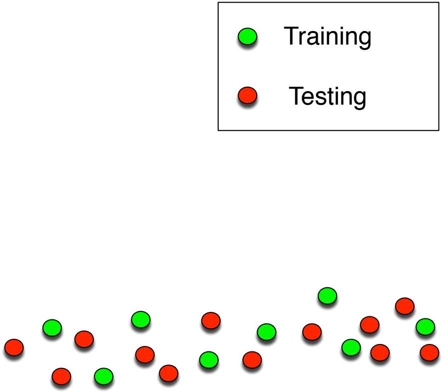
L2: Supervised Learning: KNN, Perceptron, Logistic Regression

Prof Javen Shi

5 August 2020

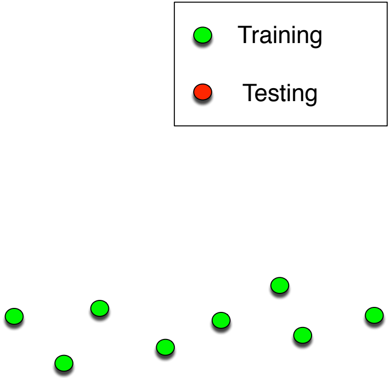
Overfitting

Fitting the training data too well cause a problem.



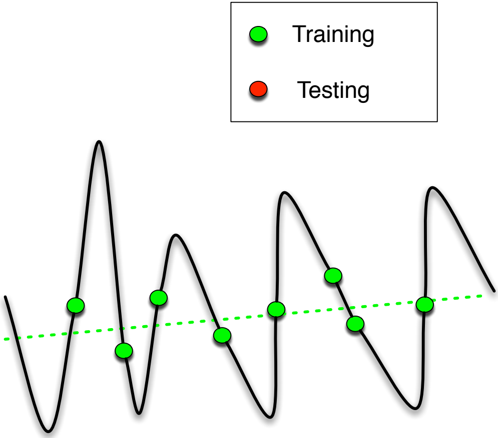
Overfitting

Train on training data (testing data are hidden from us).



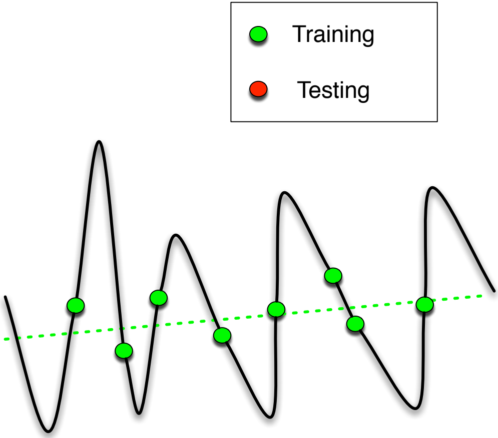
Overfitting

Two possible models. Which model fits the training data better?



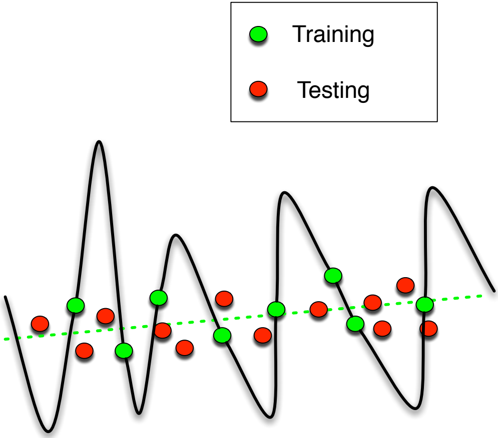
Overfitting

Two possible models. Which model fits the testing data better?



Overfitting

Reveal the testing data.



Occam's Razor

"The simplest model that fits the data is also the most plausible."

Occam's Razor

"The simplest model that fits the data is also the most plausible."

Two questions:

1. What does it mean for a model to be simple?

Occam's Razor

"The simplest model that fits the data is also the most plausible."

Two questions:

1. What does it mean for a model to be simple?
2. Why simpler is better?

Simpler means less complex

Model complexity – two types:

1. complexity of the function g: order of a polynomial, MDL
    - a straight line (order 0 or 1) is simpler than a quadratic function (order 2).
    - computer program: 100 bits simpler than 1000 bits
2. complexity of the space $\mathcal{G}$: $|\mathcal{G}|$, VC dimension, noise-fitting, ...
    - Often used in proofs.

Simpler is better

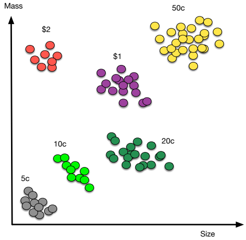
1. What do you mean by "better"?
   - smaller generalisation error (*e.g.* smaller expected testing error).
2. Why simpler is better?
   - Practically implemented by regularisation techniques, which will be covered today.
   - Theoretically answered by generalisation bounds, beyond this course.

