CS5014: Machine Learning, Revision Notes

Gergely Flamich

May 11, 2018

1 Overview

In ML, we usually have some data in the following general setup:

- **Features**: literally any sort of observations form the real world, collected conveniently into a matrix:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{bmatrix}$$

  where each column $\mathbf{x}^{(i)}$ contains m observations of the i-th feature

- **Labels**: In supervised learning we also have another set of data, called the ground truth of the object we are trying to predict from the features. These are conveniently colleted into a vector $\mathbf{y}$ with m elements, each y_i corresponding to a set of observations, row $\mathbf{x}_i$ in X.

The general assumption for supervised learning is that each feature is associated with a random variable X_i over some distributions and the ground truth is associated with a random variable Y over some distribution. It is usually assumed that we know the distributions of X_i and Y is hidden. fianlly, we assume that there is some function f such that

$$Y = f(\mathbf{X}, \theta) + \epsilon$$

for some model paramters θ and some small error term ϵ. Then, the way we will perform prediction is by using the distribution

$$\hat{Y} = f(\mathbf{X}, \theta).$$

f is usually referred to as the **model function**.

1.1 Machine Learning Pipeline

1. Obtain data
2. Clean data
3. Visualise & Analyse data
4. Prepare data: augumentation, feature selection etc.
5. Perform the learning task
6. Evaluate algorithm
7. Test algorithm

1.2 Flavours of ML

Supervised learning:

- Output is continuous: regression
- Output is discrete: classification

Unsupervised learning: Clustering

1.3 Common issues in ML

- **Overfitting - High Variance**: The model is too specific to the data. It fits it very well, but doesn't generalise well.
- **Underfitting - High Bias**: The model is too general, it doesn't explain the training data well, and it doesn't explain new data that well either.
- **Data Leakage**: Some information from the test set "leaks" into the training/ validation set, and so the model will be biased to the seen data.
- **Dataset bias**: The training or validation set is not representative of the general domain of the problem.

2 Data Preparation Methods

- **Cleaning**: Remove nonsensical/null values, put data in the right format
- **Feature Extraction**: use raw features and combine/ transform them to purify some information to aid learning: requires domain knowledge usually.
- **Feature Selection**: Improve interpretability and computational feasibility. Remove unnecessary features (based on domain knowledge or statistical analysis): univariate tests, forward stepwise selection, backward stepwise selection etc.

- **One-Hot Encoding**: For classification tasks with k classes, it is commonplace to represent different classes as pairwise orthogonal unit vectors in $\mathbb{R}^k$. This ensures that $||c_i - c_j|| = \mathit{constant}$ for every class, i.e. there is no bias towards any class. Example:

$$\begin{bmatrix} c_1 \\ c_4 \\ c_3 \\ \vdots \\ c_2 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- **Decorrelation**: remove linear dependencies from the features: whitening, PCA.

- **Normalisation**:

$$x_i \quad \rightarrow \quad \frac{x_i - \min \mathbf{x}}{\max \mathbf{x} - \min \mathbf{x}}.$$

- **Standardisation**: if X_k is assumed to be symmetrically, or even better, normally distributed with mean μ_k and STD σ_k, this gives better results than normalisation:

$$x_i \quad \rightarrow \quad \frac{x_i - \mu_k}{\sigma_k}$$

- **Data Augmentation**: artificially generate more data from our preexisting set by introducing small modifications to it.

3 General ML Methodology

3.1 Regularisation

Purpose: Improve the generalisability of the model and prevent overfitting.

General Idea: Penalise an individual parameter growing too large.

Example: Squared Loss

$$L(\theta) = ||X\theta - \mathbf{y}||^2 \quad \rightarrow \quad J(\theta) = ||X\theta - \mathbf{y}||^2 + \lambda\, ||\theta||^2$$

for some **regularisation parameter** $\lambda \in \mathbb{R}^+$. If we regularise using the L_2 norm of θ, then the algorithm is called **ridge regression**, if we use the L_1 norm, then it's called **lasso**.

3.2 Model Evaluation

Train-Validation-Test Split Common ratios: 60 - 20 - 20, 80 - 10 - 10, etc.

- **Training data**: we train our model using the training data
- **Validation data**: we evaluate a trained model on the (preferrably) unseen validation data according to some metric(s). Based the model performance, we can tune hyperparameters/ choose a different model.
- **Test data**: Should always be unseen data to avoid bias in the trained model towards previously seen data. If this is not done, it constitutes data leakage and our results will be meaningless. The results on the test data are the results that we usually report, because it tells us how well our model generalises.

Cross Validation : If we don't have enough data to perform a proper train-validation-test split, we perform cross-validation. We perform a train-test separation, say 80-20. Then, taking the training data, we split into k equal portions. Then we train our models k times on $k-1$ different portions and validate on the one that was left out. Then, this helps mitigate some of the bias in our dataset. We call this k-fold CV.

Stratified Sampling : ensures consistent representation of classes in training and validation sets, by sampling each subpopulation (class) of the dataset independently, and then merging the results.

