Regression and Gradient Descent

Version 1.0, August 2024

1 Regression

1.1 Linear regression

- We can augment the feature vector to incorporate offset:

$$X^T = [1, x_1, x_2, x_3, \ldots, x_n]$$

- The linear mapping as scalar product:

$$\hat{Y} = \sum_{i=1}^{n} w_i x_i = W^T X$$

1.1.1 Evaluating predictions

- The loss function can be expressed as absolute loss$|Y - \hat{Y}|$ or square loss $(Y - \hat{Y})^2$
- The aim is to minimize square loss over training points:

$$min(g(w_i)) = \sum_{i=1}^{n} (w_i x_i - y_i)^2 = (W^T X - Y)^T (W^T X - Y)$$

- The closed form solution:

$$W = (X^T X)^{-1} X^T Y$$

Note 1.

$$g(W) = (W^T X - Y)^T (W^T X - Y) = W X^T W^T X - W X^T Y - Y^T W^T X - Y^T Y$$

$$\nabla_W g(W) = 2W X^T X - 2W X^T Y = 0$$

$$\Rightarrow W = (X^T X)^{-1} X^T Y$$

However, for some cases, such as when the data contains auto-correlation, we consider more complex GLS and HLS regression. In this case, even if W satisfies the least squares, it is not necessarily the optimal solution.

1.1.2 Computational complexity

- Computational bottlenecks:
  - Matrix multiply of $X^T X$ is $O(nk^2)$ operations, where X is a $n \times k$ matrix
  - Matrix inverse of $X^T X$ is $O(k^3)$ operations
  - The storage requirement is $O(nk)$ floats

1.2 Linear classification

Idea: threshold by sign

$$\hat{Y} = \sum_{i=1}^{n} w_i x_i = W^T X \Rightarrow \hat{Y} = sign(W^T X)$$

Let's interpret this rule:

- $\hat{Y} = 1 : W^T X > 0$
- $\hat{Y} = -1 : W^T X < 0$
- decision boundary: $W^T X = 0$

So, we use logistic (or sigmoid) function: $P[Y = 1|X] = \sigma(W^T X)$.

If we give $\sigma(z) = \frac{1}{1+exp(-z)}$, then in the Logistic regression,

$$P[Y = 1|X] = \sigma(W^T X)$$

$$P[Y = 0|X] = -\sigma(W^T X)$$

W can be obtained by minimizing the log-likelihood or cross-entropy error. Log-likelihood of the whole training data D is

$$logP(D) = \sum_{i=1}^{n} y_i log\sigma(w^T x_i) + (1 - y_i)log[1 - \sigma(w^T x_i)]$$

It is convenient to work with its negation, which is called cross-entropy error function, and we can get the first derivation as

$$\nabla_w f(w) = \sum_{i=1}^{n} \sigma(w^T x_i) - y_i x_i$$

2 Gradient descent

2.1 Batch gradient descent

Algorithm 1: Batch Gradient descent Algorithm

Data: Optimization function: $f(w)$
Result: the best w_k s.t. $f(w_k)$ is the minimum

```
Randomly select a data in the domain: w_0 s.t. w_k = w_0 and learning rate: η_0 ;
while i ≤ 10000 or ∇_w f(w_i) < 10^{-6} do
    Computing the gradient ∇_w f(w)|_{w_i};
    Judge:
    if ∇_w f(w)|_{w_i} > 0 then
        f(w_{i+1}) < f(w_i) as w_{i+1} < w_i
    end
    if ∇_w f(w)|_{w_i} < 0 then
        f(w_{i+1}) < f(w_i) as w_{i+1} > w_i
    end
    Update model parameters: w_{i+1} = w_i − η_i * ∇_w f(w)|_{w_i} and η_{i+1} = η/(n√i)
end
```

2.2 Stochastic gradient descent

In batch gradient descent, the Loss of all samples needs to be calculated, while SGD only calculates the Loss of one sample and then performs gradient descent.

- In batch gradient descent we update $w_{i+1} = w_i - \eta_i * \nabla_w f(w)|_{w_i}$.
- While in stochastic Gradient descent we update $w_{i+1} = w_i - \eta_i * \nabla_w f_j(w)|_{w_i}$.

Here are some pros and cons of Stochastic Gradient descent:

- Pros:
  - Less computation
  - n times cheaper that gradient descent at each iteration
  - This faster pre-iteration cost might lead to faster overall convergence
- Cons:
  - Less stable convergence than gradient descent
  - In terms of iterations: slower convergence than batch gradient descent
  - More iterations

2.3 Mini - batch SGD

In the Mini - batch SGD, we update as $w_{i+1} = w_i - \eta_i * \nabla_w f_{B_i}(w)|_{w_i}$, where $B_i \subseteq i, \ldots, n$ sampled at random. The essence of this algorithm is to transform the previous random single into a random combination. We can think that when $B_j = 1$, it is Stochastic gradient descent. Also, here are some pros and cons:

- Pros:
  - More computation than SGD
  - Less computation that GD (might lead to faster overall convergence)
  - Can tune computation v.s. communication depending on batch size
- Cons:
  - In terms of iterations: slower convergence than gradient descent
  - In terms of iterations: Another parameter to tune(batch size)
  - Still might be too much communication