What do we know so far?

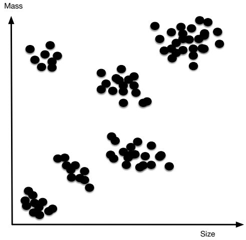
- What's machine learning?
- All you care is the testing error (not the training error).
- Train too well is not good (overfitting).
- The simplest model that fits the data is also the most plausible (Occam's Razor).

Recap continues

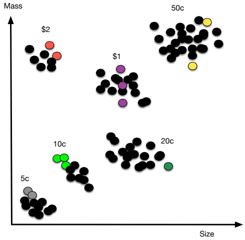
3 types of learning:



(a) Supervised



(b) Unsupervised

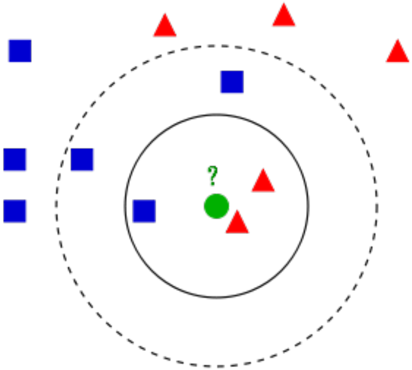


(c) Semi-supervised

Figure : Recognising coins by the features of their mass and size

1st glance at a classification algorithm

K Nearest Neighbour (KNN):



KNN: majority vote of the k Nearest Neighbours of the test point (green). If $k = 3$, the test point is predicted as red, if $k = 5$, the test point is predicted as blue. Picture courtesy of wikipedia

Questions

Thousands of classification algorithms out there. How can we possibly study they all?

Many algorithms come out every year, how do we keep up with them?

Answers

Learning theory analyses sets of algorithms' behaviour (beyond the scope of the course)

Many algorithms can be formulated in a unified framework called Empirical Risk Minimisation (ERM).

Risks

Given a loss $\ell(\mathbf{x}, y, \mathbf{w})$,

(True) Risk

$$R(\mathbf{w}, \ell) = \mathbb{E}_{(\mathbf{x},y)\sim p}\, \ell(\mathbf{x}, y, \mathbf{w})$$

Empirical Risk

$$R_n(\mathbf{w}, \ell) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{x}_i, y_i, \mathbf{w})$$

For example:

(SVM) Hinge loss $\ell_H(\mathbf{x}, y, \mathbf{w}) = \max\{0, 1 - y(\langle \mathbf{x}, \mathbf{w}\rangle)\}$.

Perceptron loss $\ell_{pern}(\mathbf{x}, y, \mathbf{w}) = \max\{0, -y\langle \mathbf{x}, \mathbf{w}\rangle\}$.

Zero-one loss $\ell_{0/1}(\mathbf{x}, y, \mathbf{w}) = \mathbf{1}_{\{g(\mathbf{x})\neq y\}}$. Here $\mathbf{1}_{\{a\}}$ is an indicator function which $= 1$ when a is true, $= 0$ otherwise.

Generalisation error

Generalisation error is the error rate over all possible testing data from the distribution P, that is the risk w.r.t. zero loss,

$$R(g) = \mathbb{E}_{(\mathbf{x},y)\sim p}[\mathbf{1}_{\{g(\mathbf{x})\neq y\}}] = P(g(\mathbf{x}) \neq y)$$

Empirical risk for zero-one loss is

$$R_n(g) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{g(\mathbf{x}_i)\neq y_i\}},$$

which is in fact the training error.

Regularised ERM

Regularised Empirical Risk Minimisation

$$g_n = \operatorname*{argmin}_{g \in \mathcal{G}} R_n(g) + \lambda \Omega(g),$$

where $\Omega(g)$ is the regulariser, *e.g.* $\Omega(g) = \|g\|^2$. $\mathcal{G}$ is the hypothesis set. Unfortunately, above is not convex. It turns out that one can optimise

$$\mathbf{w}_n = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} R_n(\mathbf{w}, \ell) + \lambda \Omega(\mathbf{w}),$$

as long as ℓ is a surrogate loss (brief def here) of the zero-one loss.

