Lecture 2 – Linear Regression

Bulat Ibragimov

bulat@di.ku.dk

Department of Computer Science
University of Copenhagen

UNIVERSITY OF COPENHAGEN

Outline

1. Recap + Proof
2. Multivariate Case
3. Implementation
4. Summary & Outlook

Computing the optimal parameters

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

- We would like to find the two coefficients w_0 and w_1 that minimize the above objective!

- We have a function with two variables w_0 and w_1 and are searching for vector $\mathbf{w} = [w_0, w_1]^T$ corresponding to a minimum w.r.t. $\mathcal{L}$. Thus, the gradient of $\mathcal{L}$ must vanish at $\mathbf{w}$ (necessary condition!):

$$\nabla\mathcal{L}(w_0, w_1) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_0} \\ \frac{\partial \mathcal{L}}{\partial w_1} \end{bmatrix} \stackrel{!}{=} \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- Task: Compute both partial derivatives!

Proof 1

Let's say $f(x)$ is differentiable at point z and reaches at z a local minimum/maximum. How do we know that $f'(z) = 0$:

$$f'(z) = \lim_{\eta \to 0} \frac{f(z+\eta) - f(z)}{\eta} = 0$$

Suppose that $f'(z) > 0$.

Using the behavior of function near limit theory, we know that:

$$\exists\, \xi > 0\colon\ \forall y \in [z-\xi; z+\xi] \quad f'(y) > 0$$

Proof 1

Let's select a random $z_1 \in [z - \xi; z]$, so:

$$z - z_1 > 0$$

We can therefore conclude that:

$$f(z) > f(z_1), \qquad because \quad \frac{f(z) - f(z_1)}{z - z_1} > 0$$

So $f(z)$ is not a local minimum

Proof 1

Suppose that $f'(z) < 0$:

Using the behavior of function near limit theory, we know that:

$$\exists\, \xi > 0\colon\ \forall y \in [z - \xi; z + \xi] \quad f'(y) < 0$$

Let's select a random $z_1 \in [z; z + \xi]$, so:

$$z_1 - z > 0$$

We can therefore conclude that:

$$f(z) > f(z_1), \qquad \textit{because}\,\frac{f(z_1) - f(z)}{z_1 - z} < 0$$

So $f(z)$ is not a local minimum. The similar calculations can be performed for local maximum scenario.

Second derivative test

Optimizing loss function:

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

If the derivatives of L at point $\overline{w}_0$ and $\overline{w}_1$ equals zero, that means that function L reaches:

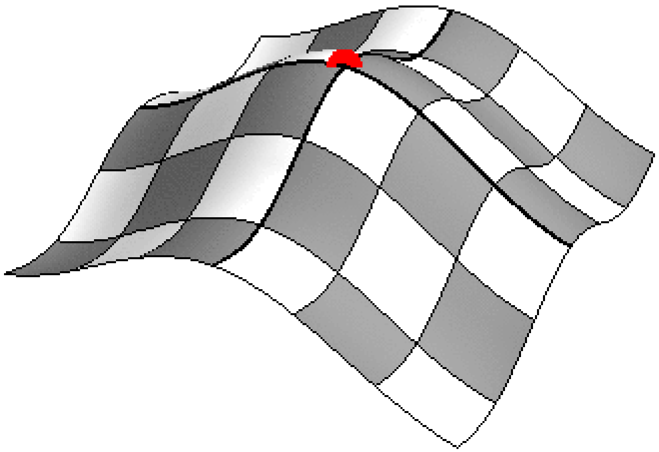
- local maximum/minimum against w_0 at point $\overline{w}_0$
- local maximum/minimum against w_1 at point $\overline{w}_1$

There are four options possible:

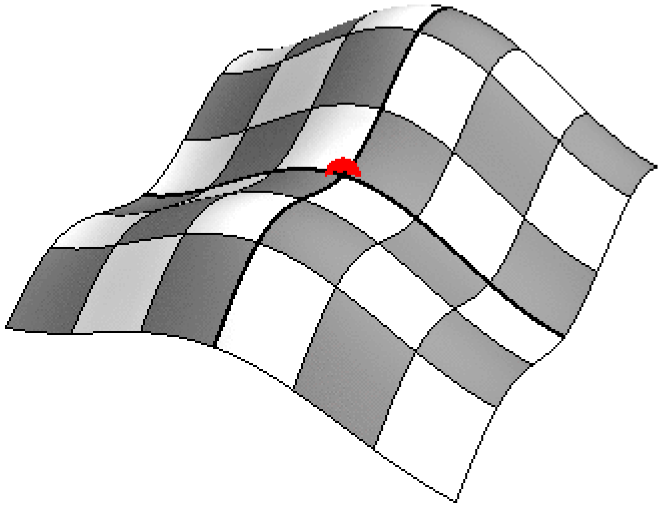
- Local maximum for w_0 and local maximum for w_1
- Local maximum for w_0 and local minimum for w_1
- Local minimum for w_0 and local minimum for w_1
- Local minimum for w_0 and local maximum for w_1

