Machine Learning Notes

Sebastian Gaume

June 11, 2020

Contents

Chapter 1

Algorithms

1.1 Gradient Descent

Algorithm 1: The gradient descent algorithm

Input: The number of iterations to run, $nIters$, and the initial values for the parameter vector, $thetaInit$.
Output: The optimised parameter vector.
$n \leftarrow 0$;
$\theta \leftarrow thetaInit$;
while $n < nIters$ **do**
  $\theta \leftarrow \theta - \alpha \nabla J(\theta)$; // $J(\theta)$ is the cost function we are optimising against
  $n \leftarrow n + 1$;
end
return θ

Chapter 2

Linear Regression

Linear regression models are used in regression problems of an arbitrary number of features and with features which can be functions of other features. In these problems, $\boldsymbol{y}^{(i)} \in \mathbb{R}$ and can represent any quantity. Our goal is to predict, based on features $\boldsymbol{x}^{(i)}$, what output value a datum corresponds to. For this model, the output of the hypothesis function is the predicted value of $\boldsymbol{y}^{(i)}$ given $\boldsymbol{x}^{(i)}$, parameterised by $\boldsymbol{\theta}$.

2.1 Hypothesis and Cost Function

In a linear regression model, our hypothesis is of the form

$$h_\theta(\boldsymbol{x}) = \boldsymbol{\theta}^\intercal \boldsymbol{x}, \quad h_\theta(\boldsymbol{x}) \in \mathbb{R} \tag{2.1}$$

or alternatively,

$$\boldsymbol{h_\theta}(\boldsymbol{X}) = \boldsymbol{X\theta}, \quad \boldsymbol{h_\theta}(\boldsymbol{X}) \in \mathbb{R}^m \tag{2.2}$$

and we use a mean square difference cost function of the form

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} \left[(h_\theta(\boldsymbol{x}^{(i)}) - \boldsymbol{y}^{(i)})^2 \right] = \frac{1}{2m} (\boldsymbol{X\theta} - \boldsymbol{y})^\intercal (\boldsymbol{X\theta} - \boldsymbol{y}). \tag{2.3}$$

2.2 The Normal Equation

One way to minimise the cost function in linear regression is to do so analytically using the normal equation. This says that

$$\boldsymbol{\theta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y} \tag{2.4}$$

gives us the $\boldsymbol{\theta}$ that minimises the cost function $J(\boldsymbol{\theta})$. This technique is useful up until $n >= 10^4$, as for these relatively low numbers of features it is more efficient than gradient descent and doesn't require us to choose a learning rate α.

2.3 Gradient Descent

We can also minimise the cost function using gradient descent. To do this, we need the gradient of $J(\boldsymbol{\theta})$ which is

$$\boldsymbol{\nabla}_j J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left[(h_\theta(\boldsymbol{x}^{(i)}) - \boldsymbol{y}^{(i)}) \cdot \boldsymbol{x}_j^{(i)} \right] \tag{2.5}$$

$$\boldsymbol{\nabla} J(\boldsymbol{\theta}) = \frac{1}{m} \boldsymbol{X}^\intercal (\boldsymbol{X\theta} - \boldsymbol{y}) \tag{2.6}$$

and so we can just use this in our gradient descent algorithm and it will optimise $\boldsymbol{\theta}$. Note that all linear regression hypotheses are convex and so gradient descent will always converge with a correct learning rate and enough iterations to the global optimum.

2.4 Regularised Linear Regression

To avoid overfitting our training data by adding higher order terms to our hypothesis, we can use regularisation, which penalises larger weights for each term causing the algorithm to prefer lower valued weights and thus reducing

the problem of overfitting. To do this, we need to change our cost function to

$$J(\boldsymbol{\theta}) = \frac{1}{2m}(\boldsymbol{X\theta} - \boldsymbol{y})^\intercal(\boldsymbol{X\theta} - \boldsymbol{y}) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2. \tag{2.7}$$

The extra term penalises higher values of θ_j for all of $\boldsymbol{\theta}$ except θ_0 by convention. It also introduces a new hyper-parameter λ which controls how much the algorithm penalises the higher values. In general, we want $\lambda >= 1$ but not too large as if it's too large it will cause the model to underfit the training data.

