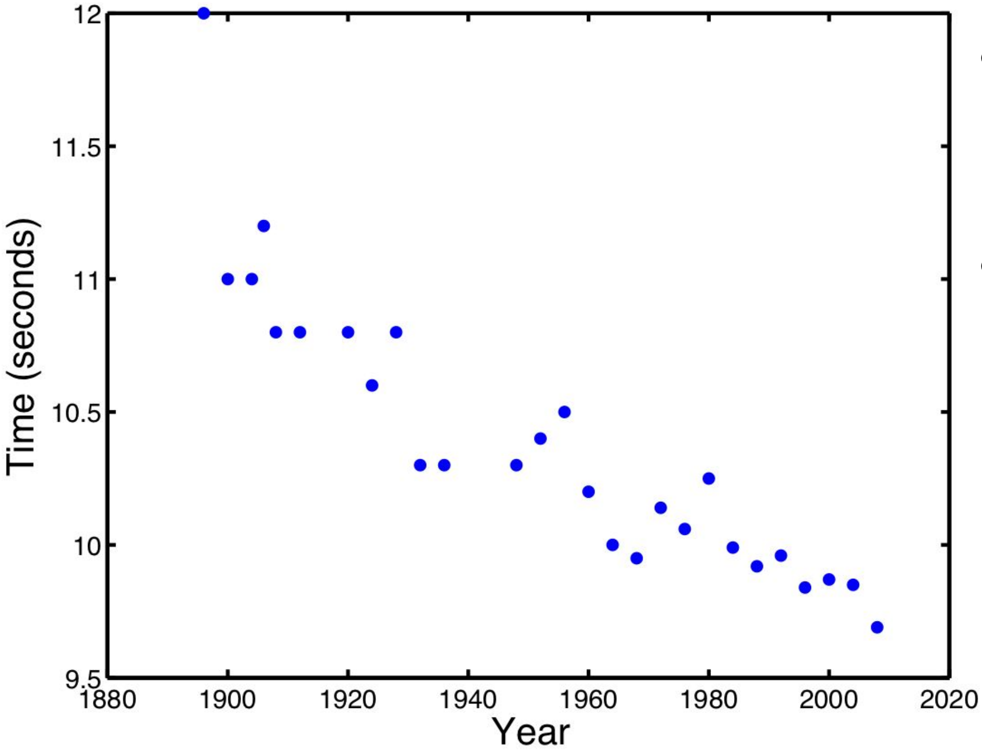
Machine Learning & Artificial Intelligence for Data Scientists: Regression (Part 1)

Ke Yuan
https://kyuanlab.org/
School of Computing Science

Some data and a problem



- Winning times for the men's Olympic 100m sprint, 1896-2008.
- In this lecture, we will use this data to predict the winning time in London 2012

Let's look at the data

In [13]:

```
import numpy as np
%matplotlib inline
import pylab as plt

data = np.loadtxt('olympic100m.txt', delimiter=',') # load olympic data
data
```

Out[13]:

```
array([[1896.  ,   12.  ],
       [1900.  ,   11.  ],
       [1904.  ,   11.  ],
       [1906.  ,   11.2 ],
       [1908.  ,   10.8 ],
       [1912.  ,   10.8 ],
       [1920.  ,   10.8 ],
       [1924.  ,   10.6 ],
       [1928.  ,   10.8 ],

       [1932.  ,   10.3 ],
       [1936.  ,   10.3 ],
       [1948.  ,   10.3 ],
       [1952.  ,   10.4 ],
       [1956.  ,   10.5 ],
       [1960.  ,   10.2 ],
       [1964.  ,   10.  ],
       [1968.  ,    9.95],
       [1972.  ,   10.14],
       [1976.  ,   10.06],
       [1980.  ,   10.25],
       [1984.  ,    9.99],
       [1988.  ,    9.92],
       [1992.  ,    9.96],
       [1996.  ,    9.84],
       [2000.  ,    9.87],
       [2004.  ,    9.85],
       [2008.  ,    9.69]])
```