Second derivative test

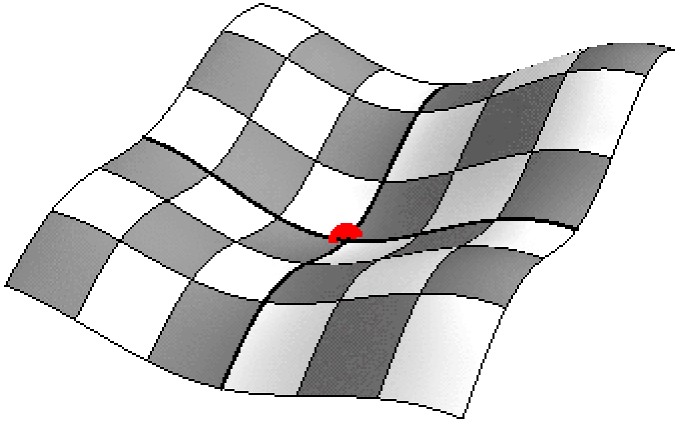
There are actually three different options:



Local maximum



Saddle point



Local minimum

Second derivative test

Hessian matrix of partial derivatives:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_0} & \frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_1} \\ \frac{\partial^2 \mathcal{L}}{\partial w_1 \partial w_0} & \frac{\partial^2 \mathcal{L}}{\partial w_1 \partial w_1} \end{bmatrix}$$

If derivative of L is zero at $\overline{w}_0, \overline{w}_1$, point $\overline{w}_0, \overline{w}_1$ is:

- $\det(\mathbf{H}(\overline{w}_0, \overline{w}_1)) > 0$, and $\frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_0} > 0$, then point $\overline{w}_0, \overline{w}_1$ is a local minimum
- $\det(\mathbf{H}(\overline{w}_0, \overline{w}_1)) > 0$, and $\frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_0} < 0$, then point $\overline{w}_0, \overline{w}_1$ is a local maximum
- $\det(\mathbf{H}(\overline{w}_0, \overline{w}_1)) < 0$

Proof 2

Let's prove that the point $\overline{w}_0, \overline{w}_1$, where the derivate is zero, is the point of minimum:

- $\det(\mathbf{H}(\overline{w}_0, \overline{w}_1)) > 0$, and $\dfrac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_0} > 0$, then point $\overline{w}_0, \overline{w}_1$ is a local minimum

- The partial derivatives are given by:

$$\frac{\partial \mathcal{L}}{\partial w_0} = 2w_0 + 2w_1 \frac{1}{N}\left(\sum_{n=1}^{N} x_n\right) - \frac{2}{N}\left(\sum_{n=1}^{N} t_n\right)$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = 2w_1 \frac{1}{N}\left(\sum_{n=1}^{N} {x_n}^2\right) + \frac{2}{N}\left(\sum_{n=1}^{N} x_n(w_0 - t_n)\right)$$

- Task: Derive the Hessian matrix!

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_0} & \frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_1} \\ \frac{\partial^2 \mathcal{L}}{\partial w_1 \partial w_0} & \frac{\partial^2 \mathcal{L}}{\partial w_1 \partial w_1} \end{bmatrix}$$

Proof 2

The Hessian matrix is:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_0} & \frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_1} \\ \frac{\partial^2 \mathcal{L}}{\partial w_1 \partial w_0} & \frac{\partial^2 \mathcal{L}}{\partial w_1 \partial w_1} \end{bmatrix} = \begin{bmatrix} 2 & \frac{2}{N}\left(\sum_{n=1}^{N} x_n\right) \\ \frac{2}{N}\left(\sum_{n=1}^{N} x_n\right) & \frac{2}{N}\left(\sum_{n=1}^{N} {x_n}^2\right) \end{bmatrix}$$