2.4.1 The Normal Equation for Regularised Linear Regression

To minimise our new cost function for regularised linear regression, we have the same two options as before, but we need to modify them slightly. For the normal equation, this means we instead find that

$$\boldsymbol{\theta} = \left(\boldsymbol{X}^\intercal\boldsymbol{X} + \lambda \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}\right)^{-1} \boldsymbol{X}^\intercal\boldsymbol{y} \tag{2.8}$$

finds the $\boldsymbol{\theta}$ that minimises the cost function $J(\boldsymbol{\theta})$. Note that the matrix that is being multiplied by λ is the $\mathbb{R}^{n+1}$ identity matrix except the $I_{1,1}$ entry is 0 instead of 1.

2.4.2 Gradient Descent for Regularised Linear Regression

We also need to change the $\boldsymbol{\nabla}J(\boldsymbol{\theta})$ we use in our gradient descent algorithm if we choose to use regularised linear regression. The new gradient is given by

$$\boldsymbol{\nabla}_0 J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left[(h_\theta(\boldsymbol{x}^{(i)}) - \boldsymbol{y}^{(i)})\right] \tag{2.9}$$

$$\boldsymbol{\nabla}_j J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left[(h_\theta(\boldsymbol{x}^{(i)}) - \boldsymbol{y}^{(i)}) \cdot \boldsymbol{x}_j^{(i)}\right] + \frac{\lambda}{m}\boldsymbol{\theta}_j \qquad \text{for } j \neq 0 \tag{2.10}$$

$$\boldsymbol{\nabla}J(\boldsymbol{\theta}) = \frac{1}{m}\boldsymbol{X}^\intercal(\boldsymbol{X\theta} - \boldsymbol{y}) + \frac{\lambda}{m}\begin{bmatrix} 0 \\ \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \\ \vdots \\ \boldsymbol{\theta}_n \end{bmatrix} \tag{2.11}$$

and we simply use these in our gradient descent algorithm instead of the previous versions.

Chapter 3

Logistic Regression

Logisitic regression models are used in binary classification problems, altough they can be extended to multi-class classification problems. In these problems, $\boldsymbol{y}^{(i)} \in \{0, 1\}$, where 0 represents the negative case and 1 represents the positive case, though these can then represent any given classes. Our goal is to predict, based on features $\boldsymbol{x}^{(i)}$, which class a datum is part of. For this model, the output of the hypothesis function is $P(\boldsymbol{y}^{(i)} = 1 | \boldsymbol{x}^{(i)}, \boldsymbol{\theta})$.

3.1 Hypothesis and Cost Function

In a logisitic regression model, our hypotheis is of the form

$$h_\theta(\boldsymbol{x}) = g(\boldsymbol{\theta}^\intercal \boldsymbol{x}), \quad h_\theta(\boldsymbol{x}) \in \mathbb{R} \tag{3.1}$$

$$g(\boldsymbol{z}) = \frac{1}{1 + e^{-\boldsymbol{z}}} \tag{3.2}$$

or equivalently

$$\boldsymbol{h_\theta}(\boldsymbol{X}) = g(\boldsymbol{X\theta}), \quad \boldsymbol{h_\theta}(\boldsymbol{X}) \in \mathbb{R}^m \tag{3.3}$$

as this ensures our hypothesis is between 0 and 1 for all possible inputs. We then use a cost logarithmic cost function which rewards $h_\theta(\boldsymbol{x}^{(i)})$ for being closer to $\boldsymbol{y}^{(i)}$ than the other possible value of $\boldsymbol{y}^{(i)}$ as it can only hold the values of 0 and 1. This cost function is

$$J(\boldsymbol{\theta}) = -\frac{1}{m} \sum_{i=1}^{m} \left[\boldsymbol{y}^{(i)} \log \left(h_\theta(\boldsymbol{x}^{(i)}) \right) + \left(1 - \boldsymbol{y}^{(i)} \right) \log \left(1 - h_\theta(\boldsymbol{x}^{(i)}) \right) \right] \tag{3.4}$$

$$= \frac{1}{m} \left(-\boldsymbol{y}^\intercal \log \left(g(\boldsymbol{X\theta}) \right) - (\boldsymbol{1} - \boldsymbol{y})^\intercal \log \left(\boldsymbol{1} - g(\boldsymbol{X\theta}) \right) \right) \tag{3.5}$$

and it is, like that of linear regression, convex. This means we can be sure gradient descent will always converge to the global optimum when used to minimise the function.

