Chapter 3

Fundamental Algorithms

In this chapter, I describe five algorithms which are not just the most known but also either very effective on their own or are used as building blocks for the most effective learning algorithms out there.

3.1 Linear Regression

Linear regression is a popular regression learning algorithm that learns a model which is a linear combination of features of the input example.

3.1.1 Problem Statement

$x_i \in \mathbb{R}^D$ ⇐
$y_i \in \mathbb{R}$

We have a collection of labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where N is the size of the collection, $\mathbf{x}_i$ is the D-dimensional feature vector of example $i = 1, \ldots, N$, y_i is a real-valued[1] target and every feature $x_i^{(j)}$, $j = 1, \ldots, D$, is also a real number.

We want to build a model $f_{\mathbf{w},b}(\mathbf{x})$ as a linear combination of features of example $\mathbf{x}$:

$$f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}\mathbf{x} + b, \tag{3.1}$$

↑ dot product or $f_{w,b}(x) = w^T x + b$
$= x^T w + b$

where $\mathbf{w}$ is a D-dimensional vector of parameters and b is a real number. The notation $f_{\mathbf{w},b}$ means that the model f is parametrized by two values: $\mathbf{w}$ and b.

We will use the model to predict the unknown y for a given $\mathbf{x}$ like this: $y \leftarrow f_{\mathbf{w},b}(\mathbf{x})$. Two models parametrized by two different pairs $(\mathbf{w}, b)$ will likely produce two different predictions

[1] To say that y_i is real-valued, we write $y_i \in \mathbb{R}$, where $\mathbb{R}$ denotes the set of all real numbers, an infinite set of numbers from minus infinity to plus infinity.

when applied to the same example. We want to find the optimal values $(\mathbf{w}^*, b^*)$. Obviously, the optimal values of parameters define the model that makes the most accurate predictions.

↳ define this.

~~You could have noticed that the form of our linear model in eq. 3.1 is very similar to the form of the SVM model. The only difference is the missing sign operator. The two models are indeed similar. However, the hyperplane in the SVM plays the role of the decision boundary: it's used to separate two groups of examples from one another. As such, it has to be as far from each group as possible.~~

On the other hand, the hyperplane in linear regression is chosen to be as close to all training examples as possible.

↳ what is Hyperplane?

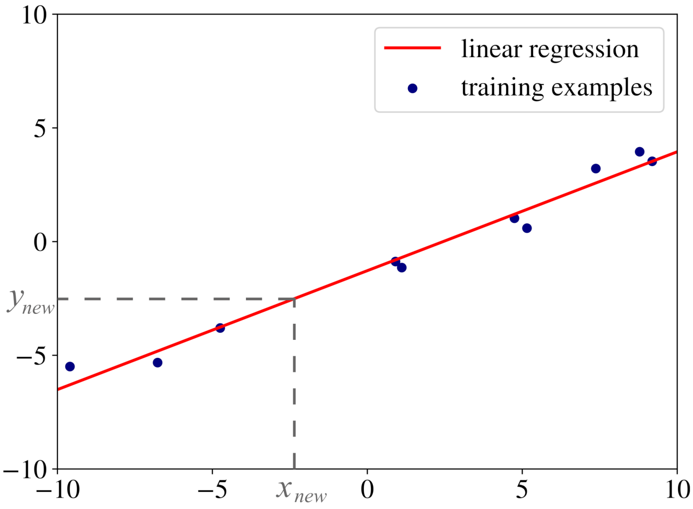

Figure 3.1: Linear Regression for one-dimensional examples.

You can see why this latter requirement is essential by looking at the illustration in Figure 3.1. It displays the regression line (in red) for one-dimensional examples (blue dots). We can use this line to predict the value of the target y_{new} for a new unlabeled input example x_{new}. If our examples are D-dimensional feature vectors (for $D > 1$), the only difference with the one-dimensional case is that the regression model is not a line but a plane (for two dimensions) or a hyperplane (for $D > 2$).

} Definition of hyperplane

Now you see why it's essential to have the requirement that the regression hyperplane lies as close to the training examples as possible: if the red line in Figure 3.1 was far from the blue dots, the prediction y_{new} would have fewer chances to be correct.

} Definition of accuracy

3.1.2 Solution

To get this latter requirement satisfied, the optimization procedure which we use to find the optimal values for $\mathbf{w}^*$ and b^* tries to minimize the following expression:

$$\frac{1}{N}\sum_{i=1\ldots N}(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2. \tag{3.2}$$

In mathematics, the expression we minimize or maximize is called an **objective function**, or, simply, an **objective**. The expression $(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2$ in the above objective is called the **loss function**. It's a measure of penalty for misclassification of example i. This particular choice of the loss function is called **squared error loss**. All model-based learning algorithms have a loss function and what we do to find the best model is we try to minimize the objective known as the **cost function**. In linear regression, the cost function is given by the average loss, also called the **empirical risk**. The average loss, or empirical risk, for a model, is the average of all penalties obtained by applying the model to the training data.