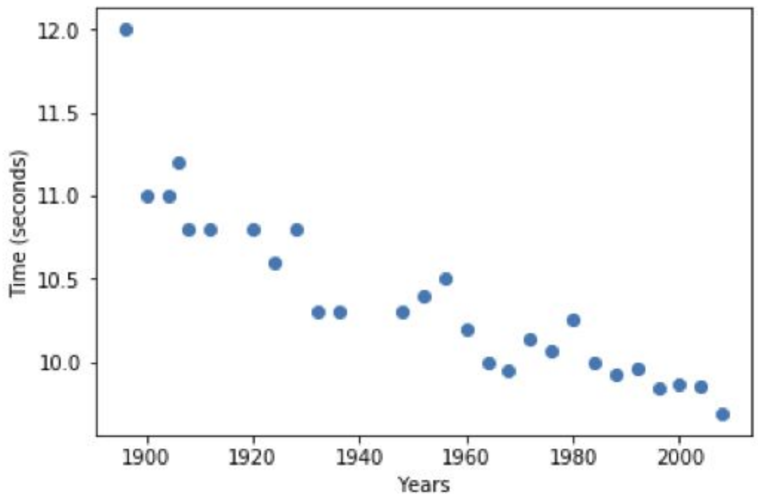
In [15]:

```
x =  data[:,0] # name years as x
t =  data[:,1] # name time as t

plt.scatter(x,t) # draw a scatter plot
plt.xlabel('Years') # always label x&y-axis
plt.ylabel('Time (seconds)') # always label x&y-axis
```

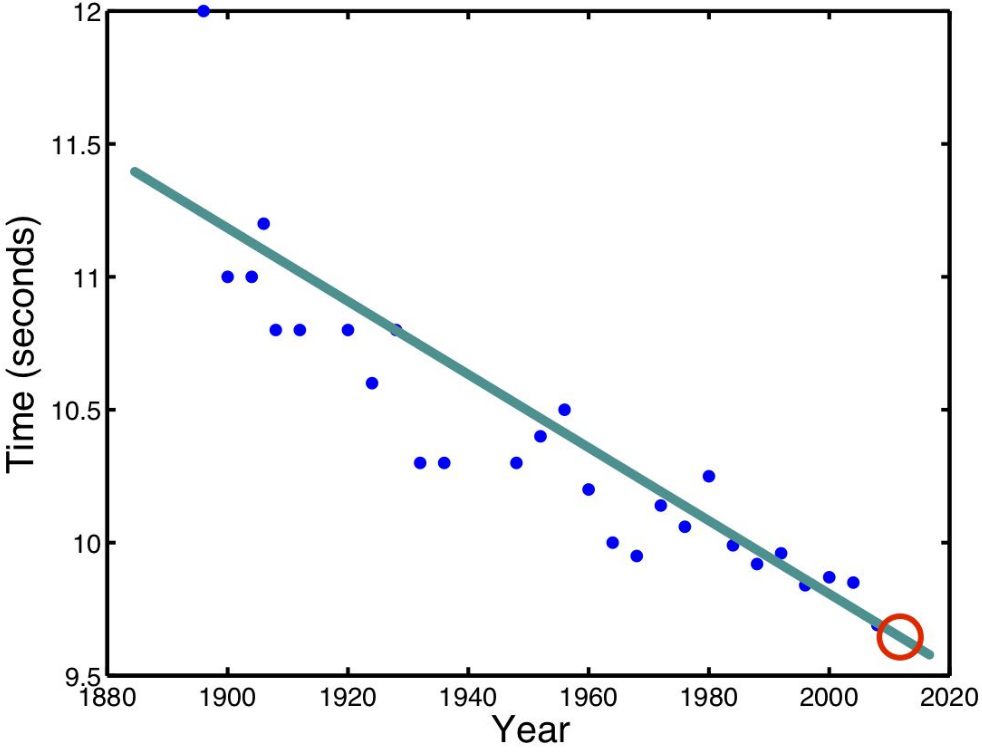
Out[15]:

```
Text(0, 0.5, 'Time (seconds)')
```



Our first scatter plot

Draw a line through it!



