Machine Learning Study Guide

WILLIAM WATSON

Johns Hopkins University billwatson@jhu.edu

1 Linear Algebra and Calculus

- 1.1 General Notation
- 1.2 Matrix Operations
- 1.3 Matrix Properties
- 1.4 Matrix Calculus

2 Convex Optimization

- 2.1 Convexity
- 2.2 Convex Optimization

2.2.1 Gradient Descent

Using $\alpha \in \mathbb{R}$ as the learning rate, we can update a set of parameters θ with respect to minimizing a function f as follows:

$$\theta := \theta - \alpha \nabla f(\theta) \tag{1}$$

Stochastic gradient descent (SGD) is updating the parameter based on each training example, and batch gradient descent is on a batch of training examples.

2.2.2 Newton's Algorithm

Newton's algorithm is a numerical method using information from the second derivative to find θ such that $f'(\theta) = 0$.

$$\theta := \theta - \frac{f'(\theta)}{f''(\theta)} \tag{2}$$

For multidimensional parameters:

$$\theta := \theta - \alpha H^{-1} \nabla_{\theta} f(\theta) \tag{3}$$

Where H is the hessian matrix of second partial derivatives.

$$H_{ij} = \frac{\partial^2 f(\theta)}{\partial \theta_i \partial \theta_j} \tag{4}$$

2.3 Lagrange Duality and KKT Conditions

3 Probability and Statistics

- 3.1 Basics
- 3.2 Conditional Probability
- 3.3 Random Variables
- 3.4 Jointly Distributed Random Variables
- 3.5 Parameter Estimation
- 3.6 Probability Bounds and Inequalities

4 Information Theory

Information Theory revolves around quantifying how much information is present in a signal. The basic intuition lies in the fact that learning an unlikely event has occured is more informative than learning that a likely event has occured. The basics are:

- 1. Likely events should have low information content, and in the extreme case, events that are guaranteed to happen should have no information content whatsoever.
- 2. Less likely events should have higher information content.
- 3. Independent events should have additive information.

We satisfy all three properties by defining self-information of an event x for a probability distribution P as:

$$I(x) = -\log P(x) \tag{5}$$

We can quantify the amount of uncertainty in a distribution using Shannon Entropy:

$$H(P) = \mathbb{E}_{\mathbf{x} \sim P}[I(x)] = -\mathbb{E}_{\mathbf{x} \sim P}[\log P(x)] \tag{6}$$

Which in the discrete setting is written as:

$$H(P) = -\sum_{x} P(x) \log P(x) \tag{7}$$

In other words, the Shannon entropy of a distribution is the expected amount of information in an event drawn from that distribution. It gives a lower bound on the number of bits needed on average to encode symbols drawn from a distribution P. If we have two separate probability distributions P(x) and Q(x) over the same random variable x, we can measure how different these two distributions are using the Kullback-Leibler (KL) divergence:

$$D_{KL}(P||Q) = \mathbb{E}_{\mathbf{x} \sim P} \left[\log \frac{P(x)}{Q(x)} \right]$$

$$= \mathbb{E}_{\mathbf{x} \sim P} \left[\log P(x) - \log Q(x) \right]$$

$$= \sum_{x} P(x) \frac{\log P(x)}{\log Q(x)}$$
(8)

In the case of discrete variables, it is the extra amount of information needed to send a message containing symbols drawn from probability distribution P, when we use a code that was designed to minimize the length of messages drawn from probability distribution Q. The KL divergence is always non-negative, and is 0 if and only if P and Q are the same. We can relate the KL divergence to cross-entropy.

$$H(P,Q) = H(P) + D_{KL}(P||Q)$$

$$= -\mathbb{E}_{\mathbf{x} \sim P} [\log Q(x)]$$

$$= -\sum_{x} P(x) \log Q(x)$$
(9)

Minimizing the cross-entropy with respect to Q is equivalent to minimizing the KL divergence, because Q does not participate in the omitted term (entropy is constant).

