COSC343: Artificial Intelligence

Lecture 8 : Non-linear regression and Optimisation

Lech Szymanski

Dept. of Computer Science, University of Otago

In today's lecture

- Error surface
- Iterative parameter search
- Non-linear regression and multiple minima
- Optimisation techniques:
  - Random walk
  - Steepest gradient
  - Simulated annealing

Recall: Least squares and linear regression

The least squares parameters that minimise $J = \frac{1}{2}\sum_i e_i^2$, where $e_i = y_i - \tilde{y}_i$, can be found by solving:

$$\frac{dJ}{d\mathbf{w}} = \sum_i \frac{de_i}{d\mathbf{w}} e_i = 0$$

Linear regression is consistent only when appropriate choice of base functions is made

For models that are linear in parameters, $y = f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_U(\mathbf{x}) \end{bmatrix}$, where

$\mathbf{w}^T = \begin{bmatrix} w_1 & \dots & w_U \end{bmatrix}$, there is a closed form solution:

Matrix inversion is computationally intensive

$$\mathbf{w} = \left(\mathbf{F}\mathbf{F}^T\right)^{-1} \mathbf{F}\tilde{\mathbf{y}}^T$$, where

This matrix is sometimes singular

$$\mathbf{F} = \begin{bmatrix} f_1(\mathbf{x}_1) & \dots & f_1(\mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ f_U(\mathbf{x}_1) & \dots & f_U(\mathbf{x}_N) \end{bmatrix} \text{ and } \tilde{\mathbf{y}} = \begin{bmatrix} \tilde{y}_1 & \dots & \tilde{y}_N \end{bmatrix}$$

Recall: Least squares and regression

The least squares parameters that minimise $J = \frac{1}{2}\sum_i e_i^2$, where $e_i = y_i - \tilde{y}_i$, can be found by solving:

$$\frac{dJ}{d\mathbf{w}} = \sum_i \frac{de_i}{d\mathbf{w}} e_i = 0$$

Can appropriate parameters be found without the closed form equation?

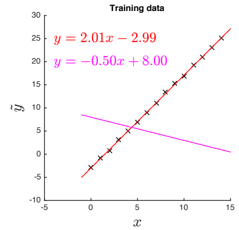
$$\cancel{\mathbf{w} = \left(\mathbf{F}\mathbf{F}^T\right)^{-1} \mathbf{F}\tilde{\mathbf{y}}^T}$$

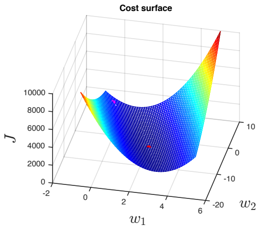
Space of possible solutions

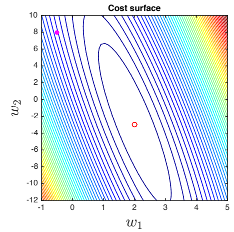
Let's think about the space of possible values that $\mathbf{w}$ can take, and the corresponding cost $J = \frac{1}{2}\sum_i e_i^2$ for some hypothesis $f(\mathbf{x}, \mathbf{w})$.

- The space of possible solutions is a U-dimensional **state-space** (where U is the number of parameters in the model);
- A **cost function** maps each point in the state space to a real number;
- The evaluations of all points in the state space can be visualised as a **state-space landscape** (in U+1 dimensions)

An example: fitting a line







$y_i = w_1 x_i + w_2$

$J = \frac{1}{2}\sum_i (y_i - \tilde{y}_i)^2$

$\mathbf{w} = (\mathbf{F}\mathbf{F}^T)^{-1}\mathbf{F}\tilde{\mathbf{y}}^T$

$\underbrace{w_1 = 2.01, w_2 = -2.99}_{\text{Closed form solution}}$

$\underbrace{\text{Epoch}0: w_1 = -0.50, w_2 = 8.00}_{\text{Initial guess}}$

Parameter search

1. Make an initial guess of the parameter values $\mathbf{w}_0$ (often *random*) and compute the cost

$$J_0 = \frac{1}{2}\sum_i (f(\mathbf{x}_i, \mathbf{w}_0) - \tilde{y}_i)^2$$

2. Update the parameters and compute the new cost

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\Delta\mathbf{w}_t$$

Learning rate parameter

Change in parameter values

$$J_t = \frac{1}{2}\sum_i (f(\mathbf{x}_i, \mathbf{w}_t) - \tilde{y}_i)^2$$

3. Go back to step 2

Random walk

Pick the weight update at random

$$\Delta\mathbf{w}_t = \left[\Delta w_{1t} \ldots \Delta w_{Ut}\right]^T, \text{ where}$$

