Lecture 2 Introduction to Machine Learning

Kian Katanforoosh

October 2, 2018

Scribed by Nandita Bhaskhar (TA)

1 Outline

- Study a wide variety of ML algorithms, specifically: Linear regression, Logistic regression, Support Vector Machines, Neural Networks and Softmax regression.
- Understand ML techniques (both in training and inference) such as regularization
- Identify common patters that bind these classes of algorithms in terms of computation and conclude how accelerator designers can utilize these patterns for efficient hardware.

2 ML Algorithms

Algorithms discussed subsequently fall under the family of ***Supervised learning*** algorithms, i.e. dealing with datasets (X, Y) with labels or *ground truths*.

2.1 Linear Regression

Regression

To find a function $f : X \to Y \in \Re$ such that $\forall\ (x, y) \in (X, Y)$, $f(x) \simeq y$

Linear Regression

Find w, b such that $\forall (x, y) \in (X, Y)$, $f(x) = wx + b \simeq y$

Example

1. (Left) Here, $y \simeq wx + b$

X	Y	X1	X2	X3	Y
height (cm)	age (years)	height (cm)	voice features	footsize (cm)	age (years)
132	11.3	:	:		:
92	4.5	:	:		:
184	43.1	:	:		:

2. (Right) Here $w = (w_1, w_2, w_3)$ and $x = (x_1, x_2, x_3)^T$

Optimization problem set up

How do we find w, b? One way is to solve the OLS (ordinary least squares) problem in closed form to get the solution. But this gets very complicated with more features.

By setting this problem as an optimzation problem, we can find the best w, b that satisfy the constraints.

Cost function (L2):

$$J(w,b) = \frac{1}{2}\sum_{i=1}^{m}\left(f(x^{(i)}) - y^{(i)}\right)^2$$

Find w, b that minimize the cost function. Once we have the w, b, we can now predict the values that an unseen x_u will result in by $f(x_u) = y_u$.

Note: We want to minimize the computations during both training (to iterate quickly and find better models and hyperparameters) and inference (for real-time fast deployment).

2.2 Logistic Regression

Say that the problem you are trying to solve is a classification problem, i.e. predicting a binary output. For example, whether an email is spam or not, cat detection, etc. For these cases, it doesn't make sense to use linear regression.
We would like

$$f(x) \in [0, 1]$$

A good way to achieve this is to keep the linear regression framework and pass it to a sigmoid function σ which is defined as:

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

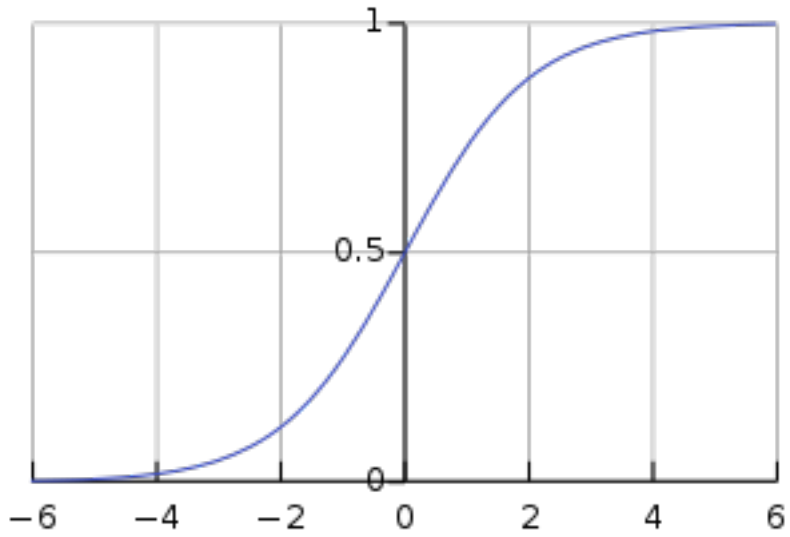

Figure 1: Sigmoid Function

Goal: Find w, b such that $\forall (x, y) \in (X \times Y), f(x) = \sigma(wx + b) \simeq y$.

