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 From likelihood to cost function 5
2.4 Implement LR with mini-batch GD 5

3 Generalization **7**
3.1 When w goes to infinity 7
3.2 L1 and L2 regularization 8

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 From likelihood to cost function

The likelihood function is defined as $l(\theta|D) = f(D|\theta)$. f can be either a PMF or a PDF. $|$ is used instead of ; because we employ the Bayesian view (not frequenist) and see θ as a random variable. l is a function of θ and doesn't integrate to 1 (with respect to 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:

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

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 Implement LR with mini-batch GD

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:

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

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

Now we update the parameters:

$$w^{t+1} = w^t - \eta_t \sum_{i=1}^{n} (\sigma(wx_i + b) - y_i)x_i$$
$$b^{t+1} = b^t - \eta_t \sum_{i=1}^{n} \sigma(wx_i + b) - y_i.$$

Now we've got the updates using GD. The updates using mini-batch GD and stochastic GD become apparent. The former is:

$$w^{t+1} = w^t - \eta_t \sum_{x_i, y_i \in batch} (\sigma(wx_i + b) - y_i)x_i$$
$$b^{t+1} = b^t - \eta_t \sum_{x_i, y_i \in batch} \sigma(wx_i + b) - y_i.$$

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.

code draft*

Chapter 3

Generalization

3.1 When w goes to infinity

When the problem is linearly separable, as w goes to infinity:

$$\lim_{w\to\infty} p\left(y_i = 1|x_i; w, b\right) = \lim_{w\to\infty} \frac{1}{1+e^{-(wx_i+b)}} = 1 \; for \; wx_i + b > 0,$$

$$\lim_{w\to\infty} p(y_i = 0|x_i; w, b) = \lim_{w\to\infty} \frac{e^{-(wx_i+b)}}{1+e^{-(wx_i+b)}} = 0 \; for \; wx_i + b < 0.$$

At this time, MLE is the largest:

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

It is consistent with our goal of maximizing the likelihood function to aim for a large w. For a linearly separable problem, LR doesn't converge, and regularization gives bounded solution.

For a non-linearly separable problem, LR can converge (mathematically, why?). But when there are too many features, the non-separable becomes the separable, again, w goes to infinity, and uncertainty regions shrink to 0. At this point, limiting the

magnitude of w leads to better generalization and gives back uncertainty regions (how? And how does it relates to bias-variance trade-off? draft* We don't discuss feature selection here, why don't we just use feature selection?).

3.2 L1 and L2 regularization