Performance Metrics : Regression:

- Squared error:
$$SE = ||\mathbf{y} - \hat{\mathbf{y}}||_2^2$$
- Absolute error:
$$AE = ||\mathbf{u} - \hat{\mathbf{y}}||_1$$
- Coefficient of determination: (sometimes called explained variance)
$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$
- Root Mean Squared Error:
$$RMSE = \sqrt{\frac{SE}{m}}$$

  RMSE is useful, because it has the same units as the data, so we can directly observe how well our model is performing.

(Binary) Classification:

- Accuracy:
$$\frac{TP + TN}{P + N}$$
- Error rates - False Positive Rate:
$$FPR = \frac{FP}{TN + FP}$$
False Negative Rate:
$$FNR = \frac{FN}{FN + TP}$$
- Specificity:
$$1 - FPR$$
- Sensitivity:
$$1 - FNR$$
- Precision:
$$\frac{TP}{TP + FP}$$
- Recall:
$$\frac{TP}{TP + FN}$$
- F1 - score:
$$2\frac{Prec \cdot Recall}{Prec + Recall}$$

The above pairings of classification metrics are "working against each other": one of them increasing will mean a decrease of their pair.

ROC Curve: Receiver Operation Characteristic Curve: TPR vs FPR

Precision-Recall Curve: duh

Average Precision (AP): integral of the precision recall curve

Metrics for multiclass classification:

- Report One-vs-All scores
- Confusion Matrix: for k classes it is a $k \times k$ matrix $C = [c_{ij}]$, where $c_{ij} =$ # of how many times an object of class c_j was predicted to be class c_i.

4 Regression Methods

4.1 Linear Regression

One of the simplest models f relies on the further assumption that Y depends linearly on $\mathbf{X}$. In particular,

$$\hat{Y} = f(\mathbf{X}, \theta) = \mathbf{X}\theta + b$$

where b is called the **bias**. No Then, we can quantify the goodness-of-fit of the model, by the squared error, specified by

$$L(\theta, b) = \, ||\mathbf{y} - f(X, \theta, b)||\,^2$$

Then, to find the optimal set of parameters theta, we can differentiate

$$\begin{aligned} \frac{\partial L}{\partial \theta} &= \frac{\partial}{\partial \theta} \, ||\mathbf{y} - f(X, \theta, b)||\,^2 \\ &= 2\,||\mathbf{y} - f(X, \theta, b)||\,^2 \frac{\partial}{\partial \theta} \, ||\mathbf{y} - f(X, \theta, b)|| \\ &= 2\,||\mathbf{y} - f(X, \theta, b)|| \, \frac{(\mathbf{y} - f(X, \theta, b))^T}{2\,||\mathbf{y} - f(X, \theta, b)||} \frac{\partial}{\partial \theta} \mathbf{y} - f(X, \theta, b) \\ &= (\mathbf{y} - X\theta)^T \cdot -X \end{aligned}$$

Now to find the minimum, set $\frac{\partial L}{\partial \theta} = 0$.

$$\begin{aligned} 0 &= -\mathbf{y}^T X + \theta^T X^T X \\ \mathbf{y}^T X &= \theta^T X^T X \\ \mathbf{y}^T X (X^T X)^{-1} &= \theta^T \end{aligned}$$

Hence

$$\theta = (X^T X)^{-1} X^T \mathbf{y}$$

The above is called the **normal equation**. An alternative way to solve the above problem is **gradient descent**. Iterate

$$\theta \leftarrow \theta - \alpha \cdot \nabla L$$

for some appropriate $\alpha \in \mathbb{R}$. Then it can be shown that the θ converges to the minimal solution. α is called the **learning rate**.

Regularised Normal Equation

$$\theta = (X^T X + \lambda I)^{-1} X^T$$

4.1.1 Basis Expansion

Provided we have k real features, we can modify our space in which we consider the linearity of our model. We can transform a set of observations $\mathbf{x}$ into the new space through an arbitrary function $h : \mathbb{R}^k \to \mathbb{R}^n$ by calculating $h(\mathbf{x})$. then, our model changes to:

$$f_\theta(X) = X\theta \quad \to \quad f_\theta(X) = h(X)\theta.$$

Crucially, f_θ is still linear in the space defined by h, but the components of h may be non-linear!

Special Case: Polynomial regression: the h_i components of h are different combinations of the input features.

4.2 Logistic Regression

Model the classification problem using the **logistic function**:

$$\sigma_\theta(\mathbf{x}) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}} \qquad \text{where } \mathbf{x}, \theta \in \mathbb{R}^{n \times 1}$$

It is easy to see that $\sigma_\theta(\mathbf{x}) \geq 0.5$ iff $\theta^T \mathbf{x} \geq 0$ and less than 0.5 otherwise.

$$B = \{\mathbf{x} \mid \sigma_\theta(\mathbf{x}) = 0.5\}$$

is called the **decision boundary** of the model.