The loss associated with this is called ***logistic loss*** given by

$$J(w, b) = -\sum_{i=1}^{m} \left(y^{(i)} log(\hat{y}^{(i)}) + (1 - y^{(i)}) log(1 - \hat{y}^{(i)}) \right)$$

2.3 Support vector machines

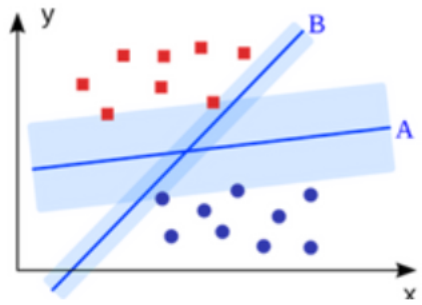

Figure 2: Line A is better than Line B

Goal: To find a hyperplane that separates the data with the 'best margin', i.e. largest distance to the nearest training data point of any of the classes.

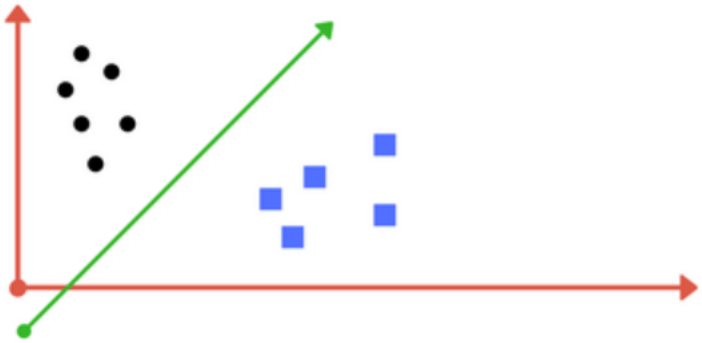

Figure 3: 2-class classification problem

For this example, say there are two classes, given by $\circ$ and $\diamond$. Say the two classes are denoted by labels $+1, -1$.

During inference, given a new x, predict

$$\hat{y} = g(wx + b) = +1, wx + b \geq 0; -1, wx + b \leq 0$$

From a logistic regression standpoint, when

1. $wx + b$ is large, $\sigma(wx + b) \sim 1 \Rightarrow$ high confidence that it is class 1.

2. $wx + b << 0$, $\sigma(wx + b) \sim 0 \Rightarrow$ high confidence that it is class -1

3. $wx + b \sim 0$, $\sigma(wx + b) \simeq 0.5 \Rightarrow$ low confidence on prediction.

We want to maximize the margin $\Delta = \frac{1}{||w||}$ such that all points are no closer than $1 (= |wx + b|)$ to the decision boundary $wx + b = 0$.

i.e., $\forall (x, y) \in (X \times Y \in \{-1, 1\})$ we want $y(wx + b) \geq 1$

Proof why $\Delta = \frac{1}{||w||}$:

Consider the support vectors $x1, x2$ from the classes -1, and +1 respectively.

$\Delta = \frac{1}{2}||x1 - x2||$ where $(x1 - x2)||w$

$||w.(x1 - x2)|| = 2 = ||w||||x1 - x2|| = ||w||(2\Delta)$

Hence, $\Delta = \frac{1}{||w||}$

Framing this as an **optimization** problem:

maximize $\frac{1}{||w||}$ such that $\forall i \in [1, m], y^{(i)}(wx^{(i)} + b) \geq 1$

This is a hard problem to optimize. The same solution can be achieved by rewriting this problem as follows:

minimize $\frac{1}{2}||w||^2$ such that $\forall i \in [1, m], y^{(i)}(wx^{(i)} + b) \geq 1$

This optimization is complex to solve but becomes easy when we use the Lagrange Dual trick to modify it, ending up with **minimizing** the new objective function

$$J(w, b) = \sum_{i=1}^{m} max\left(0, 1 - y^{(i)}(wx^{(i)} + b)\right) + R(w)$$

2.4 Kernel trick

When data is not linearly separable, project it into higher dimensions and then find a hyperplane that will separate it linearly.

Example: $x = (x1, x2)$ and $\phi(x) = (x1, x2, x1^2 + x2^2)$
But this can get very complex computationally since we are projecting to higher dimensions intentionally. This is where the Kernel trick comes into play.

