Artificial Intelligence and Machine Learning

Abhijit Amrendra Kumar

August 2023

Chapter 1

Introduction

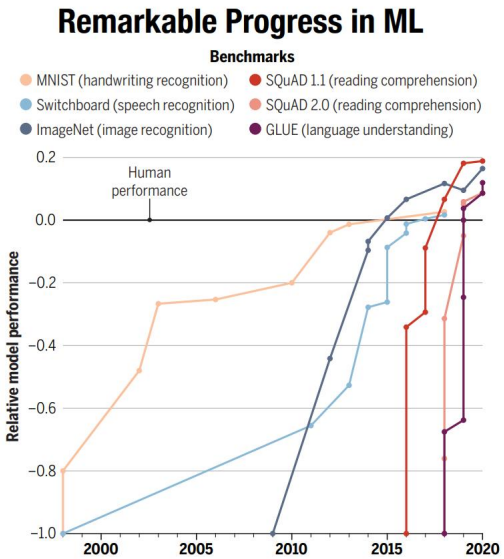
1.1 What is Machine Learning ?

Machine learning is the ability of machines to **learn** from data or past experiences, which come from various sources such as sensors, domain knowledge, experimental runs, etc. Mathematically, it involves making a theoretical function/model $f : X \to Y$ which predicts outcomes from given inputs, and optimizing it and training it to increase it's prediction accuracy.

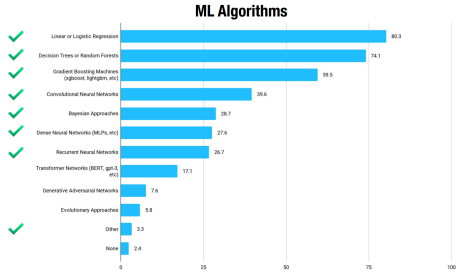
1.2 Why Machine Learning ?

We need machine learning in the following cases

- When we need to perform tasks that are easy for humans, but complex for computer systems to emulate (eg. distinguishing between a muffin and a Chihuahua)
- When we need to perform tasks which are beyond human capabilities, like the analysis of large and complex datasets



Here is a usage histogram for various ML algorithms, which indicates that linear regression and decision trees are very likely to be useful in a machine learning problem.



1.3 ML Pipeline

- **Data**: Collect data for your problem
  - Is the data labelled or unlabelled ?
- **Representation**: Choose features that represent your data
  - Is the data representation raw, expert-derived or learned ?
- **Modeling**: Choose a model for a task
  - Should the model be linear/non-linear ? Also, what will be the computational overheads ?
- **Training/Learning**: Model will (likely) be parameterized, and these parameters will be learned by training over the data. So, choose a loss function to optimize.
  - Which loss function to choose ? And how to best optimise this loss function ?
- **Prediction/Inference**: Given a model, assign labels to unseen test instances, then choose an evaluation metric.
  - Is the evaluation manual or automatic ?

1.4 Resources

Here are some links to find various datasets

- https://archive.ics.uci.edu/
- https://huggingface.co/datasets
- http://www.openslr.org/

Chapter 2

Linear Regression

2.1 Notation

In general we format variables as: x for scalars, $\mathbf{x}$ for vectors, and X for matrices.

- Input / Feature space / Attributes Space : $\mathcal{X} = \mathbb{R}^d$ for some $d \in \mathbb{N}$
- Output / *Label space* / Response space : $\mathcal{Y} = \mathbb{R}$
- Dataset : $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}\ \forall\ i \in \{1 \dots n\}$

We consider a target function $f : \mathcal{X} \to \mathcal{Y}$ on the training dataset $\mathcal{D}$, i.e. $\mathbf{x} \overset{f}{\mapsto} y\ \forall\ (\mathbf{x}, y) \in \mathcal{D}$.

We focus on the task of finding a *hypothesis function* $h : \mathcal{X} \to \mathcal{Y}$ that ideally closely approximates f. We call the family of the hypothesis functions as $\mathcal{H}$, the *Hypothesis Class*. It follows that $h \in \mathcal{H}$ This now brings the following questions:

1. What are the possibilities for the predictor function h ? [**Hypothesis Class**]
2. How do you quantify the performance of the predictor? [**Loss/Error Function**]
3. How do we find the best predictor? [**Optimization**]

2.2 What are the possible predictor functions?

Let us first play with a simpler case with a one-dimensional feature space. We may consider the problem as a line fitting problem, taking our hypothesis class to be all linear functions.