The $\det(\mathbf{H}) =$

$$\frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_0} \frac{\partial^2 \mathcal{L}}{\partial w_1 \partial w_1} - \left(\frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_1}\right)^2$$

$$= \quad 4 \cdot \left(\frac{1}{N}\sum_{n=1}^{N} {x_n}^2\right) - 4\left(\frac{1}{N}\sum_{n=1}^{N} x_n\right)^2$$

Proof 2

The $\det(\mathbf{H})$

$$= \quad 4 \cdot \left(\frac{1}{N} \sum_{n=1}^{N} {x_n}^2 \right) - 4 \left(\frac{1}{N} \sum_{n=1}^{N} x_n \right)^2$$

$$= \quad 4 \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})^2$$

Try to derive this step

If we assume that not all x_n are the same:

$$= \quad 4 \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})^2 > 0$$

Proof 2

The $\det(\mathbf{H}) > 0$

The $\frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_0} > 0$

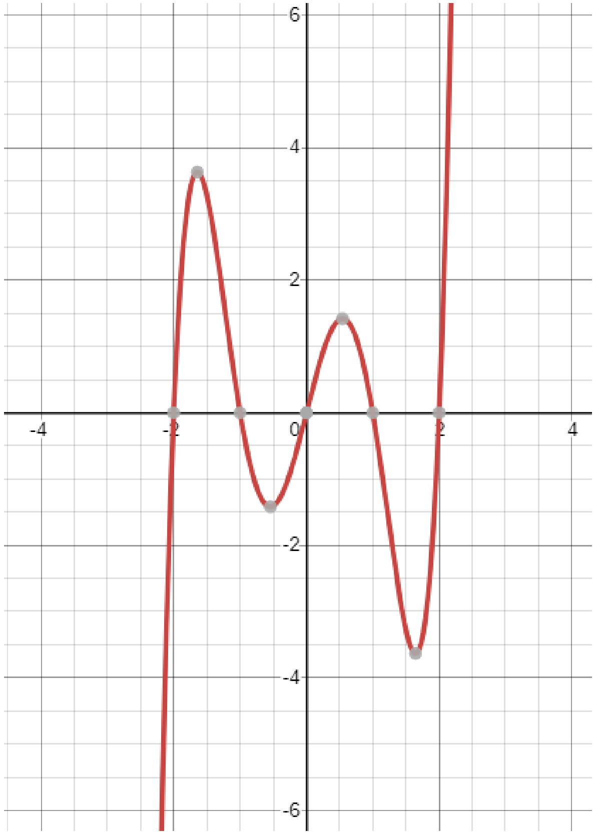
$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_0} & \frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_1} \\ \frac{\partial^2 \mathcal{L}}{\partial w_1 \partial w_0} & \frac{\partial^2 \mathcal{L}}{\partial w_1 \partial w_1} \end{bmatrix} = \begin{bmatrix} 2 & \frac{2}{N}\left(\sum_{n=1}^{N} x_n\right) \\ \frac{2}{N}\left(\sum_{n=1}^{N} x_n\right) & \frac{2}{N}\left(\sum_{n=1}^{N} {x_n}^2\right) \end{bmatrix}$$

According to the second derivative test, the point $\overline{w}_0, \overline{w}_1$ is a local minimum

Global minimum for linear regression

The point $\overline{w}_0, \overline{w}_1$ is a local minimum, but is it also a global minimum

Let's have function $f(x) = x^5 - 5x^3 + 4x$



Its derivative equals $5x^4 - 15x^2 + 4$:

- $x_1 = -1.644$
- $x_2 = -0.543$
- $x_3 = 0.543$
- $x_4 = 1.644$

x_2 and x_4 are local minima
but none of them is the global minimum

Global minimum for linear regression

We want to solve this system of equations

$$\frac{\partial \mathcal{L}}{\partial w_0} = 2w_0 + 2w_1 \frac{1}{N}\left(\sum_{n=1}^{N} x_n\right) - \frac{2}{N}\left(\sum_{n=1}^{N} t_n\right) = 0$$

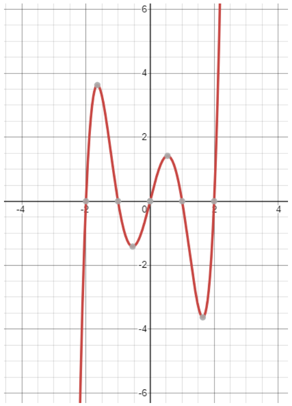
$$\frac{\partial \mathcal{L}}{\partial w_1} = 2w_1 \frac{1}{N}\left(\sum_{n=1}^{N} {x_n}^2\right) + \frac{2}{N}\left(\sum_{n=1}^{N} x_n(w_0 - t_n)\right) = 0$$

This is a system of two linear equations with two variables w_0 and w_1, so it has either no, or one, or infinite number of solutions

How many solutions does it actually have?

