CSE422: Artificial Intelligence
Linear Models & Perceptron

Rafiad Sadat Shahir

1 Machine Learning

1.1 Notations

We will use $x^{(i)}$ to denote the input variables or input features, and $y^{(i)}$ to denote the output or target feature. A pair $(x^{(i)}, y^{(i)})$ is a training example. The dataset that will be used for learning is list of n training examples $\{(x^{(i)}, y^{(i)});\ i = 1, \ldots, n\}$. We will also use $\mathcal{X}$ to denote the space of input values, and $\mathcal{Y}$ to denote the space of output values.

1.2 Supervised Learning

Given a training set, the goal in a supervised learning problem is to learn a function $h : \mathcal{X} \mapsto \mathcal{Y}$. The function h is called a **hypothesis**. Pictorially, the process is as follows.

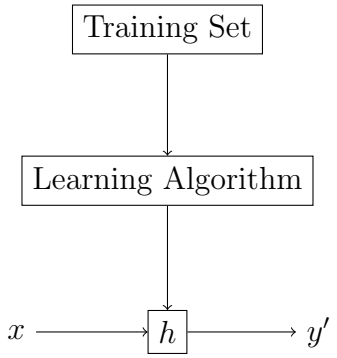

Figure 1: Hypothesis diagram.

When the target feature y is continuous, we call the learning problem a regression problem. When y can take on only a small number of discrete values, we call it a classification problem.

2 Linear Regression

2.1 Linear Hypothesis

To perform supervised learning, we first define the hypotheses h. Consider y as a linear function of x. Thus, h can be defined as

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n$$

Here, θs are the parameters (also known as weights) parameterizing the space of linear functions that map $\mathcal{X}$ to $\mathcal{Y}$. To simplify our notation, consider $x_0 = 1$. h can be expressed as

$$h(x) = \sum_{i=0}^{d} \theta_i x_i = \theta^\top x$$

2.2 Loss Function

The values of the parameters θ are selected so that $h(x)$ is close to y. To formalize this, a function is defined that measures, for each value of θs, how close $h(x^{(i)})$s are to the corresponding $y^{(i)}$s. The least squares loss function is defined as

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 .$$

2.3 Gradient Descent Algorithm

Gradient descent is the search algorithm that can be used to choose θ to minimize $J(\theta)$. The algorithm starts with some initial θ, and repeatedly performs the update.

$$\theta_j := \theta_j - \eta \frac{\partial}{\partial \theta_j} J(\theta)$$

Here, η is called the learning rate. This is a very natural algorithm that repeatedly takes a step in the direction of the steepest decrease of J.
The partial derivative term on the right-hand side can be expressed for one training example (x, y) as

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

For a single training example, this gives the update rule:

$$\theta_j := \theta_j + \eta \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

The rule is called the least mean squares update rule. Although gradient descent can be susceptible to local minima in general, the optimization problem we have posed here for linear regression has only one global and no other local optima. Thus, the gradient

descent always converges to the global minimum, assuming that the learning rate η is not too large.

2.3.1 Batch Gradient Descent

The method can be modified for a training set of more than one example by grouping coordinate updates into an update of the vector θ as in the following algorithm:

Algorithm 1 Batch gradient descent.

repeat

$$\theta := \theta + \eta \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x^{(i)}$$

until convergence

This method updates the parameters based on the entire training set at each step and is known as batch gradient descent.

2.3.2 Stochastic Gradient Descent

Consider the following algorithm:

Algorithm 2 Stochastic gradient descent.

repeat

  for $i = 1$ to n **do**

$$\theta := \theta + \eta \left(y^{(i)} - h_\theta^{(i)}\right) x^{(i)}$$

  end for

until convergence

The algorithm runs through the training set and updates the parameters using the gradient of the loss function for a single training example only. This algorithm is called stochastic gradient descent. Whereas batch gradient descent scans through the entire training set before taking a single step, stochastic gradient descent progresses with individual examples. Often, stochastic gradient descent gets θ close to the minimum much faster than batch gradient descent. However, it may never converge to the minimum and the parameters θ may continue to oscillate around the minimum of $J(\theta)$; but in practice most of the values near the minimum are reasonably good approximations to the true minimum. Thus, when the training set is large, stochastic gradient descent is often preferred over batch gradient descent.