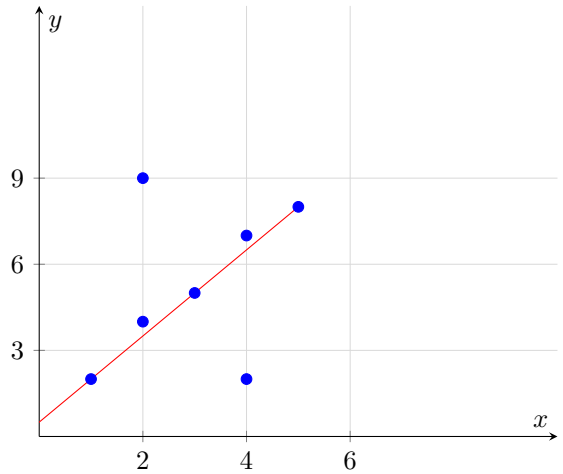


Figure 2.1: Fitting a line to the data

Let us parametrize the line with w_0, w_1 (intercept and slope). We can vectorize our parameters as $W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$. Then we may write our hypothesis function $h_{\mathbf{w}}$ parametrised by $\mathbf{w}$ as.

$$h_{\mathbf{w}}(x) = w_0 + w_1 x$$

Let us also vectorize our input as $\mathbf{x} = \begin{bmatrix} 1 \\ x \end{bmatrix}$, so that

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$

We can now extend this to the multidimensional case. We consider our function to return a linear combination of our d dimensional features.

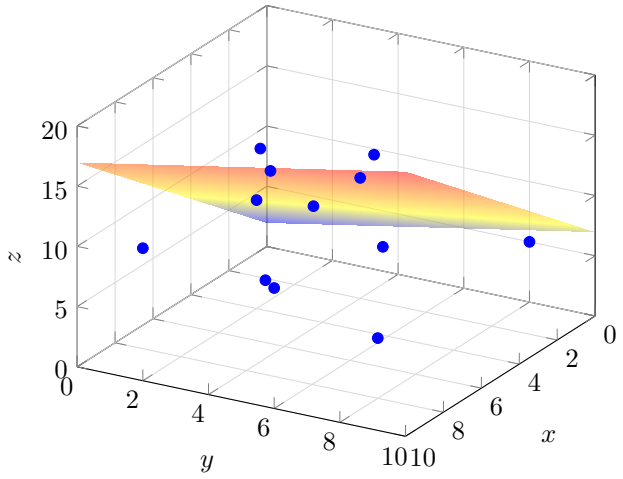


Figure 2.2: For higher dimensional feature spaces, linear regression is akin to *Hyperplane Fitting*

We now have our parameter $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \in \mathbb{R}^{d+1}$ and input vector $\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$ to get a identical expression -

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$

Our hypothesis class is, then -

$$\mathcal{H} = \{h_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^{d+1}\}$$

This is essentially the *linear* in linear regression. However, note that it does not mean we are restricted to linear functions of the features, we may transform the feature space to another space to regress.

2.3 How to quantify the performance of a predictor ?

We define a function that operates on the predictor function and dataset to quantify the "mismatch" between the two. Higher the loss function, lesser is the given predictor suitable for the dataset. Given that our function is parametrized, we may also define the loss function in terms of the parameter.

$$\text{Loss Function: } \mathcal{L}(h, \mathcal{D}) \text{ or } \mathcal{L}(\mathbf{w}, \mathcal{D})$$

Let us consider a singleton dataset $\mathcal{D}_{test} = \{(\mathbf{x}, y)\}$ where $\mathbf{x} = [1 \ x_1 \ \dots \ x_d]^\top$. We define a loss for this dataset as

$$\mathcal{L}(\mathbf{w}, \mathcal{D}_{test}) = (y - h_{\mathbf{w}}(\mathbf{x}))^2 = |y - \hat{y}|^2$$

We define $h_{\mathbf{w}}(x) = \hat{y}$, and $|y - \hat{y}|$ is called a *residual*.