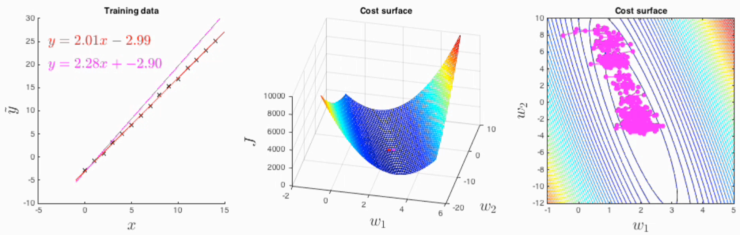
$$\Delta w_{jt} \sim \mathcal{N}(0, 1)$$

Parameter update is a random variable (in this case with a normal distribution)

keep the update $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\Delta\mathbf{w}_t$ if

$$J_{t+1} < J_t$$

An example: fitting a line with random walk



$y_i = w_1 x_i + w_2$

$J = \frac{1}{2}\sum_i (y_i - \tilde{y}_i)^2$

$\mathbf{w} = (\mathbf{F}\mathbf{F}^T)^{-1}\mathbf{F}\tilde{\mathbf{y}}^T$
$w_1 = 2.01, w_2 = -2.99$
$J = 0.18$

Random walk:
$\Delta\mathbf{w}_t = [\Delta w_{1t}\ \Delta w_{2t}]^T$
$\Delta w_{jt} \sim \mathcal{N}(0,1)$
$\mathbf{w}_{t+1} = \mathbf{w}_t + 0.50\Delta\mathbf{w}_t$

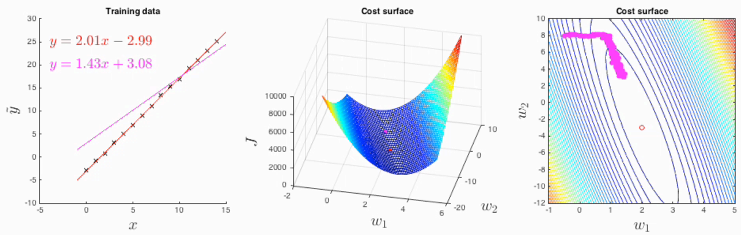
Epoch 500 : $w_1 = 2.28, w_2 = -2.90$ $J = 41.94$

Learning rate parameter

Learning parameter controls how big the *update steps* are when parameters are updated.

- Small α results in smaller steps: more ordered and direct path towards the minimum, but it takes longer to get there
- Large α results in bigger steps: faster convergence, but less direct path and more chance of overshooting the minimum

An example: random walk with smaller α



$y_i = w_1 x_i + w_2$

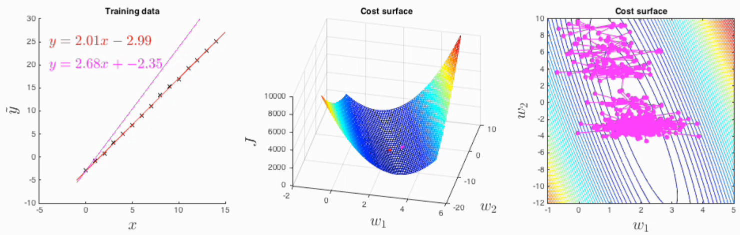
$J = \frac{1}{2}\sum_i (y_i - \tilde{y}_i)^2$

$\mathbf{w} = (\mathbf{F}\mathbf{F}^T)^{-1}\mathbf{F}\tilde{\mathbf{y}}^T$
$w_1 = 2.01, w_2 = -2.99$
$J = 0.18$