This model allows for **binary classification**

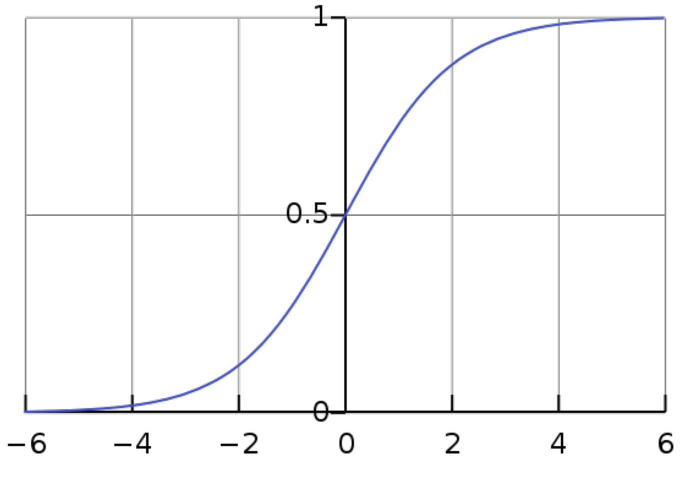

Figure 1: Logistic function $\sigma_\theta(\mathbf{x})$

The loss function for logistic regression is rooted in information gain, so L is defined to be the cross-entropy of the data:

$$L(\theta) = -\sum_{i=1}^{m} y_i \log(f_\theta(\mathbf{x}_i)) + (1 - y_i) \log(1 - f_\theta(\mathbf{x}_i))$$

Optimise this with gradient descent as before.

4.2.1 Multi-Class Logistic Regression

One-vs-All : For k classes train k classifiers: for each class c_i one class is c_i and one class is everything else. Then, to predict, take prediction with highest confidence.

One-vs-One : For k classes train $\binom{k}{2}$ classifier: one for each pair of c_i, c_j classes. Then, take a majority vote.

5 Bayesian Decision Theory

Main idea of Bayesian Decision Theory: minimise expected loss.

5.1 MLE

Assume set of distributions: $\{p_\theta \mid \theta \in \Theta\}$.
Assume the data is an i.i.d. sample of the above family: $X_1, \ldots, X_n \sim p_\theta$.
Then, attempt to estimate the true θ that generated the sample:

$$\theta_{MLE} = \arg\max_{\theta \in \Theta} \mathbb{P}[X \mid \theta]$$

where $\mathbb{P}[X \mid \theta]$ is called the **likelihood function**.

Pros

- Easy to compute
- Interpretable
- Nice asymptotic properties
- Invariant under reparametrisation:

$$g(\theta_{MLE}) = (g(\theta))_{MLE}$$

Cons

- Point estimate
- Overfitting

5.2 MAP

Similar assumptions to MLE. Instead of the likelihood function, estimate the **posterior distribution** of θ. We wanna estimate $\mathbb{P}[X \wedge \theta]$, we can use Bayes' Rule:

$$\mathbb{P}[\theta \mid X] = \frac{\mathbb{P}[X \mid \theta]\mathbb{P}[\theta]}{\mathbb{P}[X]}$$

here $\mathbb{P}[X]$ is constant, so we need to minimise

$$\theta_{MAP} = \arg\max_{\theta \in \Theta} \mathbb{P}[\theta \mid X] = \arg\max_{\theta \in \Theta} \mathbb{P}[X \mid \theta]\mathbb{P}[\theta]$$

Pros

- Easy to compute
- Intrerpretable
- Avoids overfitting by putting a prior on θ - Regularisation
- Converges to MLE

Cons

- Point estimate
- Not invariant under reparametrisation:

$$g(\theta_{MAP}) \neq (g(\theta))_{MAP}$$

- Must assume prior on θ

5.3 Density Estimation

Local density estimation assume that the density of the data is locally constant. Then for some volume V around some point $\mathbf{X} = [X_1, \dots, X_P]^T$, if we have K samples in the volume out of a total of N samples, then define

$$\mathbb{P}[X] = \frac{K}{NV}.$$

How we pick K and V determines the method.

Histograms: split up the space into a grid of cubes called **bins**. Then, we fixed V and we'll get K by counting up the samples in every bin.

Pro: Cheap

Con: often too crude.

Parzen density estimation: Given some point X, assume that the density is some local function of nearby samples. So define:

$$\mathbb{P}[X] = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{V}\cdot K\left(\frac{X - x_n}{h}\right)$$

where h is the window size, i.e. it controls the locality of the **kernel function** K. The kernel functions must be symmetric probability density functions themselves.
Examples:

- Tophat kernel: K is a uniform density function between $-h/2$ and $h/2$.
- Gaussian: $K = \mathcal{N}(0, h/2)$.
- Exponential
- Linear

5.4 K-Nearest Neighbours

If we fix K and let V grow according to

$$\mathbb{P}[X] = \frac{K}{NV}$$

then we can estimate class likelihoods according to Bayes' Rule:

$$\mathbb{P}[C_i \mid X] = \frac{\mathbb{P}[X \mid C_i]\mathbb{P}[C_i]}{\mathbb{P}[X]} = \frac{K_i}{K}$$

6 SVMs

7 Neural Networks and Deep Learning