COMP30030: Introduction to Artificial Intelligence

Neil Hurley

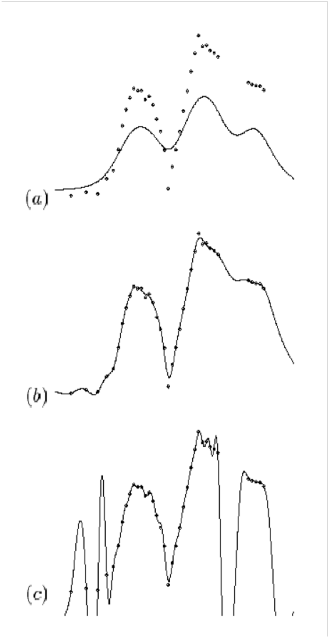
School of Computer Science
University College Dublin
neil.hurley@ucd.ie

November 1, 2018

Overfitting I

- The training data contains information about the regularities in the mapping from input to output. But it also contains noise
- The target values may be unreliable.
- There will be accidental regularities just because of the particular training cases that were chosen.
- When we fit the model, it cannot tell which regularities are real and which are caused by sampling error.
- If the model is very flexible it can model the sampling error really well. This is a disaster.

Overfitting II



Regularisation I

- While the loss function captures the error incurred over the training instances, we are most interested in how well the system performs on unseen data.
- To avoid overfitting, we can include a regulisation term in the optimisation problem:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \mathcal{L}(y_j, \sum_{i=1}^{m} w_i x_i) + \lambda \mathcal{R}(\mathbf{w})$$

- The regularisation term imposes some structure on the allowed solutions. For example, $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|^2$ constrains the norm (size) of the weight vector. The constant λ determines the strength of the regularisation effect.

Classification Algorithms

- Many different classification algorithms can be formulated in the above manner:
  1. Logistic regression. (Logistic loss function)
  2. Ridge regression. (Square loss function)
  3. Support vector machines (SVMs) (Hinge loss function)

Optimisation

- The process as described above allows us to turn the learning problem into an optimisation problem.
- Since the weights $\mathbf{w}$ can take values in $\mathbb{R}$, methods from calculus can be applied to solving these problems.
- In some cases (e.g. for squared loss function) it is possible to find an exact expression for the optimal weights.
- Otherwise, an approximate techniques, such as gradient descent is used to find the weights.

Ridge Regression I

- We could like to find the set of weights $(w_0, w_1, \ldots, w_m)$, that minimise the squared loss on the training set.
- The training set consists of a set of n instances which are m-dimensional vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ e.g.

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})^T$$

- We can gather these together into an $n \times m$ matrix, where the *rows* of the matrix are the feature vector for each instance.

$$\mathrm{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}$$

- Bear in mind that this is a **training** set of known instances — all of these x_{ij} are fixed known numbers.

Ridge Regression II

- To obtain the output label, we need to compute:

$$w_0 + w_1 x_{i1} + w_2 x_{i2} + \dots w_m x_{im} = \hat{y}_i$$

- So far, we **do not know** $\mathbf{w} = (w_0, w_1, \dots, w_m)^T$. We want to find this vector so that the squared loss is minimised.
- For convenience, we can gather the **bias** into the matrix by adding a column of ones at the beginning:

Ridge Regression III

$$\mathrm{X} = \begin{pmatrix} \color{red}{1} & x_{11} & x_{12} & \cdots & x_{1m} \\ \color{red}{\vdots} & \vdots & \vdots & \vdots & \vdots \\ \color{red}{1} & x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}$$

- Now we can write all the equations together into a single equation

$$\mathrm{X}\mathbf{w} = \hat{\mathbf{y}}$$

- The squared loss requires that we minimise

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- Putting the above together, we get

$$\sum_{i=1}^{n} (y_i - \sum_{j=0}^{m} x_{ij} w_j)^2$$

Ridge Regression IV

- In matrix vector notation, this can be written as

$$\|\mathbf{y} - \mathrm{X}\hat{\mathbf{y}}\|^2$$

- In Ridge Regression, we add in a regularisation term, so, given some $\lambda > 0$, overall we want to minimise