Random walk:
$\Delta\mathbf{w}_t = [\Delta w_{1t}\ \Delta w_{2t}]^T$
$\Delta w_{jt} \sim \mathcal{N}(0,1)$
$\mathbf{w}_{t+1} = \mathbf{w}_t + 0.200\Delta\mathbf{w}_t$

Epoch 500 : $w_1 = 1.43, w_2 = 3.08$ $J = 77.51$

An example: random walk with larger α



$y_i = w_1 x_i + w_2$

$J = \frac{1}{2}\sum_i (y_i - \tilde{y}_i)^2$

$\mathbf{w} = (\mathbf{F}\mathbf{F}^T)^{-1}\mathbf{F}\tilde{\mathbf{y}}^T$
$w_1 = 2.01, w_2 = -2.99$
$J = 0.18$

Random walk:
$\Delta\mathbf{w}_t = [\Delta w_{1t}\ \Delta w_{2t}]^T$
$\Delta w_{jt} \sim \mathcal{N}(0,1)$
$\mathbf{w}_{t+1} = \mathbf{w}_t + 0.800\Delta\mathbf{w}_t$

Epoch 500 : $w_1 = 2.68, w_2 = -2.35$ $J = 280.02$

Steepest gradient descent

The weight update is the negative gradient of the cost function with respect to the parameters

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \Delta \mathbf{w}_t$$

Gradient is the positive slope (going up) of the cost surface at w_t

and

$$\Delta \mathbf{w}_t = -\frac{dJ}{d\mathbf{w}} = -\begin{bmatrix} \frac{\partial J}{\partial w_{1t}} & \dots & \frac{\partial J}{\partial w_{Ut}} \end{bmatrix}^T$$

- Also referred to as "steepest gradient ascent" or "hill climbing" if the objective function is being maximised

An example: fitting a line with steepest gradient descent

$$\Delta \mathbf{w}_t = -\frac{dJ}{d\mathbf{w}} = -\begin{bmatrix} \frac{\partial J}{\partial w_{1t}} & \dots & \frac{\partial J}{\partial w_{Ut}} \end{bmatrix}^T$$

Since $J = \sum_i e_i^2$,

then $\frac{\partial J}{\partial w_j} = \sum_i \frac{\partial e_i}{\partial w_j} e_i$.

Since $e_i = y_i - \tilde{y}_i$,

then $\frac{\partial e_i}{\partial w_j} = \frac{\partial y_i}{\partial w_j}$.

Since $y_i = \sum_j w_j f_j(\mathbf{x}_i)$,

then $\frac{\partial y_i}{\partial w_j} = f_j(\mathbf{x}_i)$.

Given: $y_i = w_1 x_i + w_2$,

Output derivatives are:

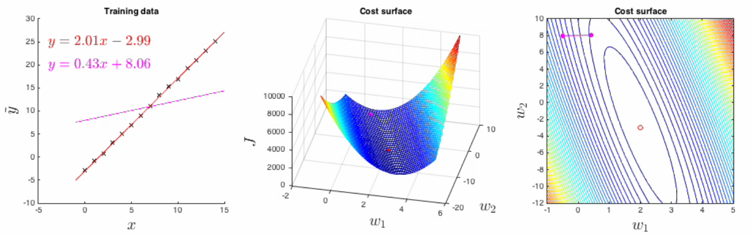
$$\frac{\partial y_i}{\partial w_1} = x_i \qquad \frac{\partial y_i}{\partial w_2} = 1$$

Cost derivatives are:

$$\frac{\partial J}{\partial w_1} = \sum_i x_i e_i \qquad \frac{\partial J}{\partial w_2} = \sum_i e_i$$

j - index over number of weights
i – index over number of training samples

An example: fitting a line with steepest gradient descent



$y_i = w_1 x_i + w_2$

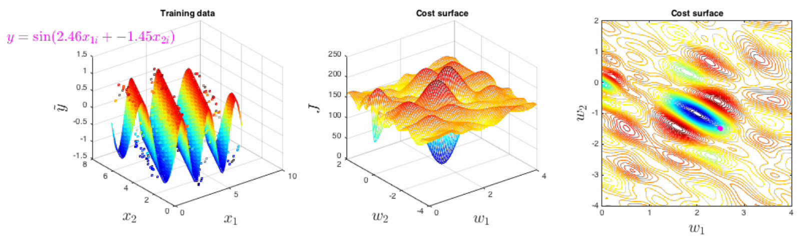
$J = \frac{1}{2}\sum_i (y_i - \tilde{y}_i)^2$

