COMP 642 — Machine Learning Jan 17, 2023

Lecture 3 Linear Classifiers I

Lecturer: Anshumali Shrivastava Scribe By: Tianhong Gao, Yanlin Wu, Yiwei Lu, Zile Xiao

1 Supervised Machine Learning Process

Supervised machine learning is the most commonly used machine learning. It uses labeled datasets to train the algorithm which can be used to predict outcomes or to classify data into specific categories.

1.1 Datasets

The datasets used by supervised machine learning can be represented by n sets of input and output data $\{x_i, y_i\}_{i=1}^n$, where x_i represents input data which is a vector in d dimension and y_i which can be a vector or a scalar represents the output given x_i. For example, x_i can be features of a house such as the total size of the house, the number of bedrooms or the location of the house and y_i in this case will be the price of the house that we want to predict.

1.2 Models

Model can be seen as a function **F** with certain input feature variables x_{new} and the output will be the prediction result $F(x_{new}) = y_{new}$ we want. This function can be anything and the key point here is to find the "best" function to make the prediction.

1.3 Loss Function

Loss function is the method we use to measure the goodness of a function. It can be written as

$$L(F) = \frac{1}{n}\sum_{i=0}^{i=n} dist(F(x_i), y_i) \tag{1}$$

Here, $dist(F(x_i), y_i)$ is the difference between the model predicted result $F(x_i)$ and the real output value y_i, a better function will have smaller differences between these two values. Keep in mind that the distance calculations can vary depending on different problem types, therefore, we need to choose proper loss function first.

After we have our loss function, we can make comparison between two models F_1 and F_2 and the one with smaller loss function value will win. But here is a problem! What if one of the model is created by simply memorizing the training datasets? In this case, the loss function of this model will be 0 which means this model can not be beaten. In practice, the solution for this problem is to split existing datasets into two parts: training data and test data. The training data is used to develop the models and the test data is used to evaluate models later.

1.4 Training Process

Now we have a standard way to measure the goodness of a model, the next step is to minimize the loss function and find the best model with smallest loss function. This process is called **Training process**. However, we will quickly realize that it is impossible to find the best function among all possible functions. What we do is to restrict F to a specific family of functions which we think will possibly contain a reasonable function. Then, we can make comparisons between functions from the same function family such as linear.
Say we have functions written as

$$F_w(X) = w^T x \tag{2}$$

The loss function can be written as

$$L(F_w(x)) = \frac{1}{n}\sum_{i=0}^{i=n} dist(F_w(x_i), y_i) \tag{3}$$

The training process will be finding the w with the minimum loss function

$$w^* = \underset{w}{argmin}\frac{1}{n}\sum_{i=0}^{i=n} dist(F_w(x_i), y_i) \tag{4}$$

One way to think about the optimization process is through **Taylor Expansion** for small enough Δw

$$\begin{align} L(w + \Delta w) &\approx L(w) + L'(w)(w + \Delta w - w) + \frac{L''(w)}{2!}(w + \Delta w - w)^2 + ... \tag{5}\\ &= L(w) + \Delta w L'(w) + ... \tag{6}\\ &= L(w) + \Delta w \frac{\partial L}{\partial w} + ... \tag{7} \end{align}$$

where $\Delta w = \eta\vec{\delta}$ and η represents the scalar size of change and $\vec{\delta}$ indicates the direction of the change.

2 Optimization algorithm

2.1 Gradient Descent

Gradient descent (GD) is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. The idea is to take repeated steps in the opposite direction of the gradient of the function at the current point, because this is the direction of steepest descent.
In the machine learning process, we use GD to find the optimal parameter w^*. We start with an initial value w_0 and update w in the form of

$$w_t = w_{t-1} - \eta \nabla_w L \Big|_{w_{t-1}} \tag{8}$$

where η represents the step size and L is a loss function explained in section 1.4. In addition, the gradient of L with respected to w is calculated by doing the partial derivative to every dimension in weights vector:

$$\nabla_w L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \vdots \\ \frac{\partial L}{\partial w_n} \end{bmatrix} \tag{9}$$

2.2 Newton's Method

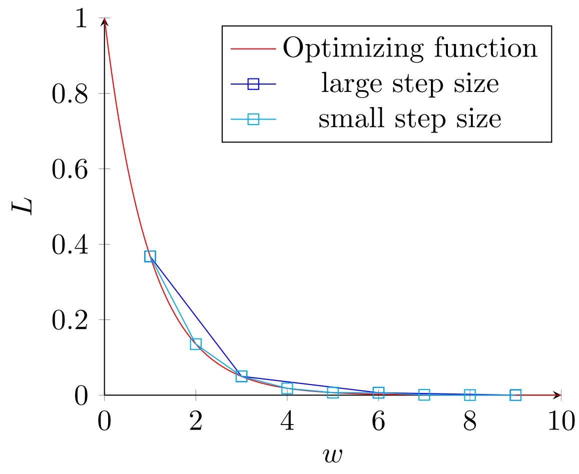
Newton's method gives us a more accurate w^* than GD does. As it uses a second-order function for optimization. For using Newton's method, we start with an initialization w_0 and update w in the form of:

