Lecture 3: Linear classification, generalisation and regularisation

EMAT31530/Jan 2018/Raul Santos-Rodriguez

Have a look at ...

... Russell and Norvig (Ch. 18.2 18.4 18.6 18.7 18.8 18.9)

... Hastie, Tibshirani, Friedman. The elements of statistical learning, (Ch. 4.5 and 7)

... Python: `https://keras.io/`

... Python: `http://scikit-learn.org/`

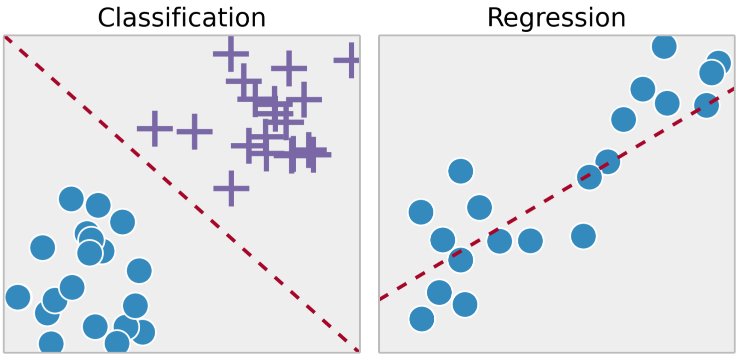
... Java: `http://www.cs.waikato.ac.nz/ml/weka/`

Outline

This lecture introduces the topics of linear classification, generalisation and regularisation. The goal of the lecture is for you to

- Understand the linear classification setting
- Understand the trade-off between bias and variance
- Understand how to make the most of your data

Classification vs Regression



Classification vs Regression

We are given a training dataset D_{train} of n instances of input-output pairs $\{\mathbf{x}_{1:n}, y_{1:n}\}$. Each input $\mathbf{x}_i \in \mathbb{R}^{d\times 1}$ is a vector with d attributes. The output (target) is y_i. Depending on the nature of y_i, there are different types of prediction tasks:

- Regression: y is a real number

$$y = \mathbf{x}^T \boldsymbol{\theta}$$

- Classification: y is discrete; yes/no (binary), one of K labels (multiclass), subset of K labels (multilabel)

$$y = sign(\mathbf{x}^T \boldsymbol{\theta})$$

Binary classification: $y \in \{+1, -1\}$

Score

Measures how confident we are in our prediction: $score = f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\theta}$

Margin

Measures how correct we are: $margin = y(\mathbf{x}^T\boldsymbol{\theta})$

Loss function

Loss function $L(y, f_{\boldsymbol{\theta}}(\mathbf{x}))$ measures how happy we are with the prediction

Question

When does a binary classifier err on an example?

- margin less than 0
- margin greater than 0
- score less than 0
- score greater than 0

First approach: Perceptron

The Perceptron Algorithm

1. Initialise t (number of iterations) to 1 and start with the all-zeroes weight vector $\boldsymbol{\theta}^1 = [0 \dots 0]^T$. Also let's automatically scale all examples $\mathbf{x}$ to have (Euclidean) length 1, since this doesn't affect which side of the hyperplane they are on.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{x}^T \boldsymbol{\theta} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t + \mathbf{x}$.
   - Mistake on negative: $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t - \mathbf{x}$.

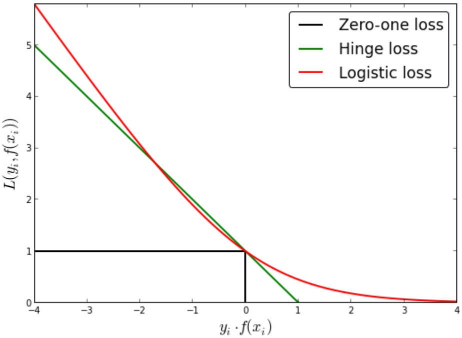
   $t \leftarrow t + 1$.

http://intelligence.org/files/AIPosNegFactor.pdf, Sec. 7.2

[Rosenblatt, 1958]