$\mathbf{w} = (\mathbf{F}\mathbf{F}^T)^{-1}\mathbf{F}\tilde{\mathbf{y}}^T$
$w_1 = 2.01, w_2 = -2.99$
$J = 0.18$

Steepest gradient descent:
$\Delta \mathbf{w}_t = [\Delta w_{1t}\ \Delta w_{2t}]^T$
$\Delta w_{1t} = -\sum_i x_i (y_i - \tilde{y}_i)$, $\Delta w_{2t} = -\sum_i (y_i - \tilde{y}_i)$
$\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{0.01}{N} \Delta \mathbf{w}_t$

Epoch 1 : $w_1 = 0.43, w_2 = 8.06$ $\quad J = 349.93$

An example: non-linear regression

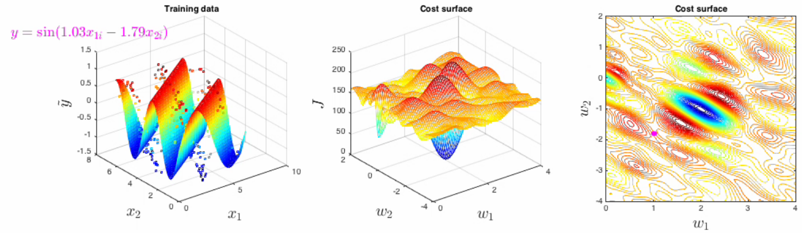


$y_i = \sin(w_1 x_{1i} + w_2 x_{2i})$ $\quad J = \frac{1}{2}\sum_i (y_i - \tilde{y}_i)^2$

Steepest gradient descent:
$\Delta \mathbf{w}_t = [\Delta w_{1t}\ \Delta w_{2t}]^T$
$\Delta w_{1t} = -\sum_i x_{1i} \cos(y_i)(y_i - \tilde{y}_i)$, $\Delta w_{2t} = -\sum_i x_{2i} \cos(y_i)(y_i - \tilde{y}_i)$
$\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{0.10}{N} \Delta \mathbf{w}_t$

Epoch1 : $w_1 = 2.46, w_2 = -1.45$ $\quad J = 78.00$

An example: non-linear regression



$y_i = \sin(w_1 x_{1i} + w_2 x_{2i})$ $\quad J = \frac{1}{2}\sum_i (y_i - \tilde{y}_i)^2$

Steepest gradient descent:

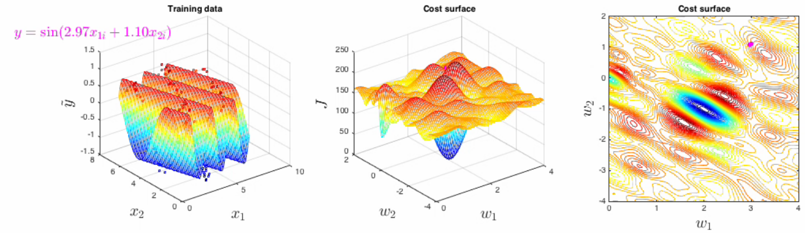
$\Delta \mathbf{w}_t = [\Delta w_{1t} \; \Delta w_{2t}]^T$

$\Delta w_{1t} = -\sum_i x_{1i} \cos(y_i)(y_i - \tilde{y}_i)$, $\Delta w_{2t} = -\sum_i x_{2i} \cos(y_i)(y_i - \tilde{y}_i)$

$\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{0.10}{N} \Delta \mathbf{w}_t$

$\text{Epoch1} : w_1 = 1.03, w_2 = -1.79$ $\quad J = 166.78$

An example: non-linear regression



$y_i = \sin(w_1 x_{1i} + w_2 x_{2i})$ $\quad J = \frac{1}{2}\sum_i (y_i - \tilde{y}_i)^2$

Steepest gradient descent:

$\Delta \mathbf{w}_t = [\Delta w_{1t} \; \Delta w_{2t}]^T$

$\Delta w_{1t} = -\sum_i x_{1i} \cos(y_i)(y_i - \tilde{y}_i)$, $\Delta w_{2t} = -\sum_i x_{2i} \cos(y_i)(y_i - \tilde{y}_i)$

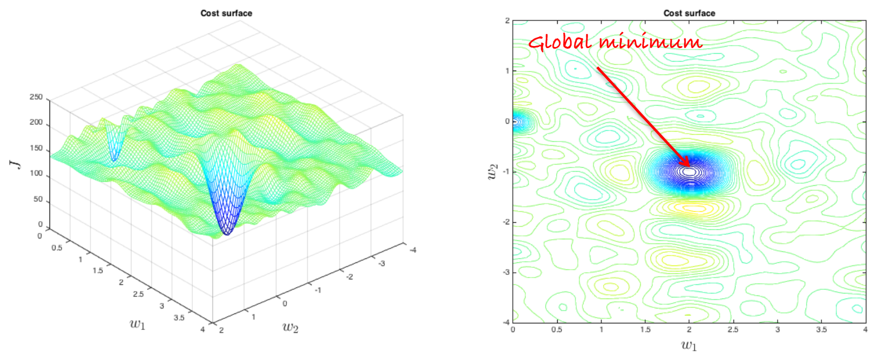
$\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{0.10}{N} \Delta \mathbf{w}_t$

$\text{Epoch1} : w_1 = 2.97, w_2 = 1.10$ $\quad J = 165.17$

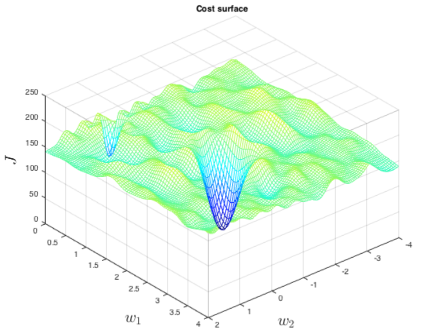
Local minima

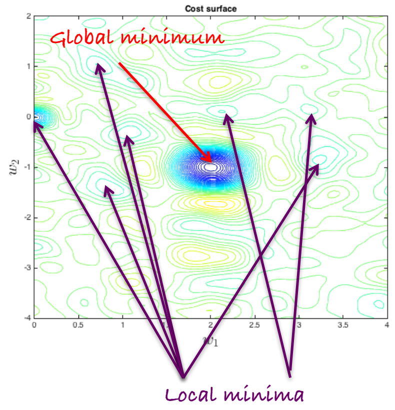
- Cost functions in non-linear optimisation are not necessarily convex
  - The cost surface may not correspond to a one giant valley, but a series of valleys separated by high cost ridges
- **Global minimum** – the state of the system that gives lowest possible cost
  - Best solution for the chosen model
- **Local minimum** – the minimum closest to the current state
  - May not be the best solution
  - Steepest gradient always points towards the local minimum, and as a result, the outcome of the training is highly dependent on the starting value of the parameters

Local minima



Local minima





Simulated annealing

Pick the weight update at random

$\Delta \mathbf{w}_t = \left[\Delta w_{1t} \ldots \Delta w_{Ut}\right]^T$, where $\Delta w_{jt} \sim \mathcal{N}(0,1)$

keep the update $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \Delta \mathbf{w}_t$ with probability

$$P(\mathbf{w}_{t+1}|\Delta \mathbf{w}_t) = \frac{1}{1+e^{-\frac{\Delta J}{T_t}}}$$, where