Global minimum for linear regression

Could there be no global minimum, i.e. function L goes to $-\infty$?



No! This is impossible, because L is the mean squared error, and the error cannot go negative

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

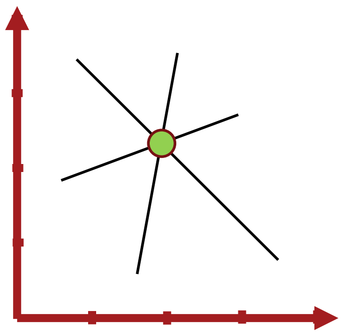
Could there be no solutions for the system $\begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_0} \\ \frac{\partial \mathcal{L}}{\partial w_1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

No! Because there exist a point where the mean squared error is lowest, and function L is continues

Global minimum for linear regression

Could there be infinite number of solutions?

Yes! Let's say we have only one data point.
Many lines can pass through it with the error of 0.



Any of these lines will be also globally optimal

Obviously we can have a situation with one solution, and it will be the global minimum considering $-\infty$ is impossible.

Linear Regression Using Matrix Notation

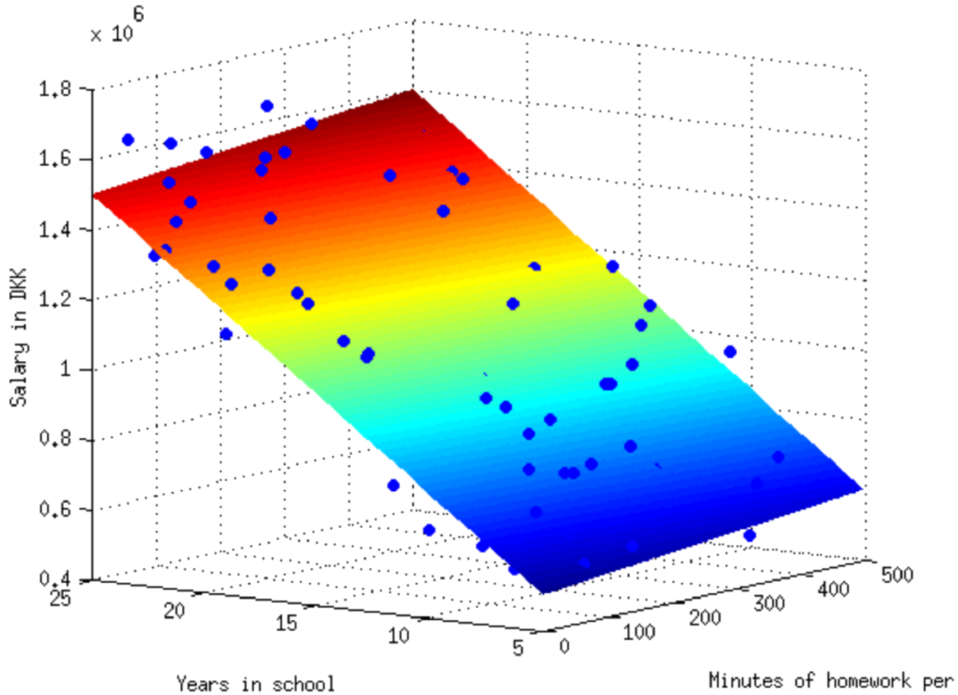
- We have: $f(x; w_0, w_1) = f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{x}^T\boldsymbol{w}$ with $\boldsymbol{x} = [1, x]^T$ and $\boldsymbol{w} = [w_0, w_1]^T$
- Let's "augment" all data points $x_1, x_2, \ldots, x_N$. This yields an augmented data matrix $\boldsymbol{X} \in \mathbb{R}^{N\times 2}$ and an associated target vector $\boldsymbol{t} \in \mathbb{R}^N$:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_N \end{bmatrix} \quad \text{and} \quad \boldsymbol{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$