A general framework: Loss minimisation

Zero-one loss: $L_{0-1}(y, f_{\boldsymbol{\theta}}(\mathbf{x})) = \mathbf{1}[(\mathbf{x}^T\boldsymbol{\theta})y \leq 0]$



Loss minimisation

$$\arg\min_{\boldsymbol{\theta}} TrainLoss(\boldsymbol{\theta})$$

where

$$TrainLoss(\boldsymbol{\theta}) = \sum_{(\mathbf{x},y)\in D_{train}} L(y, f_{\boldsymbol{\theta}}(\mathbf{x}))$$

Gradient Descent

Gradient

The gradient $\nabla TrainLoss(\boldsymbol{\theta})$ is the direction that increases the loss the most.

Gradient Descent (GD)

$\boldsymbol{\theta} \leftarrow$ any point in the parameter space, e.g., $\boldsymbol{\theta} = [0 \ldots 0]^T$

do

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla TrainLoss(\boldsymbol{\theta})$$

until convergence **or** $t \leq max\_iterations$

Problem: each iteration requires going over all training examples (expensive when have lots of data!)

Stochastic gradient descent

Gradient Descent (GD)

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla \mathit{TrainLoss}(\boldsymbol{\theta})$

Stochastic Gradient Descent (SGD)

For each $(\mathbf{x}, y) \in D_{train}$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla L(\boldsymbol{\theta}, \mathbf{x}, y)$

$\boxed{\text{Quantity, not quality!}}$

Question

What's the true objective of machine learning?

- Minimise error on the training set
- Minimise error on unseen future examples

Training error

Loss minimisation

$$\arg\min_{\boldsymbol{\theta}} \mathit{TrainLoss}(\boldsymbol{\theta})$$

where

$$\mathit{TrainLoss}(\boldsymbol{\theta}) = \sum_{(\mathbf{x},y)\in D_{train}} L(y, f_{\boldsymbol{\theta}}(\mathbf{x}))$$

Is this a good objective?

A classifier that minimises the training error

1. **During training**: store all pairs in D_{train}
2. **During prediction**: predict the output for x_{new}

   if x_{new} in D_{train} **then return** its corresponding y

   else return error!

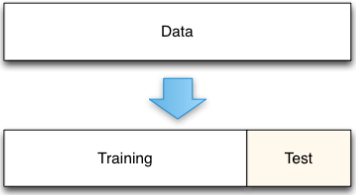
Minimises the objective perfectly (zero), but ...

Key idea: our goal is to minimise error on unseen future examples..

... but we dont have unseen examples. The next best thing:

Test set

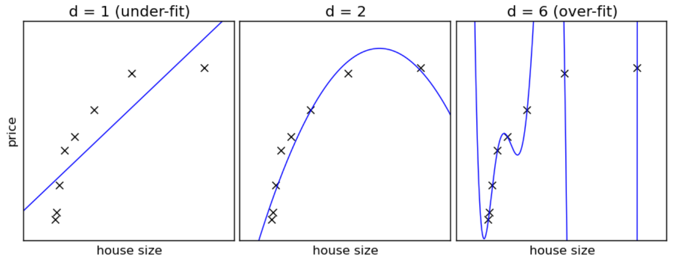
Test set D_{test} contains examples not used for training.