Why is the loss in linear regression a quadratic function? Why couldn't we get the absolute value of the difference between the true target y_i and the predicted value $f(\mathbf{x}_i)$ and use that as a penalty? We could. Moreover, we also could use a cube instead of a square.

Now you probably start realizing how many seemingly arbitrary decisions are made when we design a machine learning algorithm: we decided to use the linear combination of features to predict the target. However, we could use a square or some other polynomial to combine the values of features. We could also use some other loss function that makes sense: the absolute difference between $f(\mathbf{x}_i)$ and y_i makes sense, the cube of the difference too; the **binary loss** (1 when $f(\mathbf{x}_i)$ and y_i are different and 0 when they are the same) also makes sense, right?

If we made different decisions about the form of the model, the form of the loss function, and about the choice of the algorithm that minimizes the average loss to find the best values of parameters, we would end up inventing a different machine learning algorithm. Sounds easy, doesn't it? However, do not rush to invent a new learning algorithm. The fact that it's different doesn't mean that it will work better in practice.

People invent new learning algorithms for one of the two main reasons:

1. The new algorithm solves a specific practical problem better than the existing algorithms.
2. The new algorithm has better theoretical guarantees on the quality of the model it produces.

One practical justification of the choice of the linear form for the model is that it's simple. Why use a complex model when you can use a simple one? Another consideration is that linear models rarely overfit. **Overfitting** is the property of a model such that the model predicts very well labels of the examples used during training but frequently makes errors when applied to examples that weren't seen by the learning algorithm during training.

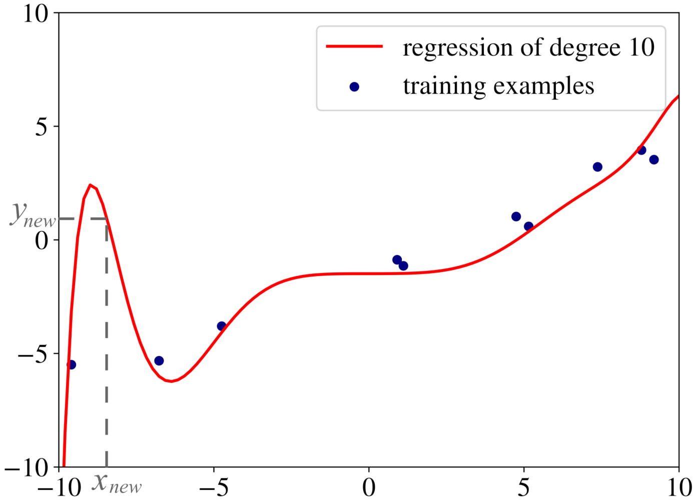

Figure 3.2: Overfitting.

An example of overfitting in regression is shown in Figure 3.2. The data used to build the red regression line is the same as in Figure 3.1. The difference is that this time, this is the polynomial regression with a polynomial of degree 10. The regression line predicts almost perfectly the targets almost all training examples, but will likely make significant errors on new data, as you can see in Figure 3.1 for x_{new}. We talk more about overfitting and how to avoid it in Chapter 5.

Now you know why linear regression can be useful: it doesn't overfit much. But what about the squared loss? Why did we decide that it should be squared? In 1805, the French mathematician Adrien-Marie Legendre, who first published the sum of squares method for gauging the quality of the model stated that squaring the error before summing is *convenient*. Why did he say that? The absolute value is not convenient, because it doesn't have a continuous derivative, which makes the function not smooth. Functions that are not smooth create unnecessary difficulties when employing linear algebra to find closed form solutions to optimization problems. Closed form solutions to finding an optimum of a function are simple algebraic expressions and are often preferable to using complex numerical optimization methods, such as **gradient descent** (used, among others, to train neural networks).

Intuitively, squared penalties are also advantageous because they exaggerate the difference between the true target and the predicted one according to the value of this difference. We might also use the powers 3 or 4, but their derivatives are more complicated to work with.

Finally, why do we care about the derivative of the average loss? If we can calculate the gradient of the function in eq. 3.2, we can then set this gradient to zero[2] and find the solution

[2] To find the minimum or the maximum of a function, we set the gradient to zero because the value of the gradient at extrema of a function is always zero. In 2D, the gradient at an extremum is a horizontal line.

Solution of Linear Regression

① Formulation:

Given: $\{(x_i, y_i)\} \to$ set of N points.

$$\left.\begin{array}{l} x_i \in \mathbb{R}^D \\ y_i \in \mathbb{R} \end{array}\right] \forall i = 1, 2, \text{---} N.$$

We want to find $\vec{w} \in \mathbb{R}^D$ & $b \in \mathbb{R}$ (hyperplane) s.t.
$f(x) = \vec{w} \cdot \vec{x} \cdot b$ is as close to points as possible

We can simplify the function by assuming $\vec{x} \in \mathbb{R}^{D+1}$ with $\vec{x}^{(D+1)} = 1$
& $\vec{w} \in \mathbb{R}^{D+1}$ with $\vec{w}$ as initial D locations & b at last pos

$$\Rightarrow \quad f(x) = \vec{w} \cdot \vec{x}$$

We want to minimize

$$E = \frac{1}{N} \sum_{i=1}^{N} \left(f(x_i) - y_i\right)^2$$

② Exact Solution - Normal form:

Define $\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^{n \times 1}$ - all inputs.