Reduction in cost

$$\Delta J = J_t - J_{t+1}$$

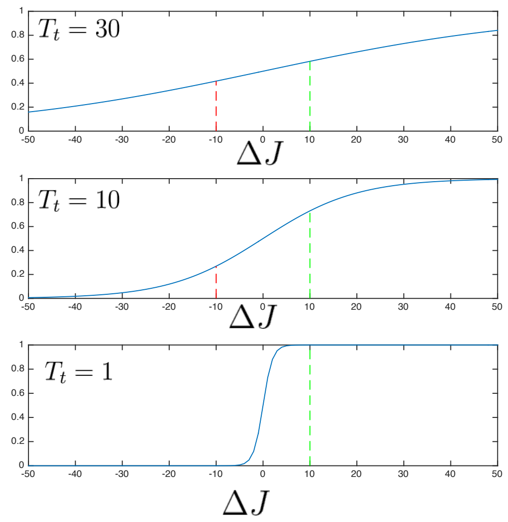
$$T_t = T_0 e^{-tT_c}$$

Temperature

Simulated annealing: probability of accepting the update

$$P(\mathbf{w}_{t+1}|\Delta \mathbf{w}_t) = \frac{1}{1+e^{-\frac{\Delta J}{T_t}}}$$

- High temperature increases the chance of accepting an update that results in higher cost
- Low temperature reduces the change of accepting an update that results in higher cost
- Start training with high temperature
  - More energy to jump out of the current minimum
- End training with low temperature
  - No energy to jump out of the current minimum

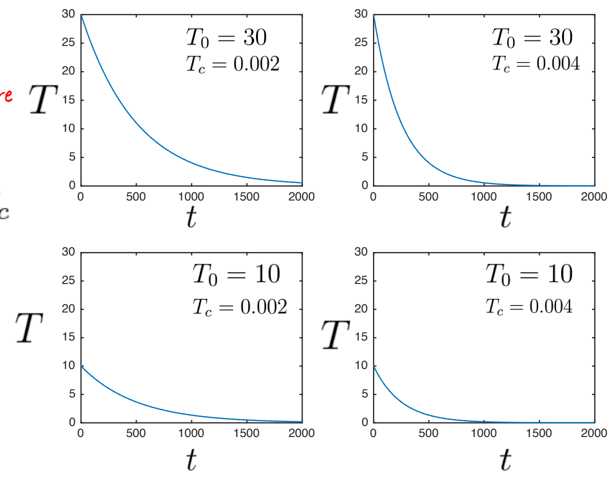


Simulated annealing: cooling schedule

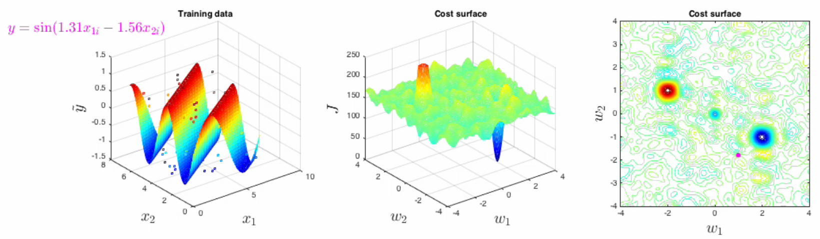
Starting temperature

$$T_t = T_0 e^{-tT_c}$$

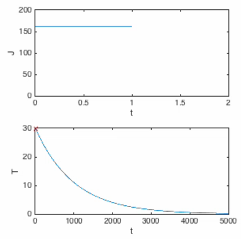
Rate of cooling



An example: non-linear regression with simulated annealing



$y_i = \sin(w_1 x_{1i} + w_2 x_{2i})$ $\quad J = \frac{1}{2}\sum_i (y_i - \tilde{y}_i)^2$



Simulated annealing:

$\Delta w_{jt} \sim \mathcal{N}(0,1)$

$\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{0.20}{N}\Delta \mathbf{w}_t$

$P(\mathbf{w}_{t+1}|\Delta \mathbf{w}_t) = \frac{1}{1+e^{-(J_t - J_{t+1})/T_t}}$, $T_t = 30e^{-t/1000}$

Epoch1 : $w_1 = 1.00, w_2 = -1.80$ $\quad J = 162.50$

Summary and reading

- Learning as parameter search over the cost surface
- Models that are non-linear in parameters tend to have multiple minima
- Random walk – slow, can’t guarantee where it goes
- Gradient descent – very fast, finds local minima
- Simulated annealing – theoretical promise to find global minimum (but slow and hard to get it to work in practice)

Reading for the lecture: AIMA Chapter 18 Section 4

Reading for next lecture: AIMA Chapter 18 Sections 7.1,7.2