Linear Regression

Prediction: $h_\theta(x) = \theta^T x$

Cost: $$J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient Descent:
for j=0 to n $\{tempj := \theta_j - \frac{2\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\}$
for j=0 to n $\{\theta_j := tempj\}$

Fitting nonlinear curves: $h_\theta(z) = \theta_0 + \theta_1 z$ where z is a feature vector $= x^2$

Regularisation: $$\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + R(\theta)$$ where R is a penalty function

Quadratic/L2 penalty: $R(\theta) = \theta^T\theta = \sum_{j=1}^{n}\theta_j^2$

L1 Penalty: $R(\theta) = \sum_{i=1}^{n}|\theta_j|$

Logistic Regression

Prediction: $h_\theta(x) = sign(\theta^T x)$

Cost: $$\frac{1}{m}\sum_{i=1}^{m}\log(1 + e^{-y^{(i)}\theta^T x^{(i)}})/log(2)$$

Gradient Descent:

Start with some θ

Repeat:

for j=0 to n $\{tempj := \theta_j + \frac{\alpha}{m}\sum_{i=1}^{m} y^{(i)}x_j^{(i)} \frac{e^{-y^{(i)}\theta^T x^{(i)}}}{1+e^{-y^{(i)}\theta^T x^{(i)}}}\}$

for j=0 to n $\{\theta_j := tempj\}$

$J(\theta)$ is convex, has a single minimum. Iteration moves downhill until it reaches the minimum

---

Fitting multiple classes:

Make it into a binary problem

- Train a classifier $h_\theta^{(i)}(x)$ for each class i
- Predicts the probability that y = i
- Training data: re-label data as y = -1 when y $\neq$ i and y = 1 when y = i

---

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \log(1 + e^{-y^{(i)}\theta^T x^{(i)}}) + \lambda\sum_{j=1}^{n}\theta_j^2$$

Regularisation:

Support Vector Machines

Prediction: same as Log Reg: $h_\theta(x) = sign(\theta^T x)$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \max(0, 1 - y^{(i)}\theta^T x^{(i)}) + \lambda\theta^T\theta$$

Cost:

Maximising the margin: maximising $\frac{y^{(i)}\theta^T x^{(i)}}{\theta^T\theta}$ is the same as minimising $-\frac{y^{(i)}\theta^T x^{(i)}}{\theta^T\theta}$

Gradient Descent: the cost function of SVM is non-continuous, so use sub-gradient

For $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \max(0, 1 - y^{(i)}\theta^T x^{(i)}) + \lambda\theta^T\theta$, subgradient with respect to θ_j is:

- $2\lambda\theta_j - \frac{1}{m}\sum_{i=1}^{m} y^{(i)}x_j^{(i)}\mathbb{1}(y^{(i)}\theta^T x^{(i)} \leq 1)$ where $\mathbb{1}(y^{(i)}\theta^T x^{(i)} \leq 1) = 1$ when $y^{(i)}\theta^T x^{(i)} \leq 1$ and zero otherwise.

So subgradient descent algorithm for SVMs is:

- Start with some θ
- Repeat:
  for j=0 to n
  $\{tempj := \theta_j - \alpha(2\lambda\theta_j - \frac{1}{m}\sum_{i=1}^{m} y^{(i)}x_j^{(i)}\mathbb{1}(y^{(i)}\theta^T x^{(i)} \leq 1))$
  for j=0 to n $\{\theta_j := tempj\}$

$J(\theta)$ is convex, has a single minimum. Iteration moves downhill until it reaches the minimum

---

Nonlinear decision boundary: add extra (polynomial features)

Regularisation: $\theta^T\theta = \sum_{j=1}^{n} \theta_j^2$