Overfitting example: regression

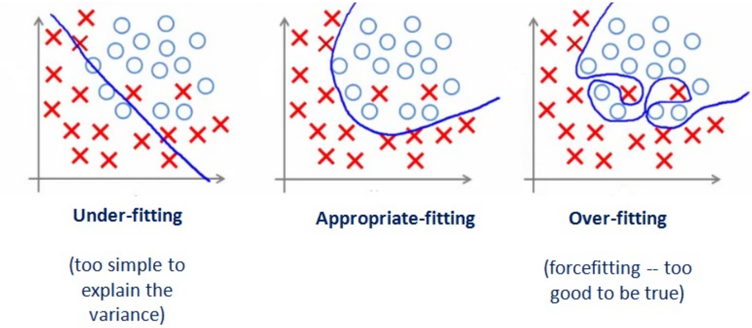
House prices

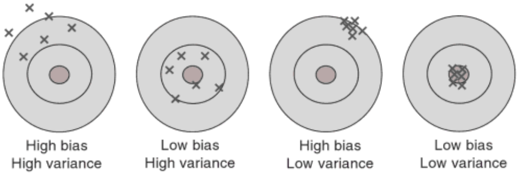
Imagine that you would like to build an algorithm which will predict the price of a house given its size. Naively, we'd expect that the cost of a house grows as the size increases, but there are many other factors which can contribute. Imagine we approach this problem with polynomial regression. We can tune the degree d to try to get the best fit.



[sklearn tutorial]

Overfitting example: classification





Bias Variance Decomposition. Figure 1. The bias-variance decomposition is like trying to hit the bullseye on a dartboard. Each dart is thrown after training our "dart-throwing" model in a slightly different manner. If the darts vary wildly, the learner is *high variance*. If they are far from the bullseye, the learner is *high bias*. The ideal is clearly to have both low bias and low variance; however this is often difficult, giving an alternative terminology as the bias-variance "dilemma" (*Dartboard analogy*, Moore & McCabe (2002))

Bias: how good is the hypothesis class?

Variance: how good is the learned predictor relative to the hypothesis class?

$\boxed{\text{Test error is bias + variance}}$

[Encyclopedia of Machine Learning]

Training error and test error

Underfitting: bias too high
Overfitting: variance too high
Fitting: balancing bias and variance

Controlling the size of the hypothesis class

Linear predictors are specified by parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$.

Two alternatives:

1. **Keep the dimensionality d small**
2. **Keep the norm $||\theta||$ (length of θ) small: Regularisation**

Controlling the size of the hypothesis class: Dimensionality

Manual feature selection:

- Add features if they reduce test error
- Remove features if they don't decrease test error

Automatic feature selection (beyond the scope of this class):

- Forward selection: Maximum Relevance Minimum Redundancy (MRMR)
- L1 regularisation: Lasso

Controlling the size of the norm: Regularisation

Regularised objective

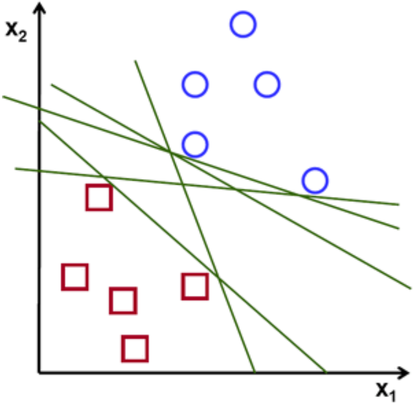
$$\min_{\boldsymbol{\theta}} \mathit{TrainLoss}(\boldsymbol{\theta}) + \frac{\lambda}{2}||\boldsymbol{\theta}||^2$$

$\boxed{\text{Idea: shrink the weights towards zero by } \lambda}$

Support Vector Machines

Idea: Large margin

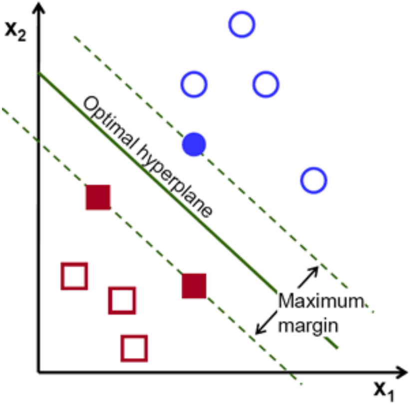
Objective: *HingeLoss* + *regularisation*



