NLP Study Notes

Du Jinrui

2022

Contents

1 Shortest-Path Algorithms and Dynamic Programming 2

1.1 Graphs 2

1.2 Dynamic programming 2

1.2.1 DP coding problems 2

1.3 The Viterbi algorithm 3

2 Logistic Regression 4

2.1 The importance of establishing a baseline 4

2.2 Understanding LR 4

2.3 Deriving the cost function 5

2.4 Gradient descent 5

2.5 Implement LR and mini-batch GD 6

3 Generalization 7

3.1 When w goes to infinity 7

Chapter 1

Shortest-Path Algorithms and Dynamic Programming

1.1 Graphs

1.2 Dynamic programming

When designing a DP algorithm, there are two things to consider:

1. Deconstruct a big problem into smaller (recursive) sub-problems.
2. Store intermediate results.

1.2.1 DP coding problems

- Nth Fibonacci Number

- Longest Increasing Sub-sequence
- Coin Change

1.3 The Viterbi algorithm

Chapter 2

Logistic Regression

2.1 The importance of establishing a baseline

We draw a function that shows decreased marginal accuracy with increasing model complexity. From this graph, we observe an upper limit. This limit helps us making informed decisions like:

1. Is this project feasible? (the requirement is 75% accuracy but the upper limit is 72%.)

2. Is it cost-effective to add model complexity?

Furthermore, if we use a complex model upfront without setting a baseline but the accuracy is bad, then it's hard for us to tell whether there was a mistake when building the model or it's because the problem is too complex.

2.2 Understanding LR

graph of 1d data draft*

Why sigmoid?

2.3 Deriving the cost function

The likelihood function is defined as $L(\theta|D) = P(D;\theta)$, ; (parameterized by) is used to differentiate it from conditional probability, and the L is not a probability density function either, but a function of θ.

The likelihood function of logistic regression is

$$\prod_{i=1}^{n} \sigma(wx_i + b)^{y_i} (1 - \sigma(wx_i + b))^{1-y_i}.$$

(see derivation) Maximizing the likelihood is equal to minimizing the negative log-likelihood:

$$l(w, b) = -\sum_{i=1}^{n} y_i \ln \sigma(wx_i + b) + (1 - y_i) \ln(1 - \sigma(wx_i + b)).$$

And we get KL divergence, or binary cross-entropy, which is convex. (Why is it convex? And what is the difference between kl divergence and cross-entropy? draft*)

2.4 Gradient descent

The cost function can't be solved analytically, hence we use gradient descent. The derivative of the sigmoid function is:

$$\sigma(x)(1 - \sigma(x)).$$

Knowing this facilitates the calculation of the gradient:

$$\frac{\partial l(w, b)}{\partial w} = \sum_{i=1}^{n} (\sigma(wx_i + b) - y_i)x_i$$

$$\frac{\partial l(w, b)}{\partial b} = \sum_{i=1}^{n} \sigma(wx_i + b) - y_i.$$

2.5 Implement LR and mini-batch GD

Between GD and stochastic GD, mini-batch GD finds the balance between robustness and efficiency. Moreover, it works well with GPU, and it helps escaping the saddle point. draft*

Chapter 3

Generalization

3.1 When w goes to infinity

If the problem is linearly separable, MLE is the largest when w goes to infinity:

$$p\left(y_i = 1|x_i; w, b\right) = \frac{1}{1 + e^{-(wx_i + b)}} \approx 1$$
$$p(y_i = 0|x_i; w, b) = \frac{e^{-(wx_i + b)}}{1 + e^{-(wx_i + b)}} \approx 0.$$

$$MLE = argmax_{w,b} \prod_{i=1}^{n} p\left(y_i = 1|x_i; w, b\right)^{y_i} p\left(y_i = 0|x_i; w, b\right)^{1-y_i}.$$

A large w is consistent with our goal of maximizing the likelihood function.

Why does logistic regression overfit in high-dimensions?