Overview: Simple Linear Regression

- Introduce the idea of building models.
- Talk about assumptions.
- Use a linear model.
- What constitutes a good model?
- Find the best linear model.
- Use it to predict the winning time in 2012.

Assumptions

---

1. That there exists a relationship between Olympic year and winning time.
2. That this relationship is linear (i.e. a straight line).
3. That this relationship will continue into the future.

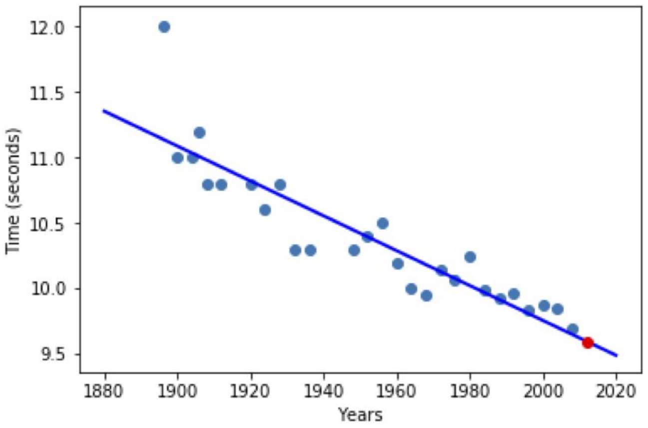
In [24]:
```
# fit a straight line
w_0 = 36.4164559025
w_1 = -0.013330885711

x_test = np.linspace(1880,2020, 100) # generate new x to plot the fitted line. Not
e better not to use the original x !
f_test = w_0 + w_1 * x_test
plt.plot(x_test,f_test,'b-',linewidth=2) # plot the fitted data
plt.plot(2012, w_0 + w_1 * 2012, 'ro')

plt.scatter(x,t) # draw a scatter plot
plt.xlabel('Years') # always label x&y-axis
plt.ylabel('Time (seconds)') # always label x&y-axis
```

Out[24]:
```
Text(0, 0.5, 'Time (seconds)')
```

Draw a line through it!



Let's reflect on the task

Attributes and targets

Typically in Supervised Machine Learning, we have a set of attributes and corresponding targets:

- **Attributes:** Olympic year.
- **Targets:** Winning time.

Key definitions

Variables

Mathematically, each is described by a variable:

- Olympic year: x.
- Winning time: t.

Key definitions

Model

Our goal is to create a model.

- This is a function that can relate x to t.

$$t = f(x)$$

- Hence, we can work out t when $x = 2012$.

Key definitions

Data

We're going to create the model from data:

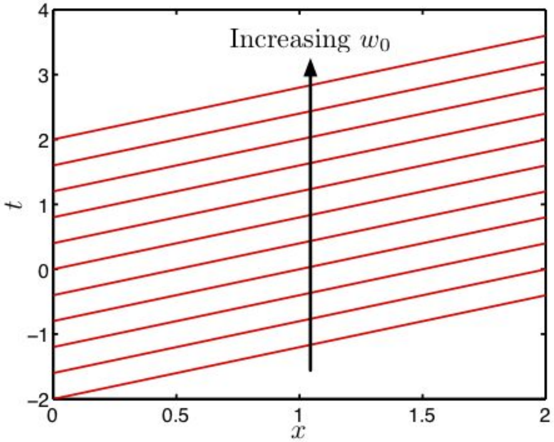
- N attribute-response pairs, (x_n, t_n)
- e.g. $(1896, 12s), (1900, 11s), \ldots, (2008, 9.69s)$
- $x_1 = 1896,\ t_1 = 12$, etc

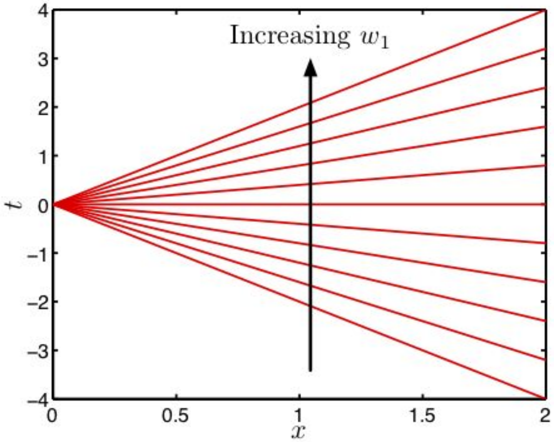
Often called **training** data

What is a model?