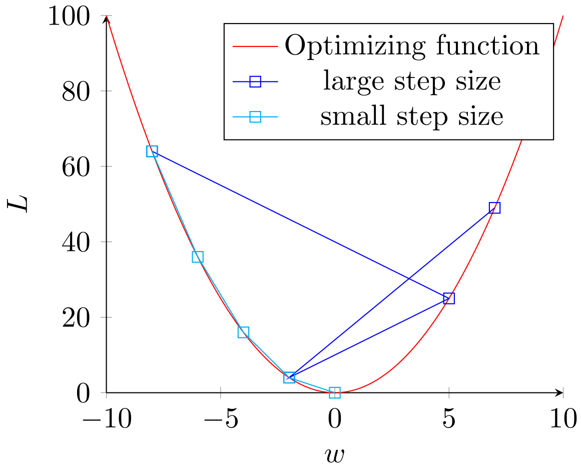
$$\begin{aligned} w_t &= w_{t-1} - \frac{\nabla_w L}{\nabla_w^2 L}\bigg|_{w_{t-1}} \\ &= w_{t-1} - H^{-1}\nabla_w L\bigg|_{w_{t-1}} \end{aligned} \tag{10}$$

where H^{-1} is the Hessian matrix of the loss function L.
But Newton's method is rarely used in machine learning applications as each iteration of Newton's method is more expensive computationally than simple gradient descent. Thus, there is a trade-off: while we need much fewer number of iterations to get to optimum (after good initialization), we pay much more per iteration.

2.3 Step size

The question on how to choose the proper step size for the optimization process is worth considering. A large step size may cause the oscillation in a function and a small one may take quite a long time to find the minimum. Therefore, one should make smart choices on choosing step size. Usually, once we have a flat, decreasing function (shown in the left graph below), such as negative log function, we can use a larger step size to speed up the process. Conversely, if we have a function such as quadratic (shown in the right graph below), we need to use a small step size to avoid oscillation.





On the other hand, find an optimal step size, η, is doable but not recommend. Since it is equally hard for a process to solve the $\arg\min_{\eta} f(x - \eta d)$ than to find the optimal $f_w(X)$.

3 Linear Classifier

3.1 Definition

A linear classifier is a supervised machine learning algorithm that separates data into different classes by finding the best linear boundary (hyperplane) that separates the data. Linear

Classifier has the form:

$$F_w(x) = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b \tag{11}$$
$$= w^Tx + b \tag{12}$$

Where w_i is the parameter we are trying to optimize for each x_i, x_i is the feature value of the input data, b is the bias value which allows for a more accurate representation of the data being modeled.

3.2 Linear Regression

3.2.1 Definition

Linear regression is a statistical method for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called simple linear regression. Linear Regression has the same form as (11) (12).

Given a training dataset

$$T = \{(x_1, y_1), (x_2, y_2) \ldots , (x_N, y_N)\} \tag{13}$$

Where, $x_i \in \mathbf{R}^n$ is the input, $y_i \in \mathbf{R}$ is the corresponding output, $i = 1, 2, \ldots , N$. We use the training set to train a model $F_w(X)$. For new input x_{N+1}, the learned model will output y_{N+1}.

3.2.2 Loss Function

One common loss function for regression learning is **Mean Squared Error (MSE)**. It has the form

$$L(F_w(X)) = \frac{1}{n}\sum_{i=0}^{i=n}(F_w(x_i) - y_i)^2 \tag{14}$$

Since MSE is a convex function (look it up by yourself) and local optimum is the global optimum, we can use **Least Squares** to minimize MSE.
Let $Y \in \mathbf{R^n}$ be the Ground Truth vector, where n is the number of data; $X \in \mathbf{R^{nxd}}$, where d is the number of features including bias variable; $w \in R^d$ be weight vector. Then MSE can be rewritten as

$$\text{MSE}(w) = \frac{1}{n}\sum_{i=1}^{i=n}(Xw - Y)^2 \tag{15}$$
$$= \frac{1}{n}(Xw - Y)^T(Xw - Y) \tag{16}$$
$$= \frac{1}{n}(w^TX^TXw - w^TX^TY - Y^TXw + Y^TY) \tag{17}$$
$$\nabla\text{MSE}(w) = \frac{2}{n}(X^TXw - X^TY) \tag{18}$$

Setting the gradient to zero yields the following condition:

$$X^TXw = X^TY \tag{19}$$

Multiplying both sides with $(X^TX)^{-1}$, one obtains the following:

$$w = (X^TX)^{-1}XY \tag{20}$$