3.2 Decision Boundaries

Because the output of the hypothesis function gives us $P(\boldsymbol{y}^{(i)} = 1 | \boldsymbol{x}^{(i)}, \boldsymbol{\theta})$, we should predict that $\boldsymbol{y}^{(i)} = 1 \iff h_\theta(\boldsymbol{x}^{(i)}) >= 0.5$. This means that if we want to visualise what our model will predict the class of a given datum is is to plot the decision boundary of the hypothesis function given the optimised $\boldsymbol{\theta}$. This decision boundary is given by

$$h_\theta(\boldsymbol{x}) = 0.5 \tag{3.6}$$

or equivalently

$$\boldsymbol{\theta}^\intercal \boldsymbol{x} = 0 \tag{3.7}$$

and it is the boundary that separates the regions where the model will predict $\boldsymbol{y}$ is 1 and where it will predict $\boldsymbol{y}$ is 0. For a small enough number of features this can be plotted and interpreted by a human to help give insight into what the model is doing.

3.3 Multi-Class Problems

We can also use logistic regression in problems where there are more than 2 classes. To do this, we use a 'one-vs-all' approach, whereby if we have n different classes, we train n different classifiers, one for each class. We then have n different hypotheses, $h_\theta^{(i)}(\boldsymbol{x})$ which correspond to the classes $i \in \{1, 2, 3, \cdots, n\}$. We train them such that $h_\theta^{(i)}(\boldsymbol{x})$

is a binary classifier using a logistic regression modifier with the positive case being that $\boldsymbol{x}$ is a member of class i and the negative case being that $\boldsymbol{x}$ is a member of any other class. As such, to find the class $\boldsymbol{x}$ is most likely to be a part of, we just compare the values of all the different $h_\theta^{(i)}(\boldsymbol{x})$, and see which is highest. If $h_\theta^{(j)}(\boldsymbol{x})$ outputs the highest value, we predict that $\boldsymbol{x}$ is a member of class j.

3.4 Gradient Descent

To fit our logistic regression model, we need to use an optimisation algorithm such as gradient descent. It is worth noting that their are other algorithms which do the same thing faster, however their implementations are best left to those who specialise in numerical methods, however they only require the cost function and its gradient to be written by the user, so we only need to know the same things as we do for gradient descent. For logistic regression, the gradient of the cost function turns out to be the same as that of linear regression's cost function, except the hypothesis function is switched out, ie

$$\boldsymbol{\nabla}_j J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left[(h_\theta(\boldsymbol{x}^{(i)}) - \boldsymbol{y}^{(i)})\cdot \boldsymbol{x}_j^{(i)}\right] \tag{3.8}$$

$$\boldsymbol{\nabla} J(\boldsymbol{\theta}) = \frac{1}{m}\boldsymbol{X}^\intercal(g(\boldsymbol{X}\boldsymbol{\theta}) - \boldsymbol{y}). \tag{3.9}$$

3.5 Regularised Logistic Regression

Like with linear regression, we can include higher order features in our model to get a better fit model, however this can cause our model to overfit the training data. To solve this, we can once again use regularisation to penalise higher values for the weights of our parameters, to discourage over-fitting. For logisitic regression, this makes the cost function

$$J(\boldsymbol{\theta}) = \frac{1}{m}\left(-\boldsymbol{y}^\intercal \log\left(g(\boldsymbol{X}\boldsymbol{\theta})\right) - (\boldsymbol{1}-\boldsymbol{y})^\intercal \log\left(\boldsymbol{1} - g(\boldsymbol{X}\boldsymbol{\theta})\right)\right) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2. \tag{3.10}$$

3.5.1 Gradient Descent for Regularised Logistic Regression

To use gradient descent, or other similar optimisation algorithms, for regularised logistic regression, we need to find the gradient of our new cost function, which turns our to be

$$\boldsymbol{\nabla}_0 J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left[(h_\theta(\boldsymbol{x}^{(i)}) - \boldsymbol{y}^{(i)})\right] \tag{3.11}$$

$$\boldsymbol{\nabla}_j J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left[(h_\theta(\boldsymbol{x}^{(i)}) - \boldsymbol{y}^{(i)})\cdot \boldsymbol{x}_j^{(i)}\right] + \frac{\lambda}{m}\boldsymbol{\theta}_j \qquad \text{for } j \neq 0 \tag{3.12}$$

$$\boldsymbol{\nabla} J(\boldsymbol{\theta}) = \frac{1}{m}\boldsymbol{X}^\intercal(g(\boldsymbol{X}\boldsymbol{\theta}) - \boldsymbol{y}) + \frac{\lambda}{m}\begin{bmatrix}0\\ \boldsymbol{\theta}_1\\ \boldsymbol{\theta}_2\\ \vdots\\ \boldsymbol{\theta}_n\end{bmatrix}. \tag{3.13}$$