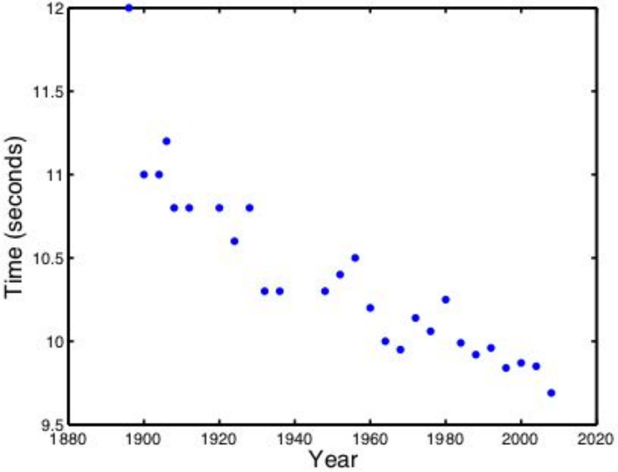
$$t = f(x) = w_0 + w_1 x = f(x; w_0, w_1)$$

- w_0 and w_1 are *parameters* of the model.
- They determine the properties of the line.

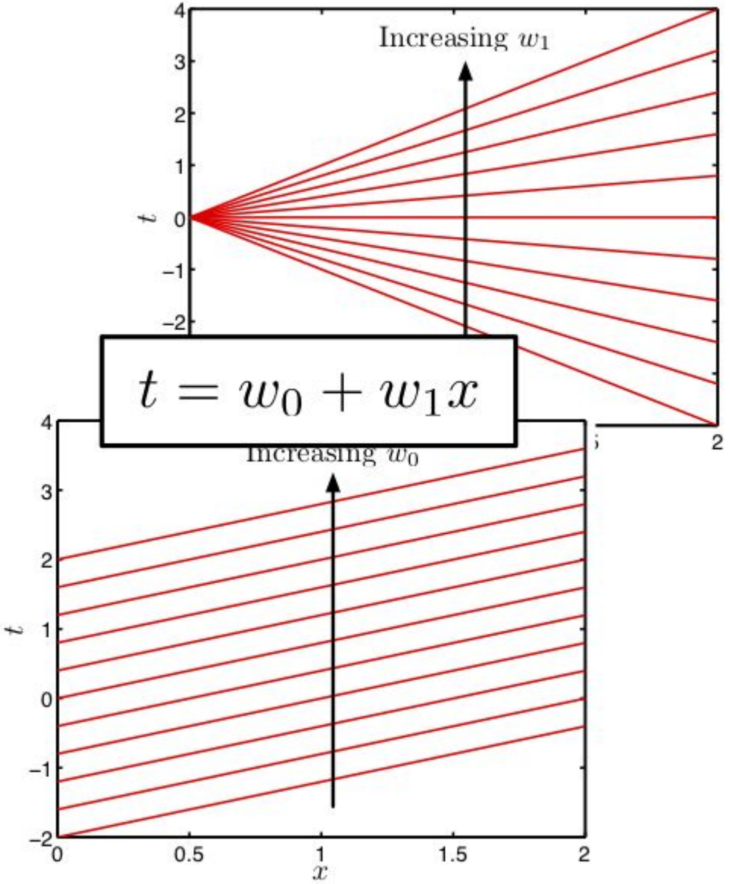




We have data and a family of models:



?



