

# Machine Learning

Sina Aleyaasin

November 3, 2017

## 1 Preliminaries

Computational approaches to *learning* from data involves some or all of following processes.

1. **Feature engineering** encoding information about a learning problem into machine processable format.
2. **Model selection** employing appropriate models to describe data
3. **Training** process existing empirical data to make hypotheses for predict future, unknown data points.

Techniques in machine learning coarsely divide into two categories

- Supervised learning
- Unsupervised learning

### 1.1 Feature engineering

Consider the problem of classifying objects from a set of examples  $\chi \in X$ . To represent the problem in the domain of a machine, we must represent the objects using numbers.

**Definition 1.1.** A feature vector is some vector representation of an object  $x$

$$x \equiv \phi(\chi) \in \mathbb{R}^d \quad (1)$$

The exercise of selecting  $\phi$  is referred to as *feature engineering*. We may express the set of examples  $X$  in this way as a matrix.

**Definition 1.2.** A **design matrix** is a matrix of  $N$  feature vectors representing  $N$  objects

$$X = \begin{bmatrix} x^{(1)T} \\ x^{(2)T} \\ \vdots \\ x^{(N)T} \end{bmatrix} \quad (2)$$

## 1.2 Supervised Learning

Consider the task of modelling the relationship between some dependent variable (*target*) on some explanatory independent variable (*feature*) given a data set.

**Definition 1.3.** A **training example** is an ordered *feature-target* pair

$$(x^{(i)}, y^{(i)}) \in \mathbb{X} \times \mathbb{Y} \quad (3)$$

**Definition 1.4.** A **training set** of size  $N$  is the set of training examples

$$\{(x^{(i)}, y^{(i)}) | i = 1, 2, \dots, N\} \quad (4)$$

The objective of supervised learning is to produce some *hypothesis*

$$h : \mathbb{X} \rightarrow \mathbb{Y} \quad (5)$$

to model the relationship by means of the corresponding training set.

**Definition 1.5.** **Regression** is supervised learning for a continuous, real valued  $\mathbb{Y} \equiv \mathbb{R}^{n+1}$ .

**Definition 1.6.** **Classification** is supervised learning for a finite, discrete  $\mathbb{Y}$ . The hypothesis of a classification problem is also known as a **classifier**.

## 1.3 Model selection

$\mathcal{H}$

## 1.4 Training

$h \in \mathcal{H}$

# 2 Linear Regression

Linear regression assumes a linear dependence of the target on the features.

$$h(x; \theta) \equiv h_{\theta}(x) = \theta^T x \quad (6)$$

where  $\theta, x \in \mathbb{R}^{n+1}$ .

*Remark.* Motivated by aesthetics, this notation adheres to the convention that every feature vector  $x$  has a constant first element  $x_0 = 1$  to account for the intercept term  $\theta_0$ .

The parameter  $\theta$  is determined by minimizing some *loss function* that aims to quantify the “error” of the classifier. We consider here the loss function giving corresponding to the **ordinary least squares** regression model

$$J(\theta) = \sum_{i=1}^N J^{(i)}(\theta) = \sum_{i=1}^N \frac{1}{2} \left( y^{(i)} - h_{\theta}(x^{(i)}) \right)^2 \quad (7)$$

## 2.1 Minimizing loss by normal equations

In some instances, as with the least mean squares regression model, the loss function may be minimized analytically to yield a closed form solution for  $\theta$

theorem and  
proof of lin-  
ear algebra  
+ calculate  
theta

## 2.2 Minimizing loss by gradient descent

For instances where no closed form solutions exist, one may perform a **gradient descent** until a desired threshold of convergence is reached.

$$\theta_{j+1} = \theta_j - \alpha \nabla_{\theta} J(\theta) \quad (8)$$

The parameter  $\alpha$  is known as the **learning rate**. This difference equation describes **batch gradient decent**. When the training set is large, one common modification can be made

$$\theta_{j+1} = \theta_j - \alpha \nabla_{\theta} J^{(i++)}(\theta_j) \quad (9)$$

where the  $++$  operator indicates iteration through the training set with each global iteration. This is referred to as a **stochastic gradient descent**.