- Then, we can write the overall loss as:

$$\mathcal{L}(w_0, w_1) = \frac{1}{N}\sum_{n=1}^{N}((w_0 + x_n w_1) - t_n)^2 = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{t})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{t})$$

Multivariate Linear Regression



General Form

- Given: Pairs of the form $(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N) \in \mathbb{R}^D \times \mathbb{R}$.
- Goal: Linear model $f(\boldsymbol{z}; \boldsymbol{w}) = w_0 + w_1 z_1 + w_2 z_2 + \ldots + w_D z_D$

Multivariate Linear Regression

- Given: Pairs of the form $(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N) \in \mathbb{R}^D \times \mathbb{R}$.
- Let's "augment" all data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$. This yields an augmented data matrix $\boldsymbol{X} \in \mathbb{R}^{N\times(D+1)}$ and an associated target vector $\boldsymbol{t} \in \mathbb{R}^N$:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \ldots & x_{1,D} \\ 1 & x_{2,1} & x_{2,2} & \ldots & x_{2,D} \\ \vdots & & & & \\ 1 & x_{N,1} & x_{N,2} & \ldots & x_{N,D} \end{bmatrix} \quad \text{and} \quad \boldsymbol{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$

- As before, we can write the overall loss in the following form:

Overall Loss

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(f(\mathbf{x}_n;\mathbf{w}) - t_n\right)^2 = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{t})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{t})$$

Simplifying the Objective

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(f(x_n;\boldsymbol{w}) - t_n)^2 = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{t})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{t})$$

$$\begin{aligned}
\mathcal{L}(\boldsymbol{w}) &= \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{t})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{t}) \\
&= \frac{1}{N}((\boldsymbol{X}\boldsymbol{w})^T - \boldsymbol{t}^T)(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{t}) \\
&= \frac{1}{N}(\boldsymbol{X}\boldsymbol{w})^T\boldsymbol{X}\boldsymbol{w} - \frac{1}{N}\boldsymbol{t}^T\boldsymbol{X}\boldsymbol{w} - \frac{1}{N}(\boldsymbol{X}\boldsymbol{w})^T\boldsymbol{t} + \frac{1}{N}\boldsymbol{t}^T\boldsymbol{t} \\
&= \frac{1}{N}\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \frac{2}{N}\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{t} + \frac{1}{N}\boldsymbol{t}^T\boldsymbol{t}
\end{aligned}$$

Gradient and Stationary Point

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - \frac{2}{N}\mathbf{w}^T\mathbf{X}^T\mathbf{t} + \frac{1}{N}\mathbf{t}^T\mathbf{t}$$

Toolbox (Table 1.4 in Rogers & Girolami)

1. $f(\mathbf{w}) = \mathbf{w}^T\mathbf{x} \Rightarrow \nabla f(\mathbf{w}) = \mathbf{x}$
2. $f(\mathbf{w}) = \mathbf{x}^T\mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = \mathbf{x}$
3. $f(\mathbf{w}) = \mathbf{w}^T\mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = 2\mathbf{w}$
4. $f(\mathbf{w}) = \mathbf{w}^T\mathbf{C}\mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = 2\mathbf{C}\mathbf{w}$ (if C is symmetric)

Task: Derive the gradient for $\mathcal{L}(\mathbf{w})$

Gradient and Stationary Point

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \frac{2}{N}\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{t} + \frac{1}{N}\boldsymbol{t}^T\boldsymbol{t}$$

Toolbox (Table 1.4 in Rogers & Girolami)

1. $f(\mathbf{w}) = \boldsymbol{w}^T\boldsymbol{x} \Rightarrow \nabla f(\boldsymbol{w}) = \boldsymbol{x}$
2. $f(\mathbf{w}) = \boldsymbol{x}^T\boldsymbol{w} \Rightarrow \nabla f(\boldsymbol{w}) = \boldsymbol{x}$
3. $f(\mathbf{w}) = \boldsymbol{w}^T\boldsymbol{w} \Rightarrow \nabla f(\boldsymbol{w}) = 2\boldsymbol{w}$
4. $f(\mathbf{w}) = \boldsymbol{w}^T\boldsymbol{C}\boldsymbol{w} \Rightarrow \nabla f(\boldsymbol{w}) = 2\boldsymbol{C}\boldsymbol{w}$ (if C is symmetric)