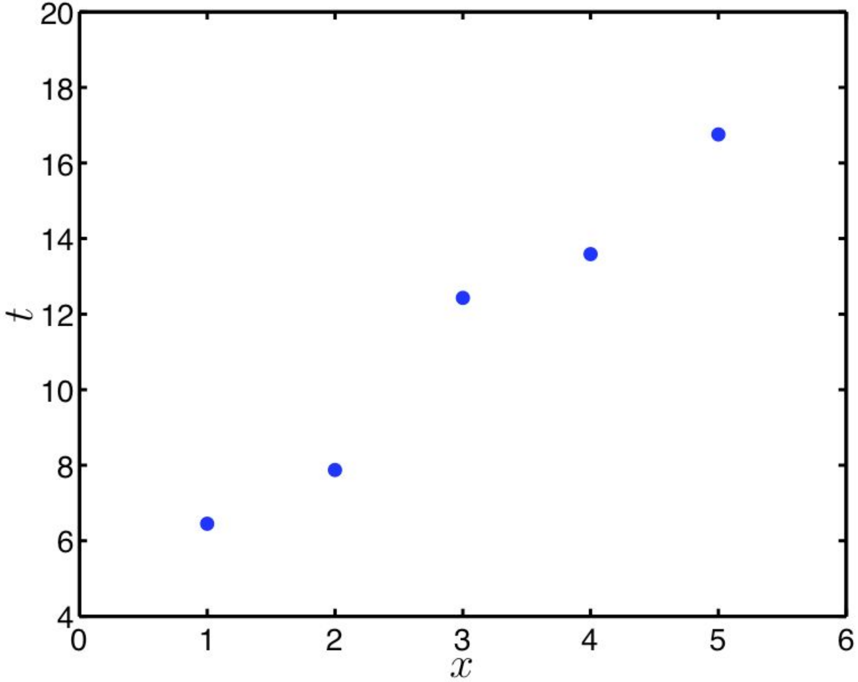
$$t = w_0 + w_1 x$$

What is Learning?

How good is a particular w_0, w_1?

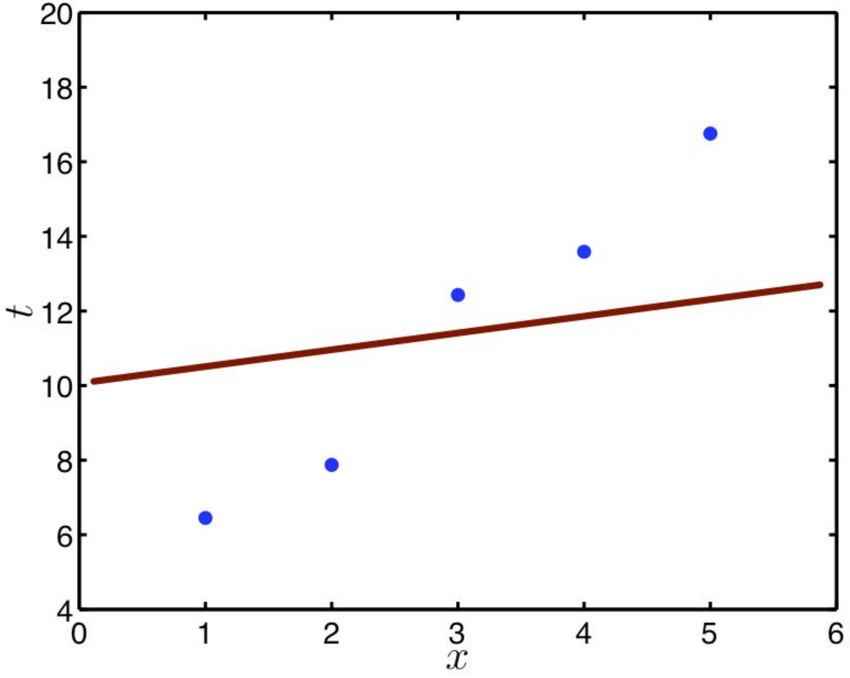
- How good is a particular line (w_0, w_1)?
- We need to be able to provide a numerical value of goodness for any w_0, w_1.
  - How good is $w_0 = 5, w_1 = 0.1$?
  - Is $w_0 = 5, w_1 = -0.1$ better or worse?
- Once we can answer these questions, we can search for the best w_0, w_1 pair.