## 2.3 Probabilistic interpretation

In order to appreciate the choice of loss function in the ordinary-least-squares model, consider the “error” produced by a linear model in a training example

$$\begin{aligned} \epsilon^{(i)} &= y^{(i)} - h(x^{(i)}) \\ &= y^{(i)} - \theta^T x^{(i)} \end{aligned} \quad (10)$$

Let us assume that the distribution of  $\epsilon^{(i)}$  in some training set is independently and identically distributed (IID) according to a Gaussian distribution

$$p(e^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{e^{(i)2}}{2\sigma^2}\right) \quad (11)$$

which by (10) implies

$$p(y^{(i)}|x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \quad (12)$$

This assumption may alternatively be expressed

$$e^{(i)} \sim \mathcal{N}(0, \sigma^2) \quad (13)$$

Equation (12) states the conditional probability of the random variable  $x^{(i)}$  given the random variable  $y^{(i)}$ , parameterised by  $\theta$ , takes the form of the given

Gaussian distribution. Minimizing the error amounts to maximizing this probability. In order to prescribe this probability to a training set, we use matrix notation. The **likelihood** function of a training set can then be expressed

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y}|X; \theta) \quad (14)$$

where  $\vec{y} = [y^{(1)}, y^{(2)}, \dots, y^{(N)}]^T$ . Using the independence assumption

$$\begin{aligned} L(\theta) &= \prod_{i=1}^N p(y^{(i)}|x^{(i)}; \theta) \\ &= \prod_{i=1}^N \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \end{aligned} \quad (15)$$

We note that maximizing the likelihood function is equivalent to maximizing any monotonically increasing function of the likelihood. We choose the logarithm function<sup>1</sup> to yield

$$\begin{aligned} \ell(\theta) &\equiv \log L(\theta) \\ &= \sum_{i=1}^N \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\ &\sim \sum_{i=1}^N \frac{1}{2} (y^{(i)} - \theta^T x^{(i)})^2 \end{aligned} \quad (16)$$

Not incidentally, we see from (6) that maximizing the likelihood (16) is equivalent to minimizing our loss function defined in (7).

### 2.3.1 Weighted linear regression

One common modification to the linear regression model is to account for *weights* in the loss function

$$J(\theta) = \frac{1}{2} \sum_{i=1}^N w^{(i)} (y^{(i)} - h_{\theta}(x^{(i)}))^2 \quad (17)$$

A standard choice for  $w^{(i)}$  is

$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^T (x^{(i)} - x)}{2\tau^2}\right) \quad (18)$$

where  $\tau$  is the **bandwidth parameter** and

<sup>1</sup>In computer arithmetic, addition is less expensive than multiplication. By using the logarithm function, products become sums and computation efficiency improves.

elaborate on  
 $x$

### 3 Logistic regression

Logistic regression is an approach to *binary classification*  $\mathbb{Y} = \{0, 1\}$  using a classifier naturally based on the **logistic function**.

$$h(x; \theta) = \frac{1}{1 + e^{-\theta^T x}} \quad (19)$$

Probabilistically, we interpret  $h$  such that

$$h_\theta(x) = P(y = 1|x; \theta) \quad (20)$$

and hence the conditional probability of  $y$  given  $x$  takes the form

$$P(y|x; \theta) = h_\theta(x)^y + (1 - h_\theta(x))^{1-y} \quad (21)$$

Extending (21) for a training set, using the assumption that the target errors are independently and identically distributed

$$\begin{aligned} L(\vec{y}) &= p(X; \theta) \\ &= \prod_{i=1}^N \left( h_\theta(x^{(i)}) \right)^{y^{(i)}} \left( 1 - h_\theta(x^{(i)}) \right)^{(1-y^{(i)})} \end{aligned} \quad (22)$$

calculation  
of log like-  
lihood plus  
derivative

### 4 Generalised Linear Model

The models discussed in the previous section may be expressed as subsets of a generalised **exponential family** of distributions which posit the target error distribution according to an arbitrary statistical model.

$$p(y; \eta) = b(y) \exp(\eta^T T(y)) - \alpha(\eta) \quad (23)$$

Define terms  
and express  
following  
distributions  
as GLM

- 4.1 Gaussian distribution
- 4.2 Bernoulli distribution
- 4.3 Multinomial distribution
- 4.4 Poisson distribution
- 4.5 Gamma distribution
- 4.6 Dirichlet distribution

### 5 Constructing a Generalised Linear Model

Lay out  
strategy for  
designing.  
Exemplify  
with OLS,  
logistic re-  
gression,  
softmax re-  
gression.