Support Vector Machines

Idea: Large margin

Objective: *HingeLoss* + *regularisation*



Controlling the size of the norm: Ridge regression

In regression, all the answers so far are of the form

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

They require the inversion of $\mathbf{X}^T\mathbf{X}$. This can lead to problems if the system of equations is poorly conditioned. A solution is to add a small element to the diagonal:

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_d)^{-1}\mathbf{X}^T\mathbf{y}$$

This is the ridge regression estimate. It is the solution to the following regularised quadratic cost function

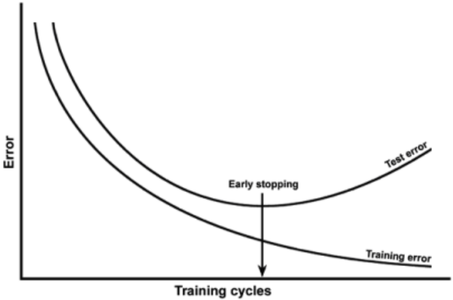
$$J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda\boldsymbol{\theta}^T\boldsymbol{\theta}$$

Controlling the size of the norm: Early stopping

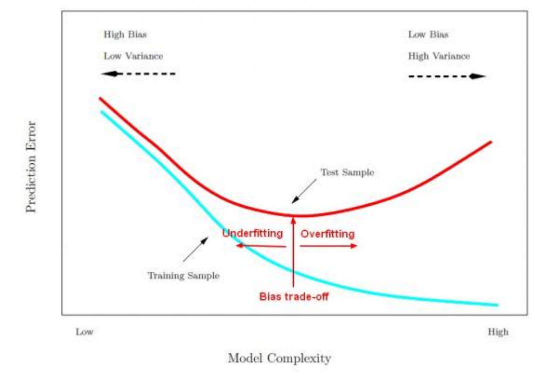
Idea simply make *max_it* smaller.

Intuition fewer updates, then $||\boldsymbol{\theta}||$ can't get too big.

Lesson try to minimise the training error, but don't try too hard.



Summary so far



$\boxed{\text{Simple solutions that fit the data well}}$

How do we choose the hyperparameters

Hyperparameters

Properties of the learning algorithm (features, regularisation parameter λ, number of iterations *max_it*, step size α, etc.).

Choose hyperparameters to minimise D_{train} error? No - solution would be to include all features, set $\lambda = 0$, $max\_it \to \infty$.

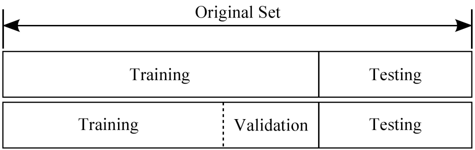
Choose hyperparameters to minimise D_{test} error? No - choosing based on D_{test} makes it an unreliable estimate of error!

Validation

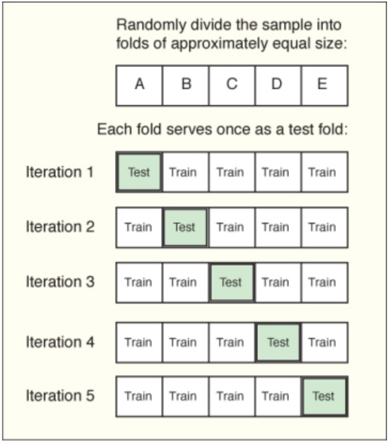
Solution: randomly take out 10-50% of training and use it instead of the test set to estimate test error.

Validation set

A validation (development) set is taken out of the training data which acts as a surrogate for the test set.



Cross-validation



Thomas W. Miller, 2013

Machine learning in practice (I)