Loss



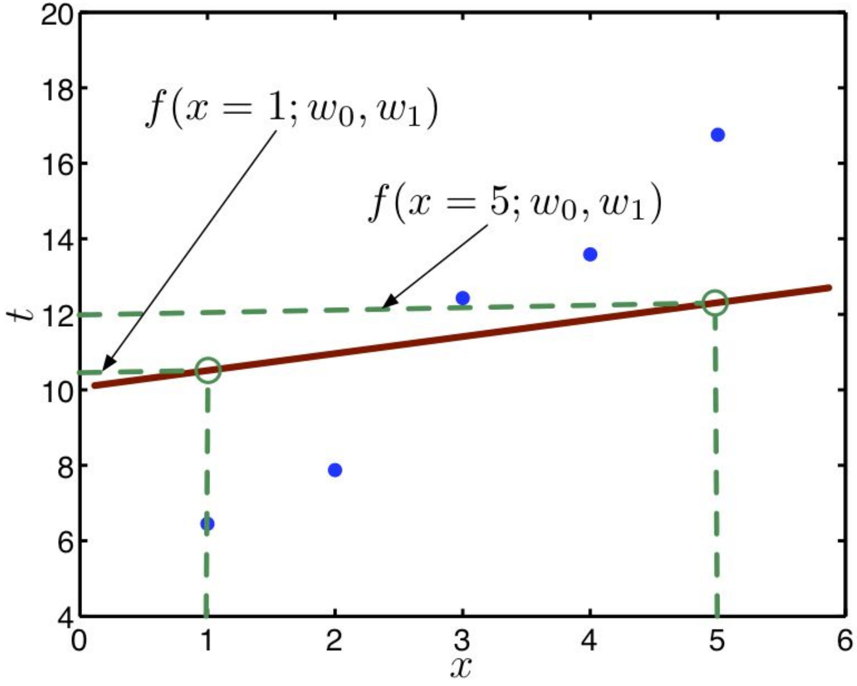
Some different data.

Loss



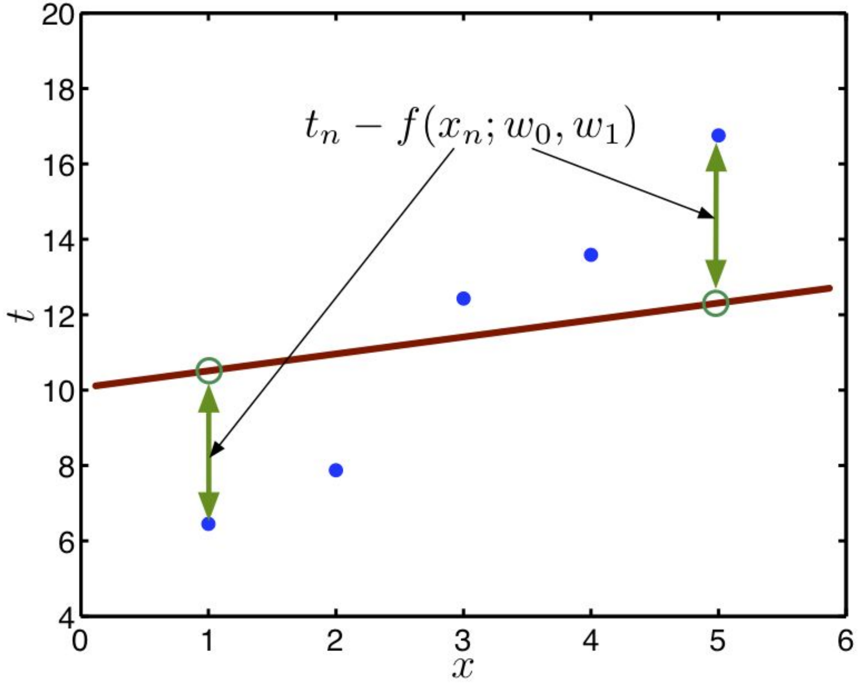
Given w_0 and w_1 you can draw a line.