5 Machine Learning Basics

- 5.1 Notation
- 5.2 Types of Learning
- 5.3 Metrics
- 5.3.1 Classification
- 5.3.2 Regression
- 5.4 Bias and Variance

6 Linear Regression

Linear Regression seeks to approximate a real valued label y as a linear function of x:

$$h_{\theta}(x) = \theta_0 + \theta_1 \cdot x_1 + \dots + \theta_n \cdot x_n \tag{10}$$

The θ_i 's are the parameters, or weights. If we include the intercept term via $x_0 = 1$, we can write our model more compactly as:

$$h(x) = \sum_{i=0}^{n} \theta_i \cdot x_i = \theta^T x \tag{11}$$

Here n is the number of input variables, or features. In Linear Regression, we seek to make h(x) as close to y for a set of training examples. We define the cost function as:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)} \right)^{2}$$
 (12)

6.1 LMS Algorithm

We seek to find a set of θ such that we minimize $J(\theta)$ via a search algorithm that starts at some initial guess for our parameters and takes incremental steps to make $J(\theta)$ smaller until convergence. This is known as gradient descent:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{13}$$

Here, α is the learning rate. We can derive the partial derivative as:

$$\frac{\partial}{\partial \theta_{j}} J(\theta) = \frac{\partial}{\partial \theta_{j}} \frac{1}{2} (h(x) - y)^{2}$$

$$= 2 \cdot \frac{1}{2} (h(x) - y) \cdot \frac{\partial}{\partial \theta_{j}} (h(x) - y)$$

$$= (h(x) - y) \cdot \frac{\partial}{\partial \theta_{j}} \left(\sum_{i=0}^{n} \theta_{i} x_{i} - y \right)$$

$$= (h(x) - y) x_{j}$$
(14)

Hence, for a single example (stochastic gradient descent):

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h \left(x^{(i)} \right) \right) x_j^{(i)} \tag{15}$$

This is called the LMS update rule. For a batched version, we can evaluate the gradient on a set of examples (batch gradient descent), or the full set (gradient descent).

$$\theta_j := \theta_j + \alpha \sum_{i=1}^m \left(y^{(i)} - h\left(x^{(i)}\right) \right) x_j^{(i)}$$
 (16)

6.2 The Normal Equations

We can also directly minimize J without using an iterative algorithm. We define X as the matrix of all samples of size m by n. We let \vec{y} be a m dimensional vector of all target values. We can define our cost function J as:

$$J(\theta) = \frac{1}{2} (X\theta - \vec{y})^T (X\theta - \vec{y}) = \frac{1}{2} \sum_{i=1}^m \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$
(17)

We then take the derivative and find its roots.

$$\nabla_{\theta} J(\theta) = \nabla_{\theta} \frac{1}{2} (X\theta - \vec{y})^T (X\theta - \vec{y})$$

$$= \frac{1}{2} \nabla_{\theta} \left(\theta^T X^T X \theta - \theta^T X^T \vec{y} - \vec{y}^T X \theta + \vec{y}^T \vec{y} \right)$$

$$= \frac{1}{2} \nabla_{\theta} \left(\operatorname{tr} \theta^T X^T X \theta - 2 \operatorname{tr} \vec{y}^T X \theta \right)$$

$$= \frac{1}{2} \left(X^T X \theta + X^T X \theta - 2 X^T \vec{y} \right)$$

$$= X^T X \theta - X^T \vec{y}$$

$$(18)$$

To minimize J, we set its derivatives to zero, and obtain the normal equations:

$$X^T X \theta = X^T \vec{y} \tag{19}$$

Which solves θ for a value that minimizes $J(\theta)$ in closed form:

$$\theta = \left(X^T X\right)^{-1} X^T \vec{y} \tag{20}$$

6.3 Probabilistic Interpretation

Why does linear regression use the least-squares cost function? Assume that the target variables and inputs are related via:

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)} \tag{21}$$

Here, $\epsilon^{(i)}$ is an error term for noise. We assume each $\epsilon^{(i)}$ is independently and identically distributed according to a Gaussian distribution with mean zero and some variance σ^2 . Hence, $\epsilon^{(i)} \sim \mathcal{N}\left(0, \sigma^2\right)$, so the density for any sample $x^{(i)}$ with label $y^{(i)}$ is $y^{(i)}|x^{(i)}$; $\theta \sim \mathcal{N}\left(\theta^T x^{(i)}, \sigma^2\right)$. This implies:

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$
(22)

The probability of a dataset X is quantified by a likelihood function:

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y}|X; \theta) \tag{23}$$

Since we assume independence on each noise term (and samples), we can write the likelihood function as:

$$L(\theta) = \prod_{i=1}^{m} p\left(y^{(i)}|x^{(i)};\theta\right)$$

$$= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \theta^{T}x^{(i)}\right)^{2}}{2\sigma^{2}}\right)$$
(24)

To get the best choice of parameters θ , we perform maximum likelihood estimation such that $L(\theta)$ is maximized. Usually we take the negative log and minimize:

$$\ell(\theta) = -\log L(\theta)$$

$$= -\log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \theta^{T} x^{(i)}\right)^{2}}{2\sigma^{2}}\right)$$

$$= -\sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \theta^{T} x^{(i)}\right)^{2}}{2\sigma^{2}}\right)$$

$$= -m \log \frac{1}{\sqrt{2\pi}\sigma} + \frac{1}{\sigma^{2}} \cdot \frac{1}{2} \sum_{i=1}^{m} \left(y^{(i)} - \theta^{T} x^{(i)}\right)^{2}$$

$$(25)$$

Hence, maximizing $L(\theta)$ is the same as minimizing the negative log likelihood $\ell(\theta)$, which for linear regression is the least squares cost function:

$$\frac{1}{2} \sum_{i=1}^{m} \left(y^{(i)} - \theta^T x^{(i)} \right)^2 \tag{26}$$

Under the previous probabilistic assumptions on the data, least-squares regression corresponds to finding the maximum likelihood estimate of θ . This is thus one set of assumptions under which least-squares regression can be justified as performing maximum likelihood estimation. Note that θ is independent of σ^2 .

6.4 Locally Weighted Linear Regression

Locally Weighted Regression, also known as LWR, is a variant of linear regression that weights each training example in its cost function by $w^{(i)}(x)$, which is defined with parameter $\tau \in \mathbb{R}$ as:

$$w^{(i)}(x) = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$$
 (27)

Hence, in LWR, we do the following:

- 1. Fit θ to minimize $\sum_{i} w^{(i)} (y^{(i)} \theta^T x^{(i)})^2$
- 2. Output $\theta^T x$

This is a non-parametric algorithm, where non-parametric refers to the fact that the amount of information we need to represent the hypothesis h grows linearly with the size of the training set.

7 Logistic Regression

We can extend this learning to classification problems, where we have binary labels y that are either 0 or 1.

7.1 The Logistic Function

For logistic regression, our new hypothesis for estimating the class of a sample x is:

$$h(x) = g\left(\theta^T x\right) = \frac{1}{1 + e^{-\theta^T x}} \tag{28}$$

where g(z) is the logistic or sigmoid function:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{29}$$

The sigmoid function is bounded between 0 and 1, and tends towards 1 as $z \to \infty$. It tends towards 0 when $z \to -\infty$. A useful property of the sigmoid function is the form of its derivative:

$$g'(z) = \frac{d}{dz} \frac{1}{1 + e^{-z}}$$

$$= \frac{1}{(1 + e^{-z})^2} (e^{-z})$$

$$= \frac{1}{(1 + e^{-z})} \cdot \left(1 - \frac{1}{(1 + e^{-z})}\right)$$

$$= g(z)(1 - g(z))$$
(30)