Decision functions (Recall)

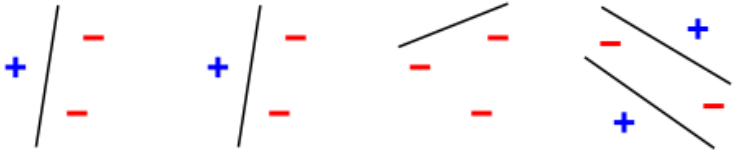
Linear decision function $g(\mathbf{x};\mathbf{w}) = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b)$ are often used (sign here is for binary classification). Here $\mathbf{x}, \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$.
Since $\langle \mathbf{x}, \mathbf{w} \rangle + b = \langle [\mathbf{x}; 1], [\mathbf{w}; b] \rangle$, for simplicity one often write

Binary $\quad g(\mathbf{x};\mathbf{w}) = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle)$.

Multi-class $\quad g(\mathbf{x};\mathbf{w}) = \underset{y}{\operatorname{argmax}}(\langle \mathbf{x}, \mathbf{w}_y \rangle)$.

Separability

Not all data are linearly separable (*e.g.* the 4-th one).



Picture courtesy of wikipedia

To deal with linearly non-separable case, non-linear decision functions are needed (often used in kernel methods).

Perceptron Algorithm

Assume $g(\mathbf{x};\mathbf{w}) = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle)$, where $\mathbf{x}, \mathbf{w} \in \mathbb{R}^d$, $y \in \{-1, 1\}$.

Input: training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, step size η, #iter T
Initialise $w_1 = \mathbf{0}$
for $t = 1$ **to** T **do**

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta \sum_{i=1}^{n} (y_i \, \mathbf{x}_i \, \mathbf{1}_{\{y_i \langle \mathbf{x}_i, \mathbf{w}_t \rangle < 0\}}) \tag{1}$$

end for
Output: $\mathbf{w}^* = \mathbf{w}_T$

The class of $\mathbf{x}$ is predicted via

$$y^* = \text{sign}(\langle \mathbf{x}, \mathbf{w}^* \rangle)$$

View it in ERM

$$\min_{\mathbf{w},\xi} \frac{1}{n}\sum_{i=1}^{n}\xi_i, \quad \text{s.t.} \quad y_i \langle \mathbf{x}_i, \mathbf{w}\rangle \geq -\xi_i, \xi_i \geq 0$$

whose unconstrained form is

$$\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}\max\{0, -y_i \langle \mathbf{x}_i, \mathbf{w}\rangle\} \Leftrightarrow \min_{\mathbf{w}} R_n(\mathbf{w}, \ell_{pern})$$

with Loss $\ell_{pern}(\mathbf{x}, y, \mathbf{w}) = \max\{0, -y\langle \mathbf{x}, \mathbf{w}\rangle\}$ and
Empirical Risk $R_n(\mathbf{w}, \ell_{pern}) = \frac{1}{n}\sum_{i=1}^{n} \ell_{pern}(\mathbf{x}_i, y_i, \mathbf{w})$.

$$\text{Sub-gradient} \quad \frac{\partial R_n(\mathbf{w}, \ell_{pern})}{\partial \mathbf{w}} = -\frac{1}{n}\sum_{i=1}^{n}(y_i\, \mathbf{x}_i\, \mathbf{1}_{\{y_i(\langle \mathbf{x}_i, \mathbf{w}_t\rangle)<0\}}).$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta' \frac{\partial R_n(\mathbf{w}, \ell_{pern})}{\partial \mathbf{w}} = \mathbf{w}_t + \eta' \frac{1}{n}\sum_{i=1}^{n}(y_i\, \mathbf{x}_i\, \mathbf{1}_{\{y_i(\langle \mathbf{x}_i, \mathbf{w}_t\rangle)<0\}})$$

Letting $\eta = \eta' \frac{1}{n}$ recovers the equation (1).

Logistic Regression for Binary Classification

For binary LR, one can assume

$$P(y = +1 | \, \mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-\langle \mathbf{w}, \mathbf{x} \rangle}}$$

Thus

$$\begin{aligned} P(y = -1 | \, \mathbf{x}; \mathbf{w}) &= 1 - P(y = +1 | \, \mathbf{x}; \mathbf{w}) \\ &= \frac{e^{-\langle \mathbf{w}, \mathbf{x} \rangle}}{1 + e^{-\langle \mathbf{w}, \mathbf{x} \rangle}} = \frac{1}{1 + e^{\langle \mathbf{w}, \mathbf{x} \rangle}} \end{aligned}$$

Above means

$$P(y | \, \mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-y \langle \mathbf{w}, \mathbf{x} \rangle}} \tag{2}$$

Alternative formulation

Alternatively if let $y \in \{0, 1\}$, one assumes

$$P(y = +1 | \mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-\langle \mathbf{w}, \mathbf{x} \rangle}}$$
$$P(y = 0 | \mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{\langle \mathbf{w}, \mathbf{x} \rangle}},$$

which means

$$P(y | \mathbf{x}; \mathbf{w}) = (\frac{1}{1 + e^{-\langle \mathbf{w}, \mathbf{x} \rangle}})^{y} (\frac{1}{1 + e^{\langle \mathbf{w}, \mathbf{x} \rangle}})^{(1-y)} \tag{3}$$

Because eq(3) is not as neat as eq(2), we will use eq(2) with $y \in \{-1, 1\}$.