Loss



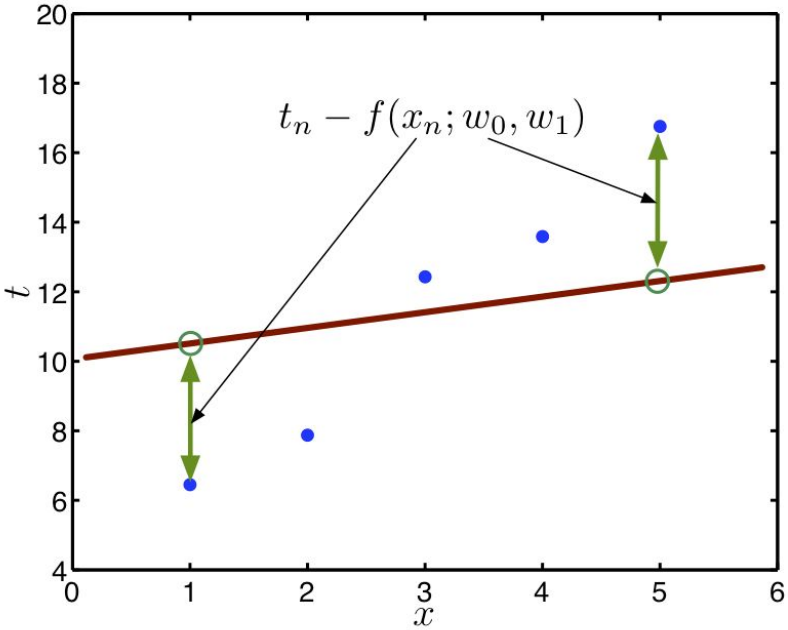
This means that we can compute $f(x_n; w_0, w_1)$ for each x_n.

Loss



$f(x_n; w_0, w_1)$ can be compared with the truth, t_n.

Squared Loss



$f(x_n; w_0, w_1)$ can be compared with the truth, t_n.
$(t_n - f(x_n; w_0, w_1))^2$ tells us how *badly* we model (x_n, t_n).

Squared Loss

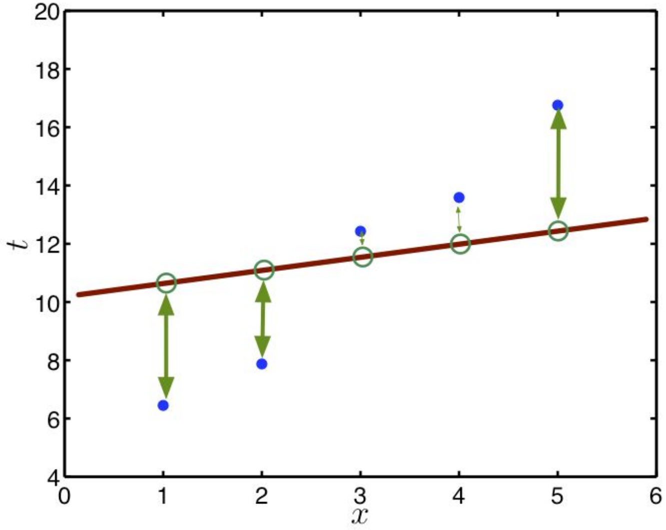
–

- The *Squared loss* of training point n is defined as:

$$\mathcal{L}_n = (t_n - f(x_n; w_0; w_1))^2$$

- It is the squared difference between the true response (winning time), t_n when the input is x_n and the response predicted by the model, $f(x_n; w_0, w_1) = w_0 + w_1 x_n$.
- The lower $\mathcal{L}_n$, the closer the line at x_n passes to t_n.

Averaged Squared Loss



Average the loss at each training point to give single figure for all data:

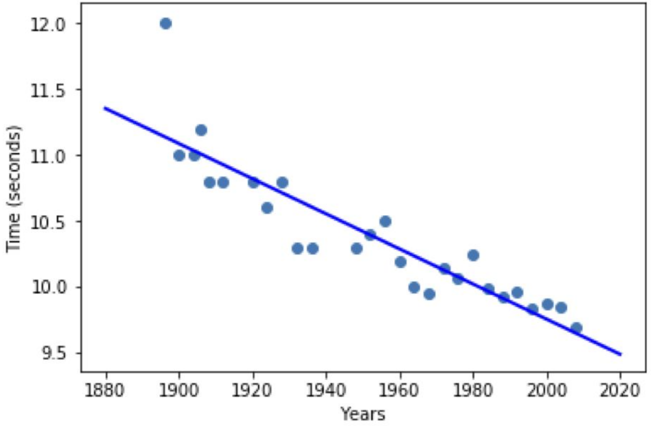
$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}(t_n - f(x_n; w_0, w_1))^2$$