The gradient is given by $\nabla\mathcal{L}(\mathbf{w}) = \frac{2}{N}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \frac{2}{N}\boldsymbol{X}^T\boldsymbol{t}$. Therefore:

$$\begin{aligned} & \nabla\mathcal{L}(\mathbf{w}) && = \mathbf{0} \\ \Leftrightarrow\quad & \tfrac{2}{N}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \tfrac{2}{N}\boldsymbol{X}^T\boldsymbol{t} && = \mathbf{0} \\ \Leftrightarrow\quad & \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} && = \boldsymbol{X}^T\boldsymbol{t} \end{aligned}$$

Gradient and Stationary Point

The gradient is given by $\nabla\mathcal{L}(\mathbf{w}) = \frac{2}{N}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \frac{2}{N}\boldsymbol{X}^T\boldsymbol{t}$. Therefore:

$$\begin{aligned} & \nabla\mathcal{L}(\mathbf{w}) && = \mathbf{0} \\ \Leftrightarrow\quad & \frac{2}{N}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \frac{2}{N}\boldsymbol{X}^T\boldsymbol{t} && = \mathbf{0} \\ \Leftrightarrow\quad & \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} && = \boldsymbol{X}^T\boldsymbol{t} \end{aligned}$$

Finally, let's multiply both sides (from left) with $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$. This yields

$$\boldsymbol{I}\mathbf{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{t}$$

where $\boldsymbol{I} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}(\boldsymbol{X}^T\boldsymbol{X})$ is the identity matrix. This yields:

Minimizer for Linear Regression

$$\hat{\mathbf{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{t}$$

Prediction

Let $\mathbf{x}_{new} \in \mathbb{R}^D$ be a new point. How can we compute the predicted value?

1. Prepend a one: $[1, \mathbf{x}_{new}^T]$
2. Compute $t_{new} = [1, \mathbf{x}_{new}^T]\hat{\mathbf{w}}$

Example

	Tumor Size	Temperature	Survival
Case1	0.5	37.6	3.2
Case2	2.3	39.1	1.9
Case3	2.9	36.2	1.0

$$X = \begin{bmatrix} 1 & 0.5 & 37.6 \\ 1 & 2.3 & 39.1 \\ 1 & 2.9 & 36.2 \end{bmatrix} \qquad t = \begin{bmatrix} 3.2 \\ 1.9 \\ 1 \end{bmatrix}$$

Minimizer for Linear Regression

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$$

Example

Minimizer for Linear Regression

$$\hat{\mathbf{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{t}$$

$$\boldsymbol{X}^T\boldsymbol{X} = \begin{bmatrix} 1 & 1 & 1 \\ 0.5 & 2.3 & 2.9 \\ 37.6 & 39.1 & 36.2 \end{bmatrix} \begin{bmatrix} 1 & 0.5 & 37.6 \\ 1 & 2.3 & 39.1 \\ 1 & 2.9 & 36.2 \end{bmatrix} =$$

$$= \begin{bmatrix} 1+1+1 & 0.5+2.3+2.9 & 37.6+39.1+36.2 \\ 0.5+2.3+2.9 & 0.5^2+2.3^2+2.9^2 & 18.8+89.93+104.98 \\ 37.6+39.1+36.2 & 18.8+89.93+104.98 & 37.6^2+39.1^2+36.2^2 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & 5.7 & 112.9 \\ 5.7 & 13.95 & 213.71 \\ 112.9 & 213.71 & 4253.01 \end{bmatrix}$$

$$(\boldsymbol{X}^T\boldsymbol{X})^{-1} = \begin{bmatrix} 364.64 & -3.05 & -9.53 \\ -3.05 & 0.37 & 0.06 \\ -9.53 & 0.06 & 0.25 \end{bmatrix}$$

Example

Minimizer for Linear Regression

$$\hat{\mathbf{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{t}$$

$$(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T = \begin{bmatrix} 364.64 & -3.05 & -9.53 \\ -3.05 & 0.37 & 0.06 \\ -9.53 & 0.06 & 0.25 \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 \\ 0.5 & 2.3 & 2.9 \\ 36.6 & 39.1 & 36.2 \end{bmatrix} =$$

$$= \begin{bmatrix} 4.92 & -14.86 & 10.94 \\ -0.54 & 0.23 & 0.25 \\ -0.35 & 0.39 & -0.29 \end{bmatrix}$$

$$(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{t} = \begin{bmatrix} 4.45 & -14.86 & 10.94 \\ -0.54 & 0.23 & 0.25 \\ -0.35 & 0.39 & -0.29 \end{bmatrix}\begin{bmatrix} 3.2 \\ 1.9 \\ 1 \end{bmatrix} = \begin{bmatrix} -1.54 \\ -0.84 \\ 0.14 \end{bmatrix}$$