Maximum Likelihood and Log loss

Maximum Likelihood

$$
\begin{aligned}
&\operatorname*{argmax}_{\mathbf{w}} \prod_{i=1}^{n} P(y_i | \mathbf{x}_i; \mathbf{w}) \\
&= \operatorname*{argmin}_{\mathbf{w}} -\log \left(\prod_{i=1}^{n} P(y_i | \mathbf{x}_i; \mathbf{w}) \right) \\
&= \operatorname*{argmin}_{\mathbf{w}} -\sum_{i=1}^{n} \log P(y_i | \mathbf{x}_i; \mathbf{w}) \quad \text{(log loss in ERM)}
\end{aligned}
$$

Gradient for binary class

Let

$$L(\mathbf{w}\,|X, Y) = -\sum_{i=1}^{n} \log P(y_i|\,\mathbf{x}_i; \mathbf{w})$$

$$\begin{aligned}
\frac{\partial L(\mathbf{w}\,|X, Y)}{\partial\, \mathbf{w}} &= \frac{\partial \sum_{i=1}^{n} \log\left(1 + e^{-y_i\langle \mathbf{w}, \mathbf{x}_i\rangle}\right)}{\partial\, \mathbf{w}} \quad \text{via eq(2)} \\
&= \sum_{i=1}^{n} \frac{e^{-y_i\langle \mathbf{w}, \mathbf{x}_i\rangle}}{1 + e^{-y_i\langle \mathbf{w}, \mathbf{x}_i\rangle}}(-y_i\,\mathbf{x}_i) \\
&= \sum_{i=1}^{n}(-y_i\,\mathbf{x}_i)(1 - P(y_i|\,\mathbf{x}_i; \mathbf{w}))
\end{aligned}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{\partial L(\mathbf{w}\,|X, Y)}{\partial\, \mathbf{w}}$$

Multi-class

For multi-class LR, let c be the number of classes. Let $\mathbf{w} = (\mathbf{w}_{y'})_{y' \in \mathcal{Y}}$, where $\mathbf{w}_{y'} \in \mathbb{R}^d$, thus $\mathbf{w} \in \mathbb{R}^{dc}$. One assumes

$$P(y|\,\mathbf{x};\mathbf{w}) = \frac{e^{\langle \mathbf{w}_y, \mathbf{x} \rangle}}{\sum_{y' \in \mathcal{Y}} e^{\left\langle \mathbf{w}_{y'}, \mathbf{x} \right\rangle}} \tag{4}$$

Note: the multi-class form can recover the binary form despite different appearance.

$$\begin{aligned} L(\mathbf{w}\,|X, Y) &= -\sum_{i=1}^{n} \log P(y_i|\,\mathbf{x}_i;\mathbf{w}) \\ &= \sum_{i=1}^{n} \log\Big(\sum_{y' \in \mathcal{Y}} e^{\left\langle \mathbf{w}_{y'}, \mathbf{x}_i \right\rangle}\Big) - \langle \mathbf{w}_y, \mathbf{x}_i \rangle \end{aligned}$$

Gradient for Multi-class

$$\frac{\partial L(\mathbf{w}\,|X, Y)}{\partial\,\mathbf{w}_y} = \sum_{i=1}^{n}\Big(\frac{e^{\langle \mathbf{w}_y, \mathbf{x}_i\rangle}}{\sum_{y'\in\mathcal{Y}} e^{\langle \mathbf{w}_{y'}, \mathbf{x}_i\rangle}}\,\mathbf{x}_i - \mathbf{x}_i\Big)$$

$$= \sum_{i=1}^{n}\mathbf{x}_i(P(y_i|\,\mathbf{x}_i;\mathbf{w}) - 1)$$

$$\frac{\partial L(\mathbf{w}\,|X, Y)}{\partial\,\mathbf{w}} = \Big(\frac{\partial L(\mathbf{w}\,|X, Y)}{\partial\,\mathbf{w}_y}\Big)_{y\in\mathcal{Y}}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\frac{\partial L(\mathbf{w}\,|X, Y)}{\partial\,\mathbf{w}}$$

That's all

Thanks!