$$\|\mathbf{y} - \mathrm{X}\hat{\mathbf{y}}\|^2 + \lambda\|\mathbf{w}\|^2$$

- This problem can be solved using multivariate calculus – the unknowns are the w_i, so we differentiate w.r.t. w_i and set the result to zero:
- You might remember that

$$\frac{d}{dx}(f(x))^2 = 2f(x)\frac{d}{dx}f(x)$$

Solving Ridge Regression Optimisation I

- Using the above, the requirement in our problem is to compute

$$
\begin{aligned}
&\frac{\partial}{\partial w_k}\left(\sum_{i=1}^{n}(y_i - \sum_{j=0}^{m} x_{ij} w_j)^2 + \lambda \sum_{j=0}^{m} w_j^2\right) \\
&= \sum_{i=1}^{n} \frac{\partial}{\partial w_k}\left((y_i - \sum_{j=0}^{m} x_{ij} w_j)^2\right) + \lambda \sum_{j=0}^{m} \frac{\partial}{\partial w_k} w_j^2 \\
&= \sum_{i=1}^{n} 2(y_i - \sum_{j=0}^{m} x_{ij} w_j) \frac{\partial}{\partial w_k}\left(y_i - \sum_{j=0}^{m} x_{ij} w_j\right) + \lambda 2 w_k
\end{aligned}
$$

Solving Ridge Regression Optimisation II

$$= 2\left(\sum_{i=1}^{n}(y_i - \sum_{j=0}^{m} x_{ij}w_j)(-x_{ik}) + \lambda w_k\right)$$

$$= 2\left(\sum_{i=1}^{n} -y_i x_{ik} + \sum_{j=0}^{m}\sum_{i=1}^{n} x_{ij}x_{ik}w_j + \lambda w_k\right)$$

- Now set the derivative to zero (we can drop the constant 2):

$$\sum_{i=1}^{n} y_i x_{ik} - \left(\sum_{j=0}^{m}\sum_{i=1}^{n} x_{ij}x_{ik}w_j + \lambda w_k\right) = 0$$

- There is one equation here for each (unknown) value of w_k, so $m+1$ equations in total.
- It is easier to write these together in a matrix vector equation:

$$\mathrm{X}^T\mathbf{y} - \mathrm{X}^T\mathrm{X}\mathbf{w} - \lambda\mathrm{I}\mathbf{w} = 0$$

Solving Ridge Regression Optimisation III

- Or:

$$(\mathrm{X}^{T}\mathrm{X} + \lambda \mathrm{I})\mathbf{w} = \mathrm{X}^{T}\mathbf{y}$$

- Which can be solved as:

$$\mathbf{w} = (\mathrm{X}^{T}\mathrm{X} + \lambda \mathrm{I})^{-1}\mathrm{X}^{T}\mathbf{y}$$

Example I

- Suppose $m = 2$, and that the **bias is zero** i.e. $w_0 = 0$.
- So, each feature vector is just a 2-dimensional vector $\mathbf{x}_i = (x_{i1}, x_{i2})$.
- Suppose we are carrying out **binary classification**, and the labels are $y \in \{-1, 1\}$.
- Suppose we have a training set of 4 instances. Say:

$$\begin{aligned} \mathbf{x}_1 &= (4, 1) & y_1 &= 1 \\ \mathbf{x}_2 &= (-1, 2) & y_2 &= 1 \\ \mathbf{x}_3 &= (0, -1) & y_3 &= -1 \\ \mathbf{x}_4 &= (3, -2) & y_4 &= -1 \end{aligned}$$

Example II

- The goal is to find w_1 and w_2, so that the line

$$w_1 x + w_2 y = 0$$

  Separates the two classes (i.e. the line is the set of points (x, y) that satisfy this equation).

- Let's do the algebra:

$$\mathrm{X}^T\mathrm{X} = \begin{pmatrix} 4 & -1 & 0 & 3 \\ 1 & 2 & -1 & -2 \end{pmatrix} \begin{pmatrix} 4 & 1 \\ -1 & 2 \\ 0 & -1 \\ 3 & -2 \end{pmatrix} = \begin{pmatrix} 26 & -4 \\ -4 & 10 \end{pmatrix}$$

$$\mathrm{X}^T\mathbf{y} = \begin{pmatrix} 4 & -1 & 0 & 3 \\ 1 & 2 & -1 & -2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 6 \end{pmatrix}$$

