_*}||}{2\sigma_*^2}}$$
$$= \max_{\boldsymbol{w_*}} \exp\left(\frac{-1}{2\sigma^2}\sum_{i\in D}(y_i - \boldsymbol{w^T x_i})^2 \;-\; \frac{1}{2\sigma_*^2}\|\boldsymbol{w}\|_2^2\right)$$

We see that maximizing $P(y|\boldsymbol{x})P(\boldsymbol{w})$ likelihood is same as minimizing $\sum_{i\in D}(y_i - \boldsymbol{w^T x_i})^2 + \lambda\|\boldsymbol{w}\|_2^2$ with $\lambda = \frac{1}{2\sigma_*^2}$.

This is, in short, the **Bayesian interpretation** of *regularised* L2 regression.

The expression one gets on simplifying $\frac{d((y-\boldsymbol{w^T x})^2+\lambda\|\boldsymbol{w}\|_2^2)}{d\boldsymbol{w}} = 0$ is $(\lambda \boldsymbol{I} + \boldsymbol{x}\boldsymbol{x^T})\boldsymbol{w}_* = y\boldsymbol{x}$.One can then use the **Sherman-Morrison formula** to get that

$$\boldsymbol{w}_* = \frac{y}{\lambda}\left(\boldsymbol{I} - \frac{\boldsymbol{x}\boldsymbol{x^T}}{\boldsymbol{\lambda} + \|\boldsymbol{x}\|_{\boldsymbol{2}}^{\boldsymbol{2}}}\right)\boldsymbol{x}$$

Unfortunately, this expression doesn't converge as $\lambda \to 0$. However, as λ becomes small, $\boldsymbol{w}_*$ becomes approximately parallel to the vector $\boldsymbol{v} := \left(\boldsymbol{I} - \frac{\boldsymbol{x}\boldsymbol{x^T}}{\|\boldsymbol{x}\|_{\boldsymbol{2}}^{\boldsymbol{2}}}\right)\boldsymbol{x}$.