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

2.2.2 Newton's Algorithm

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:

$$I(x) = -\log P(x) \tag{1}$$

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

$$H(P) = \mathbb{E}_{\mathrm{x}\sim P}[I(x)] = -\mathbb{E}_{\mathrm{x}\sim P}[\log P(x)] \tag{2}$$

Which in the discrete setting is written as:

$$H(P) = -\sum_x P(x) \log P(x) \tag{3}$$

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:

$$\begin{aligned} D_{\mathrm{KL}}(P \| Q) &= \mathbb{E}_{\mathbf{x} \sim P}\left[\log \frac{P(x)}{Q(x)}\right] \\ &= \mathbb{E}_{\mathbf{x} \sim P}\left[\log P(x) - \log Q(x)\right] \\ &= \sum_x P(x) \frac{\log P(x)}{\log Q(x)} \end{aligned} \tag{4}$$

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.

$$\begin{aligned} H(P, Q) &= H(P) + D_{\mathrm{KL}}(P \| Q) \\ &= -\mathbb{E}_{\mathbf{x} \sim P}\left[\log Q(x)\right] \\ &= -\sum_x P(x) \log Q(x) \end{aligned} \tag{5}$$

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:

$$h_\theta(x) = \theta_0 + \theta_1 \cdot x_1 + \cdots + \theta_n \cdot x_n \tag{6}$$

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:

$$h(x) = \sum_{i=0}^{n} \theta_i \cdot x_i = \theta^T x \tag{7}$$

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:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)} \right)^2 \tag{8}$$

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 know as gradient descent:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{9}$$

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

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h(x) - y\right)^2 \\ &= 2 \cdot \frac{1}{2} \left(h(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(h(x) - y\right) \\ &= \left(h(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\ &= \left(h(x) - y\right) x_j \end{aligned} \tag{10}$$

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

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h\left(x^{(i)}\right) \right) x_j^{(i)} \tag{11}$$

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).

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h\left(x^{(i)}\right) \right) x_j^{(i)} \tag{12}$$

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:

$$J(\theta) = \frac{1}{2} (X\theta - \vec{y})^T (X\theta - \vec{y}) = \frac{1}{2} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)} \right)^2 \tag{13}$$

We then take the derivative and find its roots.

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) \\
&= \frac{1}{2}\nabla_\theta\left(\theta^T X^T X\theta - \theta^T X^T\vec{y} - \vec{y}^T X\theta + \vec{y}^T\vec{y}\right) \\
&= \frac{1}{2}\nabla_\theta\left(\operatorname{tr}\theta^T X^T X\theta - 2\operatorname{tr}\vec{y}^T X\theta\right) \\
&= \frac{1}{2}\left(X^T X\theta + X^T X\theta - 2X^T\vec{y}\right) \\
&= X^T X\theta - X^T\vec{y}
\end{aligned} \tag{14}$$

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

$$X^T X\theta = X^T\vec{y} \tag{15}$$

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

$$\theta = \left(X^T X\right)^{-1} X^T\vec{y} \tag{16}$$

6.3 Probabilistic Interpretation

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

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)} \tag{17}$$

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:

$$p\left(y^{(i)}|x^{(i)};\theta\right) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right) \tag{18}$$

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

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y}|X;\theta) \tag{19}$$

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

$$\begin{aligned}
L(\theta) &= \prod_{i=1}^{m} p\left(y^{(i)}|x^{(i)};\theta\right) \\
&= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right)
\end{aligned} \tag{20}$$

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:

$$\begin{aligned}\ell(\theta) &= -\log L(\theta) \\ &= -\log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right) \\ &= -\sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right) \\ &= -m \log \frac{1}{\sqrt{2\pi}\sigma} + \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{m} \left(y^{(i)} - \theta^T x^{(i)}\right)^2\end{aligned} \tag{21}$$

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:

$$\frac{1}{2} \sum_{i=1}^{m} \left(y^{(i)} - \theta^T x^{(i)}\right)^2 \tag{22}$$

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:

$$w^{(i)}(x) = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right) \tag{23}$$

Hence, in LWR, we do the following:

1. Fit θ to minimize $\sum_i w^{(i)} \left(y^{(i)} - \theta^T x^{(i)}\right)^2$
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:

$$h(x) = g\left(\theta^T x\right) = \frac{1}{1 + e^{-\theta^T x}} \tag{24}$$

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

$$g(z) = \frac{1}{1 + e^{-z}} \tag{25}$$

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:

$$\begin{aligned} g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\ &= \frac{1}{\left(1+e^{-z}\right)^2}\left(e^{-z}\right) \\ &= \frac{1}{\left(1+e^{-z}\right)} \cdot \left(1 - \frac{1}{\left(1+e^{-z}\right)}\right) \\ &= g(z)(1-g(z)) \end{aligned} \tag{26}$$

7.2 Cost Function

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

$$\begin{aligned} P(y=1|x;\theta) = h(x) \\ P(y=0|x;\theta) = 1-h(x) \end{aligned} \tag{27}$$

This can be written more compactly as:

$$p(y|x;\theta) = (h(x))^y\,(1-h(x))^{1-y} \tag{28}$$

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

$$\begin{aligned} L(\theta) &= p\left(\vec{y}|X;\theta\right) \\ &= \prod_{i=1}^{m} p\left(y^{(i)}|x^{(i)};\theta\right) \\ &= \prod_{i=1}^{m}\left(h\left(x^{(i)}\right)\right)^{y^{(i)}}\left(1-h\left(x^{(i)}\right)\right)^{1-y^{(i)}} \end{aligned} \tag{29}$$

And taking the negative log likelihood to minimize:

$$\begin{aligned} \ell(\theta) &= -\log L(\theta) \\ &= -\sum_{i=1}^{m} y^{(i)}\log h\left(x^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-h\left(x^{(i)}\right)\right) \end{aligned} \tag{30}$$

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:

$$\begin{aligned} \frac{\partial}{\partial\theta_j}\ell(\theta) &= -\left(y\frac{1}{g\left(\theta^T x\right)} - (1-y)\frac{1}{1-g\left(\theta^T x\right)}\right)\frac{\partial}{\partial\theta_j} g\left(\theta^T x\right) \\ &= -\left(y\frac{1}{g\left(\theta^T x\right)} - (1-y)\frac{1}{1-g\left(\theta^T x\right)}\right) g\left(\theta^T x\right)\left(1-g\left(\theta^T x\right)\right)\frac{\partial}{\partial\theta_j}\theta^T x \\ &= -\left(y\left(1-g\left(\theta^T x\right)\right) - (1-y)g\left(\theta^T x\right)\right)x_j \\ &= -\left(y-h(x)\right)x_j \end{aligned} \tag{31}$$

This therefore gives us the stochastic gradient ascent rule:

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h\left(x^{(i)} \right) \right) x_j^{(i)} \tag{32}$$

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.