In [4]:
```
# fit a straight line
w_0 = 36.4164559025
w_1 = -0.013330885711

x_test = np.linspace(1880,2020, 100) # generate new x to plot the fitted line. Not
e better not to use the original x !
f_test = w_0 + w_1 * x_test
plt.plot(x_test,f_test,'b-',linewidth=2) # plot the fitted data

plt.scatter(x,t) # draw a scatter plot
plt.xlabel('Years') # always label x&y-axis
plt.ylabel('Time (seconds)') # always label x&y-axis
```

Out[4]:
```
Text(0, 0.5, 'Time (seconds)')
```



Compare two different models

Model 1 :
w_0 = 36.4164559025
w_1 = -0.013330885711

In [5]:
```
sum((t-36.4164559025 - (-0.013330885711)*x)**2)/t.shape[0]
```

Out[5]:
```
0.05030711047565771
```

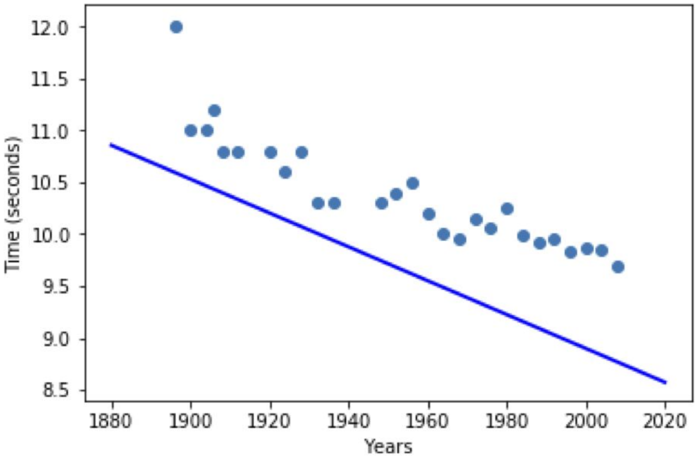
```
In [6]:
# fit a straight line
w_0 = 41.5
w_1 = -0.0163

x_test = np.linspace(1880,2020, 100) # generate new x to plot the fitted line. Not
e better not to use the original x !
f_test = w_0 + w_1 * x_test
plt.plot(x_test,f_test,'b-',linewidth=2) # plot the fitted data

plt.scatter(x,t) # draw a scatter plot
plt.xlabel('Years') # always label x&y-axis
plt.ylabel('Time (seconds)') # always label x&y-axis
```

```
Out[6]: Text(0, 0.5, 'Time (seconds)')
```

Compare two different models



Model 2 :
w_0 = 41.5
w_1 = -0.0163

```
In [7]: sum((t-41.5- (- 0.013330885711)*x)**2)/t.shape[0]

Out[7]: 25.892727700891623
```

Model fitting

```
In [52]: from sklearn.linear_model import LinearRegression # import

         x = x[:,None] # 27 x 1 array
         t = t[:,None] # 27 x 1 array

         reg = LinearRegression().fit(x, t)
```

```
In [53]: [reg.intercept_, reg.coef_]

Out[53]: [array([36.4164559]), array([[-0.01333089]])]
```

```
In [54]: reg.predict(np.array([[2012]]))

Out[54]: array([[9.59471385]])
```

Summary

- Introduced some ideas about modelling.
- Found some data.
- Derived a way of saying how good a model is.
- Found an expression for the best model.
- Used this to fit a model to the Olympic data.
- Made a prediction for the winning time in 2012.

Assumptions again

---

1. That there exists a relationship between Olympic year and winning time.
2. That this relationship is linear (i.e. a straight line).
3. That this relationship will continue into the future.

Assumptions are wrong

- Relationship is clearly not perfectly linear.
- Winning time cannot decrease forever - it must be positive.
- It can't increase forever into the past.

The model is 'wrong' but it might still be useful. How useful depends on the questions we wish to answer.