To keep things consistent for matrix product, define

$$X = \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{D+1} \Big\updownarrow n \qquad n \times (D+1)$$

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \\ b \end{bmatrix}_{(D+1)\times 1}$$

s.t. $XW - Y$ gives well defined difference.

Our target is

$$\text{minimize } \frac{1}{N} \|XW - Y\|^2$$

or minimize $\|XW - Y\|^2$ where Y & X are given & we need to find W.

$\Rightarrow$ Let $E = \|XW - Y\|^2$ minimize E.

$$\frac{\partial E}{\partial W} = 0 \quad \text{iff} \quad \frac{\partial}{\partial W} \|XW - Y\|^2 = 0$$

This can be done by using:

① Assuming $\frac{\partial E}{\partial W}$ is column vector.

$$\Rightarrow \frac{\partial x^2}{\partial x} ; \left(\frac{\partial x^2}{\partial x}\right)_i = \frac{\partial}{\partial x_i} x^2 = \frac{\partial}{\partial x_i} \sum_{j=1}^{N} x_j^2 = 2x_i \Rightarrow \frac{\partial x^2}{\partial x} = 2X$$

$$\& \quad \frac{\partial}{\partial x} Wx = W^T \quad \& \quad \frac{\partial A}{\partial b} = \frac{\partial c}{\partial b}\frac{\partial A}{\partial c}$$

$$\Rightarrow \frac{\partial E}{\partial W} = \frac{\partial}{\partial W} \|XW - Y\|^2 = \left(\frac{\partial}{\partial W}(XW - Y)\right) 2(XW - Y) = 0$$

$$\Rightarrow X^T X W - X^T Y = 0$$

$$\Rightarrow W = (X^T X)^{-1} X^T Y$$

$$((D+1) \times (D+1)) \times ((D+1) \times n) \times (n \times 1) = (D+1) \times 1$$

$\rightarrow$ Consistent

→ Why not $(X^T X)^{-1} X^T Y = X^{-1} (X^T)^{-1} X^T Y$
$= X^{-1} Y$?

Sol: left as exercise.

→ Why $(X^T X)^{-1}$ should exist?

Sol: left as exercise.

→ Any simplifications?

Trial: let $X = U \Sigma V^T \Rightarrow X^T X = V \Sigma^T U^T U \Sigma V^T$

$$\rightarrow X^T = V \Sigma^T U^T$$

$$= V \Sigma^T \Sigma V^T$$

$$\Rightarrow (X^T X)^{-1} = (V^T)^{-1} (\Sigma^T \Sigma)^{-1} V^{-1}$$

$$\rightarrow (X^T X)^{-1} X^T = (V^T)^T (\Sigma^T \Sigma)^{-1} V^{-1} V \Sigma^T U^T$$

$$= (V^T)^{-1} \left((\Sigma^T \Sigma)^{-1} \Sigma^T\right) U^T$$

$$= \left(V^{-1}\right)^T (\Sigma^{-1})^T U^T$$

③ Gradient Descent:

Keeping things same, we know

$$E = \frac{1}{N} \|XW - Y\|^2$$

we have input: W & output E.

$$\Rightarrow w^{(i+1)} = w^{(i)} - \alpha^{(i)} \left.\frac{\partial E}{\partial w}\right|_{w=w^{(i)}}$$

$$\Rightarrow \frac{\partial E}{\partial w} = \frac{\partial}{\partial w}\left(\frac{1}{N} \|XW - Y\|^2\right) = \left(\frac{2}{N}\right)(X^T)(XW - Y)$$

$$\Rightarrow \left.\frac{\partial E}{\partial w}\right|_{w^{(i)}} = \frac{2}{N}\left(X^T X w^{(i)} - X^T y\right)$$

$$\Rightarrow w^{(i+1)} = w^{(i)} - \frac{2\alpha}{N}\left(\underbrace{X^T X}_{\text{Precompute}} w^{(i)} - \underbrace{X^T y}_{\text{Precompute}}\right)$$

④ Stochastic gradient descent:

let we have $x^{(i)} \in \mathbb{R}^{D+1}$, $y^{(i)} \in \mathbb{R}$ in i^{th} iteration of gradient descent.

$$E = (w x^{(i)} - y^{(i)})^2$$

$\Rightarrow$ for $\frac{\partial E}{\partial w}$, we have

$$\frac{\partial E}{\partial w_j} = \frac{\partial}{\partial w_j}\left[\left(\sum_{k=1}^{D+1} w_k x_k^{(i)}\right) - y^{(i)}\right]^2$$

$$= 2\left(w x^{(i)} - y^{(i)}\right) x_j^{(i)}$$

$$\Rightarrow \frac{\partial E}{\partial w} = 2\left(w x^{(i)} - y^{(i)}\right) x^{(i)}$$

$$\Rightarrow w^{(i+1)} = w^{(i)} - 2\alpha^{(i)} x^{(i)}\left(w^{(i)} x^{(i)} - y^{(i)}\right)$$