2.4 Probabilistic Interpretation

The assumption for linear regression is that y and x are related by means of the following equation.

$$y^{(i)} = \theta^\top x^{(i)} + \epsilon^{(i)}$$

where $\epsilon^{(i)}$ is a noise term. Here, $\epsilon^{(i)}$ are independently and identically distributed (IID) according to a Gaussian distribution with mean zero and some variance σ^2. The assumption can be expressed as $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$, i.e. the density of $\epsilon^{(i)}$ is given by

$$f(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$$

This implies that

$$f(y^{(i)} \mid x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^\top x^{(i)})^2}{2\sigma^2}\right)$$

The notation "$p(y^{(i)} \mid x^{(i)}; \theta)$" denotes the probability density function of $y^{(i)}$ given $x^{(i)}$ parameterized by θ. Note that we should not condition on θ, since θ is not a random variable. The distribution of $y^{(i)}$ can be expressed as $(y^{(i)} \mid x^{(i)}; \theta) \sim \mathcal{N}(\theta^\top x^{(i)}, \sigma^2)$.
The function $f(y^{(i)} \mid x^{(i)}; \theta)$ is typically viewed as a function of y and x, for a fixed value of θ. If we want to view this as a function of θ, we call it the likelihood function:

$$L(\theta) = L(\theta; x, y) = p(y \mid x; \theta)$$

By the IID assumption, $L(\theta)$ can be written as

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^\top x^{(i)})^2}{2\sigma^2}\right). \end{aligned}$$

According to the maximum likelihood principle, θ should be chosen to make the data as high probability as possible, i.e., we should choose θ to maximize $L(\theta)$.
Instead of maximizing $L(\theta)$, we can also maximize any strictly increasing function of $L(\theta)$. In particular, the derivations will be a bit simpler if instead we maximize the log

likelihood $\ell(\theta)$:

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

Hence, maximizing $\ell(\theta)$ gives the same answer as minimizing

$$\frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \theta^\top x^{(i)}\right)^2,$$

which is the least-squares loss function $J(\theta)$.

2.5 Higher Dimensional Feature Mapping

The data may always not be linear. Thus, adding an additional feature x^2 to h, and fitting $y = \theta_0 + \theta_1 x + \theta_2 x^2$ may obtain a slightly better fit to the data. However, adding too many features can result in overfitting.

3 Logistic Regression

3.1 Logistic Hypothesis

If we use the linear regression algorithm to predict y given x where $y \in \{0, 1\}$, the method should perform poorly as $h_\theta(x)$ should not take values larger than 1 or smaller than 0. For such binary classification problems, the hypotheses $h_\theta(x)$ can be defined as

$$h_\theta(x) = g(\theta^\top x) = \frac{1}{1 + e^{-\theta^\top x}}$$

where

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

$g(z)$is called the sigmoid function, which is a logistic function. Here, $g(z) \to 1$ as $z \to \infty$, and $g(z) \to 0$ as $z \to -\infty$. Moreover, $g(z)$, and hence also $h(x)$, is always bounded

between 0 and 1. The derivative of the sigmoid function g' is

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

3.2 Probabilistic Interpretation

The assumption for logistic regression is the following.

$$p(y \mid x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y} \tag{1}$$

The distribution of $y^{(i)}$ can be expressed as $(y^{(i)} \mid x^{(i)}; \theta) \sim \mathcal{B}(h_\theta(x))$. By the IID assumption, the likelihood of the parameters can be expressed as

$$\begin{aligned} L(\theta) &= p(y \mid x; \theta) \\ &= \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{n} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

Then, the log likelihood can be expressed as

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

is called the log loss function. The derivative of the loss function is the following.

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \left(y \frac{1}{g(\theta^\top x)} - (1-y)\frac{1}{1 - g(\theta^\top x)}\right) \frac{\partial}{\partial \theta_j} g(\theta^\top x) \\ &= \left(y \frac{1}{g(\theta^\top x)} - (1-y)\frac{1}{1 - g(\theta^\top x)}\right) g(\theta^\top x)(1 - g(\theta^\top x)) \frac{\partial}{\partial \theta_j} \theta^\top x \\ &= \left(y(1 - g(\theta^\top x)) - (1-y) g(\theta^\top x)\right) x_j \\ &= (y - h_\theta(x)) x_j \end{aligned}$$

Thus, the update rule can be expressed as

$$\theta := \theta + \eta \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)}) \right) x^{(i)}$$

If we compare this with the linear regression update rule, we see that it looks identical; but this is not the same algorithm because $h_\theta(x^{(i)})$ is now defined as a non-linear function of $\theta^\top x^{(i)}$.

4 Perceptron

Consider changing the definition of g in logistic regression to be the threshold function:

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Let $h_\theta(x) = g(\theta^\top x)$ as before but using this modified definition of g, and if we use the update rule

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

then we have the perceptron learning algorithm. In the 1960s, the perceptron was argued to be a model for how individual neurons in the brain work.