If our algorithm shows high bias, try

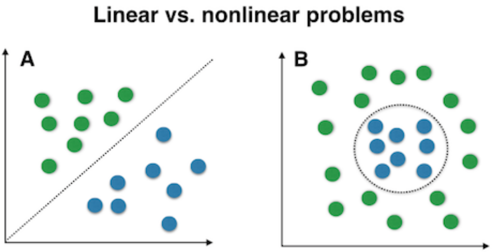
- **Adding more features**. In the example of predicting home prices, it may be helpful to make use of information such as the neighbourhood the house is in, the year the house was built, the size of the lot, etc. Adding these features to the training and test sets can improve a high-bias estimator.
- **Using a more sophisticated model**. Adding complexity to the model can help improve on bias. For a polynomial fit, this can be accomplished by increasing the degree. Each learning technique has its own methods of adding complexity.
- **Using fewer samples**. Though this will not improve the classification, a high-bias algorithm can attain nearly the same error with a smaller training sample. For algorithms which are computationally expensive, reducing the training sample size can lead to very large improvements in speed.
- **Decreasing regularisation**. Regularisation is a technique used to impose simplicity in some machine learning models, by adding a penalty term that depends on the characteristics of the parameters. If a model has high bias, decreasing the effect of regularisation can lead to better results.

Machine learning in practice (II)

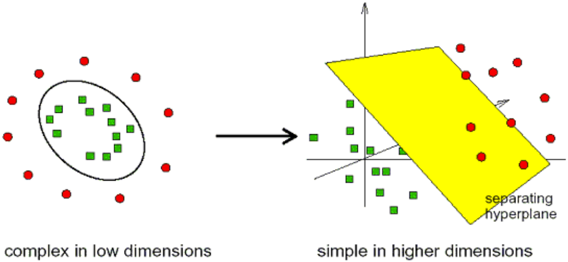
If our algorithm shows high variance, try

- **Using fewer features**. Using a feature selection technique may be useful, and decrease the overfitting of the estimator.
- **Using more training samples**. Adding training samples can reduce the effect of over-fitting, and lead to improvements in a high variance estimator.
- **Increasing regularisation**. Regularisation is designed to prevent over-fitting. In a high-variance model, increasing regularisation can lead to better results.

Linear vs Non-linear

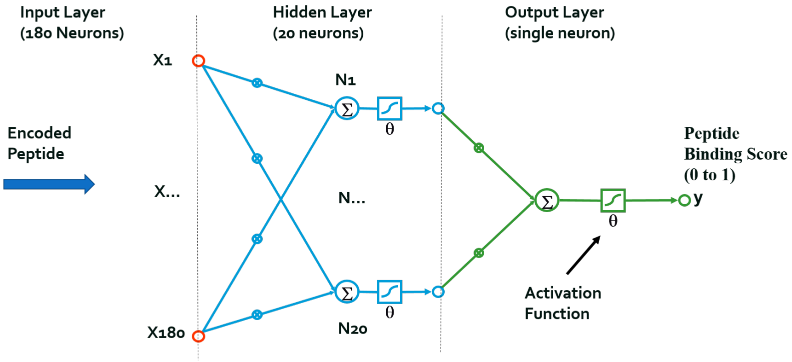


Non-linear prediction



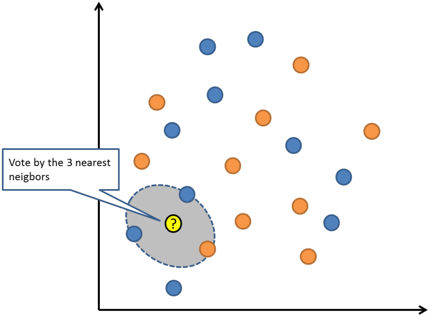
E.g., Kernel methods (Suppor Vector Machines)

Non-linear prediction



E.g., Neural Networks (Multi-layer Perceptron)

Non-linear prediction



E.g., Non-parametric methods (K-Nearest Neighbours)

Summary

We've looked at:

- Supervised learning: perceptron
- Loss minisation
- Generalisation: bias and variance
- Regularisation

A few things to have a look at:

- Proof that, if the data is separable, the perceptron algorithm converges!
- `https://techcrunch.com/2017/01/30/is-a-master-algorithm-the-solution-to-our-machine-learning-problem`

Next lecture

We will introduce recommender systems. We will also start discussing the concepts of unsupervised learning:

- clustering

- dimensionality reduction (PCA)