7.2 Cost Function

To fit θ for a set of training examples, we assume that:

$$P(y = 1|x; \theta) = h(x)$$

$$P(y = 0|x; \theta) = 1 - h(x)$$
(31)

This can be written more compactly as:

$$p(y|x;\theta) = (h(x))^{y} (1 - h(x))^{1-y}$$
(32)

Assume m training examples generated independently, we define the likelihood function of the parameters as:

$$L(\theta) = p(\vec{y}|X;\theta)$$

$$= \prod_{i=1}^{m} p\left(y^{(i)}|x^{(i)};\theta\right)$$

$$= \prod_{i=1}^{m} \left(h\left(x^{(i)}\right)\right)^{y^{(i)}} \left(1 - h\left(x^{(i)}\right)\right)^{1 - y^{(i)}}$$
(33)

And taking the negative log likelihood to minimize:

$$\ell(\theta) = -\log L(\theta)$$

$$= -\sum_{i=1}^{m} y^{(i)} \log h\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h\left(x^{(i)}\right)\right)$$
(34)

This is known as the binary cross-entropy loss function.

7.3 Gradient Descent

Lets start by working with just one training example (x,y), and take derivatives to derive the stochastic gradient ascent rule:

$$\frac{\partial}{\partial \theta_{j}} \ell(\theta) = -\left(y \frac{1}{g(\theta^{T}x)} - (1 - y) \frac{1}{1 - g(\theta^{T}x)}\right) \frac{\partial}{\partial \theta_{j}} g(\theta^{T}x)$$

$$= -\left(y \frac{1}{g(\theta^{T}x)} - (1 - y) \frac{1}{1 - g(\theta^{T}x)}\right) g(\theta^{T}x) \left(1 - g(\theta^{T}x)\right) \frac{\partial}{\partial \theta_{j}} \theta^{T}x$$

$$= -\left(y \left(1 - g(\theta^{T}x)\right) - (1 - y)g(\theta^{T}x)\right) x_{j}$$

$$= -\left(y - h(x)\right) x_{j}$$
(35)

This therefore gives us the stochastic gradient ascent rule:

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h \left(x^{(i)} \right) \right) x_j^{(i)} \tag{36}$$

We must use gradient descent for logistic regression since there is no closed form solution for this problem.

8 Softmax Regression

A softmax regression, also called a multiclass logistic regression, is used to generalize logistic regression when there are more than 2 outcome classes.

8.1 Softmax Function

The softmax function creates a probability distribution over a set of k classes for a training example x, with θ_k denoting the set of parameters to be optimzed for the k-th class.

$$P(y = k|x; \theta) = \frac{\exp\left(\theta_k^T x\right)}{\sum_i \exp\left(\theta_i^T x\right)}$$
(37)

- 8.2 MLE and Cost Function
- 8.3 Gradient Descent
- 9 Generalized Linear Models
- 9.1 Exponentional Family
- 9.2 Assumptions of GLMs
- 9.3 Examples
- 9.3.1 Ordinary Least Squares
- 9.3.2 Logistic Regression
- 9.3.3 Softmax Regression
- 10 Perceptron
- 11 Support Vector Machines
- 12 Generative Learning: Gaussian Discriminant Analysis
- 12.1 Assumptions
- 12.2 Estimation
- 13 Generative Learning: Naive Bayes
- 13.1 Assumptions
- 13.2 Estimation
- 14 Tree-based Methods
- 15 K-Nearest Neighbors
- 16 K-Means Clustering
- 16.1 Hierarchical Clustering
- 16.2 Clustering Metrics
- 17 Expectation-Maximization
- 17.1 Mixture of Gaussians
- 17.2 Factor Analysis
- 18 Principal Component Analysis
- 19 Independent Component Analysis
- 20 Reinforcement Learning
- 20.1 Markov Decision Processes
- 20.2 Policy and Value Functions
- 20.2 Value Iteration Algorithm