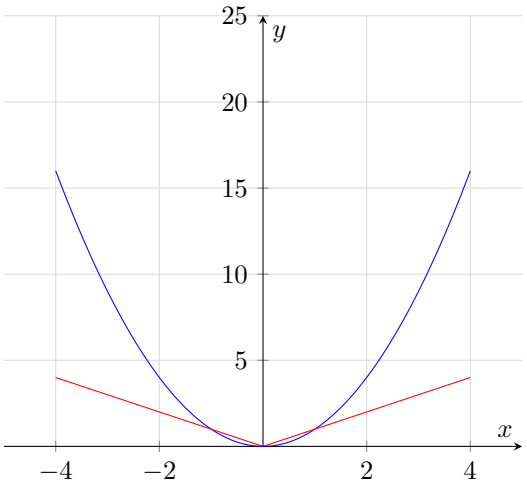
Now for a dataset containing n datapoints,

$$\textbf{Least Squares Loss}: \ \mathcal{L}(\mathbf{w}, \mathcal{D}_{train}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - h_{\mathbf{w}}(\mathbf{x}_i))^2 = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|^2$$

Note that the least squares loss is the mean of the squares of the residuals. We can also define a loss equal to the mean residual value.

$$\text{Mean Absolute Error Loss: } \mathcal{L}(\mathbf{w}, \mathcal{D}_{train}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

The Mean Absolute Error does not penalise high deviations as much as the Squares Loss, thus it may be more appropriate to use when the dataset contains many outliers.



Here we are interested in the least squares loss. We can vectorize the loss function as follows

$$\boxed{\mathcal{L}(\mathbf{w}, \mathcal{D}_{\text{train}}) = \frac{\mathbf{1}}{\mathbf{n}}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|_{\mathbf{2}}^{\mathbf{2}}}$$

$$\text{, where } X = \begin{bmatrix} \longleftarrow \mathbf{x}_1{}^\top \longrightarrow \\ \longleftarrow \mathbf{x}_2{}^\top \longrightarrow \\ \vdots \\ \longleftarrow \mathbf{x}_n{}^\top \longrightarrow \end{bmatrix}_{n\times(d+1)} \quad , \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n\times 1}$$

2.4 How do we find the best predictor?

We use the loss function to find our optimum hypothesis h^*, where

$$h^* = \arg\min_{h\in\mathcal{H}} \mathcal{L}(h, \mathcal{D}_{train})$$

[Note that the optimisation is performed only over the training dataset]
We may also define the objective in terms of the parameter $\mathbf{w}$ corresponding to h.

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathcal{D}_{train})\mathbf{w}_{LS} \qquad = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} (y_i - \mathbf{w}^\top x_i)^2 \tag{2.1}$$

We set out to find a closed form solution for $\mathbf{w}_{LS}$ for d dimensional data:
To find the optimum, we set the derivative[1] to zero. We may drop the $\frac{1}{n}$ term.

$$\nabla_{\mathbf{w}}\mathcal{L} = \frac{\partial \mathcal{L}(\mathbf{w}, \mathcal{D}_{train})}{\partial \mathbf{w}} = \left[\frac{\partial \mathcal{L}(\mathbf{w}, \mathcal{D}_{train})}{\partial w_i}\right]_{i=1}^{n} = \left[-2(y_i - \mathbf{w}^\top x_i)x_i\right]_{i=1}^{n} = 0$$
$$\implies 2(-X^\top \mathbf{y} + X^\top X\mathbf{w}) = 0$$
$$\implies \mathbf{w}_{LS} = (X^\top X)^{-1}X^\top Y$$

We can also derive this result with vector-derivative identities[1],

$$\begin{aligned}
&\nabla_{\mathbf{w}} \mathcal{L} = 0 \\
\implies &\frac{\partial}{\partial \mathbf{w}} \|\mathbf{y} - X\mathbf{w}\|_2^2 = \frac{\partial}{\partial \mathbf{w}} \left(\mathbf{y}^\top \mathbf{y} - 2\mathbf{w}^\top X^\top \mathbf{y} + \mathbf{w}^\top X^\top X \mathbf{w}\right) = 0 \\
\implies &-2X^\top \mathbf{y} + 2X^\top X \mathbf{w} = 0 \\
\implies &X^\top X \mathbf{w} = X^\top \mathbf{y} \\
&\boxed{\mathbf{w} = (X^\top X)^{-1}(X^\top \mathbf{y})}
\end{aligned}$$

Note that $X^\top X$ need not be invertible.

2.5 Homework

For 1D data, $h_{\mathbf{w}}(x) = w_0 + w_1 x$, prove that –

$$\mathbf{w}_1{}^* = \frac{\Sigma(x_i - \overline{x})(y_i - \overline{y})}{\Sigma(x_i - \overline{x})^2}$$

where $\overline{x} = \frac{\Sigma x_i}{N}$ and $\overline{y} = \frac{\Sigma y_i}{N}$.

[1]There are two conventions for the derivative of a scalar by a vector, i.e., whether the result is a row or column. Here we will stick with the latter.