Example III

- Take $\lambda = 1$

$$\mathrm{X}^T\mathrm{X} + \lambda\mathrm{I} = \begin{pmatrix} 26 + 1 & -4 \\ -4 & 10 + 1 \end{pmatrix} = \begin{pmatrix} 27 & -4 \\ -4 & 11 \end{pmatrix}$$

$$(\mathrm{X}^T\mathrm{X} + \lambda\mathrm{I})^{-1} = \frac{1}{27 \times 11 - (-4) \times (-4)} \begin{pmatrix} 9 & 4 \\ 4 & 25 \end{pmatrix}$$

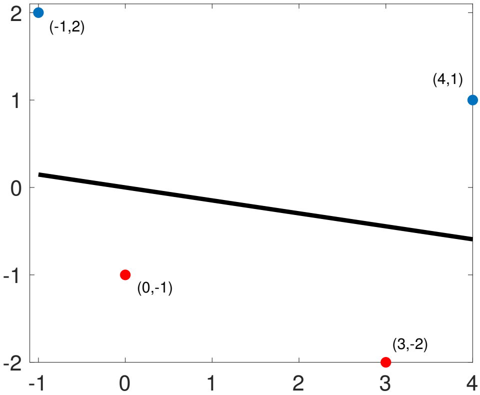
$$= \frac{1}{281} \begin{pmatrix} 11 & 4 \\ 4 & 27 \end{pmatrix}$$

- Therefore:

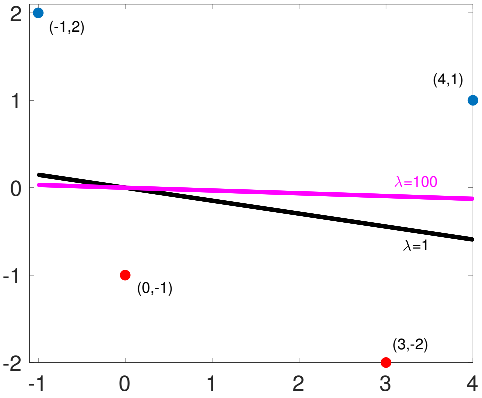
$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \frac{1}{281} \begin{pmatrix} 11 & 4 \\ 4 & 27 \end{pmatrix} \begin{pmatrix} 0 \\ 6 \end{pmatrix} = \frac{1}{281} \begin{pmatrix} 24 \\ 162 \end{pmatrix}$$

$$= \begin{pmatrix} 0.0854 \\ 0.5765 \end{pmatrix}$$

Ridge Regression Result



Ridge Regression Result



Training and Test Sets I

- Given a set of training instances, with their desired outputs, Machine Learning analysts typically divide it into two disjoint sets – a training set and a test set.
- Typically, the training set contains 80% of the instances and the test set the remaining 20% (chosen randomly).
- The training set is used to train the weights of the algorithm, by minimising the regularised loss function – the optimisation technique may achieve low error rates on the training set.
- The test set is then used to test how well the trained algorithm generalises to unseen examples. The error will typically be higher on the test set instances – but this error is closer to the expected performance of the classifier when it is used in practise.

Classification Algorithm Performance

- Some performance measures

Confusion Matrix

A confusion matrix shows the number of correct and incorrect predictions made by the classification model compared to the actual outcomes (target value) in the data. The matrix is NxN, where N is the number of target values (classes). Performance of such models is commonly evaluated using the data in the matrix. The following table displays a 2x2 confusion matrix for two classes (Positive and Negative).

Confusion Matrix		**Target**			
		Positive	Negative		
Model	Positive	a	b	*Positive Predictive Value*	a/(a+b)
	Negative	c	d	*Negative Predictive Value*	d/(c+d)
		Sensitivity	*Specificity*	**Accuracy** = (a+d)/(a+b+c+d)	
		a/(a+c)	d/(b+d)		

- **Accuracy** : the proportion of the total number of predictions that were correct.
- **Positive Predictive Value** or **Precision** : the proportion of positive cases that were correctly identified.
- **Negative Predictive Value** : the proportion of negative cases that were correctly identified.
- **Sensitivity** or **Recall** : the proportion of actual positive cases which are correctly identified.
- **Specificity** : the proportion of actual negative cases which are correctly identified.