### 5.1 Ordinary least squares

### 5.2 Logistic regression

### 5.3 Softmax regression

## 6 Generative learning

A *generative model* aims to model data with a predictive capacity for both the feature and target variables of a data set. This is in contrast to *discriminative* models such as the regression models in the previous section which aim to model the target variable based on input features. Probabilistically, we may interpret generative learning as determining a probability distribution of  $x \in \mathbb{X}$  given  $y \in \mathbb{Y}$

$$p(x|y) \tag{24}$$

### 6.1 Gaussian discriminative analysis

### 6.2 Naive Bayes

The Naive Bayes assumption states that the conditional probability of individual elements in a feature set given the target are *independent*. That is to say, for  $x \in \mathbb{R}^{\{n+1\}}$ ,

$$p(x|y) \equiv p(x_1, x_2, \dots, x_{n+1}|y) = \prod_{i=1}^{n+1} p(x_i|y) \tag{25}$$

Alternatively, this may be more intuitively expressed

$$p(x_i|y) = p(x_i|y, x_j) \quad \forall i, j \in \{1, 2, \dots, n+1\} \tag{26}$$

### 6.3 Learning Theory

#### 6.3.1 Training error

The *training error* of a classifier describes its failure ratio in predicting output  $y^{(i)}$  in the training set

$$\xi_n(h) = \frac{1}{n} \sum_{i=1}^n \mathbb{I} \left[ h(x^{(i)}) \neq y^{(i)} \right] \tag{27}$$

where the operator  $\mathbb{I}$  maps a proposition to the binary value corresponding to its truth value.

$$\mathbb{I} [true] = 1 \quad \mathbb{I} [false] = 0$$

A small training error does not necessarily an indication of the quality of a classifier. A classifier may have a low training error but may not generalise well to data outside the training set. Such a classifier is the result of *overtraining*.

### 6.3.2 Generalization error

Generalization is the application of a classifier to data beyond the training set. The *generalisation error* measures the failure ratio of predicting  $y^{(i)}$  outside the training set. A good quality classifier is one that generalises well.

$$\xi(h) = \frac{1}{n'} \sum_{i=n}^{n+n'} \left[ h(x^{(i)}) \neq y^{(i)} \right] \quad (28)$$

## 7 Linear classification

A linear classifier demarcates two classes within a feature space by means of a hyperplane known as a *decision boundary*. Objects are classified according to which side of the hyperplane they lie.

**Definition 7.1.** For a feature space  $\mathbb{X} \in \mathbb{R}^{d+1}$ , a decision boundary is defined by the normal vector  $\theta \in \mathbb{R}^{d+1}$

$$P_\theta = \{z | z \cdot \theta = 0\} \quad (29)$$

The side to which a point  $x \in \mathbb{R}^d$  lies may be qualified by

$$\text{sign}(x \cdot \theta) \equiv x \cdot \theta / \|x\| \|\theta\| \quad (30)$$

Thus, we may express an arbitrary binary linear classifier  $h : \mathbb{R}^{d+1} \rightarrow \{+1, -1\}$  as

$$h(x, \theta) = \text{sign}(x \cdot \theta) \quad (31)$$

### 7.1 Separability

A training set  $T = \{(x_i, y_i) | i = 1, 2 \dots n\}$  is said to be *linearly separable* if there exists a decision boundary that correctly classifies every example  $(x_i, y_i)$

$$\exists \theta \forall (y, x) [y(x \cdot \theta) > 0] \quad (32)$$

### 7.2 Mistake-driven algorithms

Mistake driven algorithms are simple procedures used *learn* the parameter  $\theta$  of a linear model for some training set.

#### 7.2.1 Perceptron

The perceptron algorithm iterates through the training set, modifying  $\theta$  accordingly each time it miss-classifies a training example.

```
def perceptron(training_set, epochs, init_theta):
    theta = init_theta
    for n in range(epochs):
```

```

    for x, y in training_set:
        if y * theta.dot(x) <= 0:
            theta += x
    return theta

```

### 7.3 Loss functions

In the perceptron algorithm, every misclassification triggers a modification of equal significance to the parameter  $\theta$ . This disregards the distance of the training example from the decision boundary. A somewhat more sophisticated approach would be to incorporate some *loss function* that accounts for the margin of error in a misclassification and modifies  $\theta$  accordingly.

If training set is linearly separable, perceptron is guaranteed to converge to solution. Prove this!

#### 7.3.1 Hinge Loss

Hinge loss gives penalizes  $\theta$  on each iteration proportionally to the margin of error

$$\text{Hinge}(x, y; \theta) = \max\{0, 1 - y(x \cdot \theta)\} \quad (33)$$

```

def hingeloss(training_set, epochs, init_theta):
    theta = init_theta
    for n in range(epochs):
        for x, y in training_set:
            theta += max(0, 1 - y * theta.dot(x))
    return theta

```

#### 7.3.2 Passive-aggressive Algorithm

### 7.4 Support Vector Machines

## 8 Recommender problems

## 9 Non-linear classification

Large margin classifiers / Passive aggressive algorithm