Trick

During inference, given new x, compute $w\phi(x) + b$ where we can write $w = \sum_{i=1}^{m} \alpha_i y^{(i)} \phi(x^{(i)})$. Thus the quantity to be computed is

$$\sum_{i=1}^{m} \alpha_i y^{(i)} \phi(x^{(i)}) \phi(x) + b$$

This is represented as

$$\sum_{i=1}^{m} \alpha_i y^{(i)} K(x^{(i)}, x) + b$$

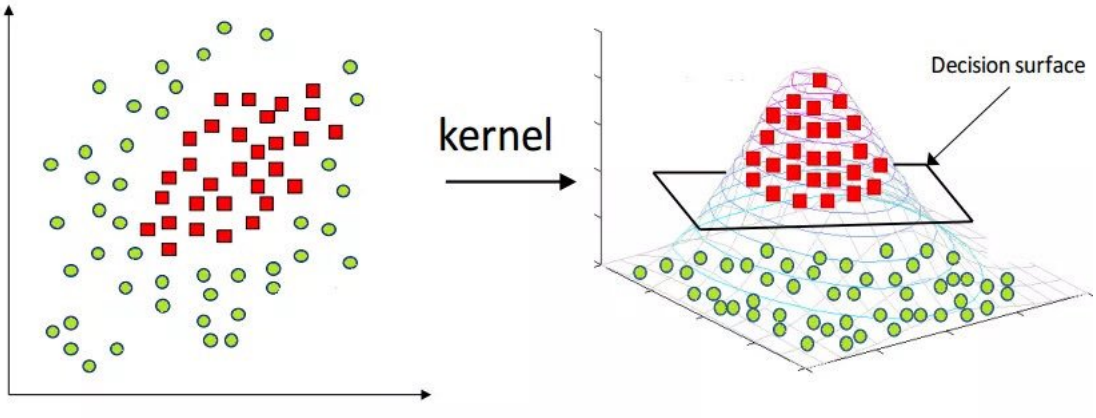

Figure 4: Linearly separable in the 3rd dimension

It turns out that we don't need to know ϕ, we only need to know K.

The Kernel can be computed fast since it involves only norm computations instead of expensive multiplications. (Refer CS229 lecture notes for details).

2.5 Fully Connected networks

A single neuron is essentially an entity that performs a non-linear activation on a linear summation of its inputs. The logistic regression from Section 2.2 can be represented as a single neuron. $\hat{y} = \sigma(\sum_{k=1}^{p} w^k a^k + b)$

A neural network is just an extension of this concept of a neuron.

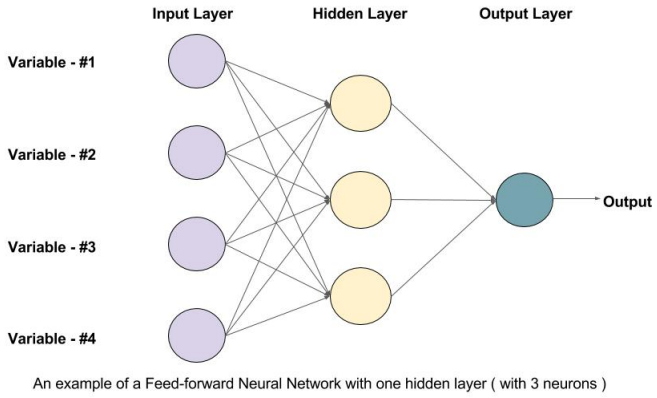

Figure 5: 2 layer fully connected NN

In the above toy NN, the number of parameters are $(3n + 3) + (3 + 1)$.

There are two aspects of NN: forward propagation and back propagation. The former is used alone during inference (prediction) while both are used during training. Back-prop is computationally very expensive since it involves computation of gradients.

3 Patterns common to these ML Algorithms

1. All are framed as optimization problems
2. The all have a loss function we will try to minimize

We can solve this optimization problem using an iterative algorithm such as GD (gradient descent).

3.1 Exceptions

Not all ML algorithms need to be expressed as Optimization problems. For example, K-NN (K nearest neighbours) algorithm is not framed as an optimization solution.

3.2 Computations

The main computations during inference and training are arithmetic operations (vector multiplication, addition, etc) and activation function (ReLu, sigmoid, etc) calculations. Training requires additional computations to calculate the gradients during the gradient descent step of the iterative solution.