

NLP Study Notes

Du Jinrui

2022

Contents

1	Shortest-Path Algorithms and Dynamic Programming	3
1.1	Graphs	3
1.2	Dynamic programming	3
1.2.1	DP coding problems	3
1.3	The Viterbi algorithm	4
2	Logistic Regression	5
2.1	The importance of establishing a baseline	5
2.2	Understanding LR	5
2.3	From likelihood to cost function	6
2.3.1	On KL-divergence and cross-entropy	6
2.3.2	Logistic loss	6
2.4	Implement LR with mini-batch GD	7
3	Generalization	8
3.1	When w goes to infinity	8
3.2	L1 and L2 regularization	9
3.3	K-fold CV	9
3.4	MLE, MAP and L1, L2	9
4	SVM	11

5	Naive Bayes, Decision Tree and Random Forest	12
5.1	Naive Bayes	12
6	Text Processing	13
6.1	Segmentation	13
6.1.1	Tools for Chinese words segmentation . . .	13
6.1.2	Max matching algorithm	13

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 frequentist) 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:

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

And we get KL divergence, or binary cross-entropy, which is convex. (Why is it convex?)

2.3.1 On KL-divergence and cross-entropy

2.3.2 Logistic loss

If the outcome space $y = \{-1, 1\}$ instead of $y = \{0, 1\}$, then

$$\begin{aligned} p(y_i = 1|f(x_i)) &= \sigma(f(x_i)) = \frac{1}{1 + e^{-f(x_i)}} \\ p(y_i = -1|f(x_i)) &= 1 - \sigma(f(x_i)) = \frac{1}{1 + e^{f(x_i)}}. \end{aligned}$$

In both cases

$$p(y_i|f(x_i)) = \frac{1}{1 + e^{-y_i f(x_i)}}.$$

The negative log-likelihood is

$$\sum_{i=1}^n \log(1 + e^{-y_i f(x_i)}).$$

Which is called the log loss/logistic loss and it's the same thing as the cross-entropy loss.

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:

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

Now we update the parameters:

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

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

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

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:

$$\begin{aligned}\lim_{w \rightarrow \infty} p(y_i = 1 | x_i; w, b) &= \lim_{w \rightarrow \infty} \frac{1}{1 + e^{-(wx_i + b)}} = 1 \text{ for } wx_i + b > 0, \\ \lim_{w \rightarrow \infty} p(y_i = 0 | x_i; w, b) &= \lim_{w \rightarrow \infty} \frac{e^{-(wx_i + b)}}{1 + e^{-(wx_i + b)}} = 0 \text{ for } wx_i + b < 0.\end{aligned}$$

At this time, MLE is the largest:

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

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

For a non-linearly separable problem, w 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 are all these happening? 1 2 Graphically, higher degree terms variables with smaller w doesn't disappear, but go 'out of range', e.g. $y = 6x_1 + 3x_2^2$ vs $y = 6x_1 + 0.1x_2^2$. draft*

We don't discuss feature selection here, why don't we just use feature selection? Is there an algorithm for separability testing?

3.2 L1 and L2 regularization

3d geometric moving representation of l1 and l2 and why l1 makes some parameters 0. draft*

There are some disadvantages of l1 regularization:

1. It's not differentiable everywhere, so gradient descent doesn't work, in this case we can use subgradient descent (I don't need to know the details).
2. When a group of collinear features exist, it randomly selects one feature, but we want the best feature. The lecturer says using elastic net can counter this problem but I don't know how. It's another topic. draft*

3.3 K-fold CV

When dataset is small, we can increase k . One extreme case is leave-one-out CV.

3.4 MLE, MAP and L1, L2

MLE:

$$p(D|\theta).$$

MAP:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \propto p(D|\theta)p(\theta).$$

MAP estimator:

$$\theta_{MAP} = \arg \max_{\theta} \text{prior} \cdot \text{likelihood}.$$

Assume prior is $p(\theta) \sim N(0, \sigma^2)$,

$$\begin{aligned} p(\theta) &= \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\theta^2}{2\sigma^2}\right) \\ &\propto \exp\left(-\frac{\theta^2}{2\sigma^2}\right), \end{aligned}$$

$$\begin{aligned} \arg \max_{\theta} \log(p(\theta)) &= \arg \max_{\theta} \log\left(\exp\left(-\frac{\theta^2}{2\sigma^2}\right)\right) \\ &= \arg \max_{\theta} -\frac{\theta^2}{2\sigma^2}, \end{aligned}$$

$$\theta_{MAP} = \arg \min_{\theta} -\log \text{likelihood} + \frac{1}{2\sigma^2}\theta^2.$$

This looks very familiar. MAP estimator with Gaussian prior equals adding a l2 regularization term to the cost function (and how does the λ coefficient relates to the variance? draft*).

Similarly when $p(\theta) \sim \text{Laplace}(0, b)$, the resulting cost function is added by l1 term.

Chapter 4

SVM

Chapter 5

Naive Bayes, Decision Tree and Random Forest

5.1 Naive Bayes

Chapter 6

Text Processing

The text analytic workflow:

raw data → word segmentation → cleaning (stopwords, lowercase, special character) → normalization (stemming, lemmazation) → feature extraction (tf-idf, word2vec) → modeling (classification, similarity measures)

6.1 Segmentation

6.1.1 Tools for Chinese words segmentation

- Jieba: <https://github.com/fxsjy/jieba>
- SnowNLP: <https://github.com/isnowfy/snownlp>
- LTP: <http://www.ltp-cloud.com/>
- HanNLP: <https://github.com/hankcs/HanLP/>

6.1.2 Max matching algorithm