Example

	Tumor Size	Temperature	Survival
Case1	0.5	37.6	3.2
Case2	2.3	39.1	1.9
Case3	2.9	36.2	1.0

$$\hat{\boldsymbol{w}} = \begin{bmatrix} -1.54 \\ -0.84 \\ 0.14 \end{bmatrix}$$

Let's verify the solution:

$$f(\boldsymbol{x};\ \boldsymbol{w}) = \boldsymbol{x}^T\boldsymbol{w} = \begin{bmatrix} \boldsymbol{bias} \\ \boldsymbol{tumor\ size} \\ \boldsymbol{temperature} \end{bmatrix} [-1.54 \quad -0.84 \quad 0.14]$$

Larger the tumor is the worse is the survival prognosis

Higher temperature is good(?) for the survival prognosis

Example

$$\hat{\boldsymbol{w}} = \begin{bmatrix} -1.54 \\ -0.84 \\ 0.14 \end{bmatrix}$$

Let's compute the survival predicted by the model:

	Tumor Size	Temperature	Survival	Predicted survival
Case1	0.5	37.6	3.2	$1.54 - 0.84 \cdot 0.5 + 0.14 \cdot 37.6 = 3.304$
Case2	2.3	39.1	1.9	
Case3	2.9	36.2	1.0	

Example: compare linear regressions

	Tumor Size	Survival	Predicted Survival
Case1	0.5	3.2	3.25
Case2	2.3	1.9	1.68
Case3	2.9	1.0	1.16

$$L(\boldsymbol{w}) = (3.2 - 3.25)^2 + (1.9 - 1.68)^2 + (1.0 - 1.16)^2 = 0.0722$$

	Tumor Size	Temperature	Survival	Predicted Survival
Case1	0.5	37.6	3.2	3.30
Case2	2.3	39.1	1.9	2.00
Case3	2.9	36.2	1.0	1.09

$$L(\boldsymbol{w}) = (3.2 - 3.30)^2 + (1.9 - 2)^2 + (1.0 - 1.09)^2 = 0.0281$$

The results for multivariate case are better. Is this a coincidence?

Example: compare linear regressions

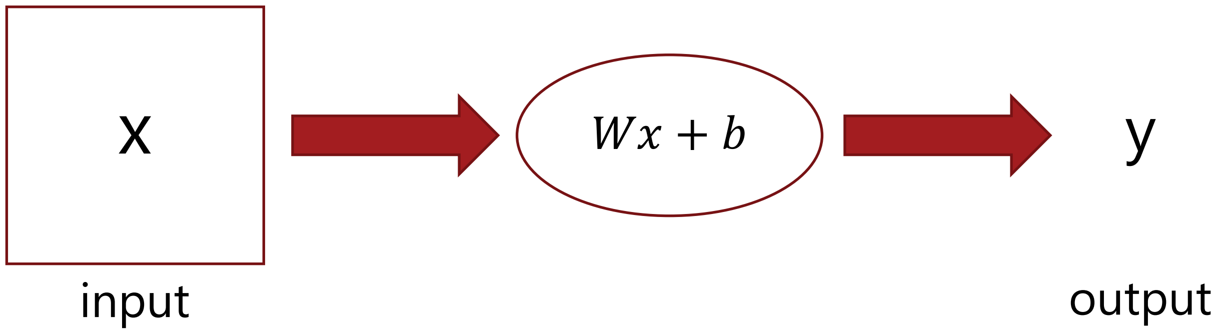
The more data dimension we add, the lower is the regression error (on training data):

- The solution for 1D regression is $\widehat{\boldsymbol{w}} = [\mathbf{3.69}, -\mathbf{0.87}]$
- The solution for 2D regression could be $\widehat{\boldsymbol{w}} = [\mathbf{3.69}, -\mathbf{0.87}, \mathbf{0}]$, which will give the same error as the solution for 1D regression
- The 2D regression can definitely reach loss of 1D regression 0.0722, and potentially improve it

Iterative optimization

Linear regression:

- Input x is 1-dimensional (tumor size)
- Output y is 1-dimensional (survival time)



Iterative optimization: initialization

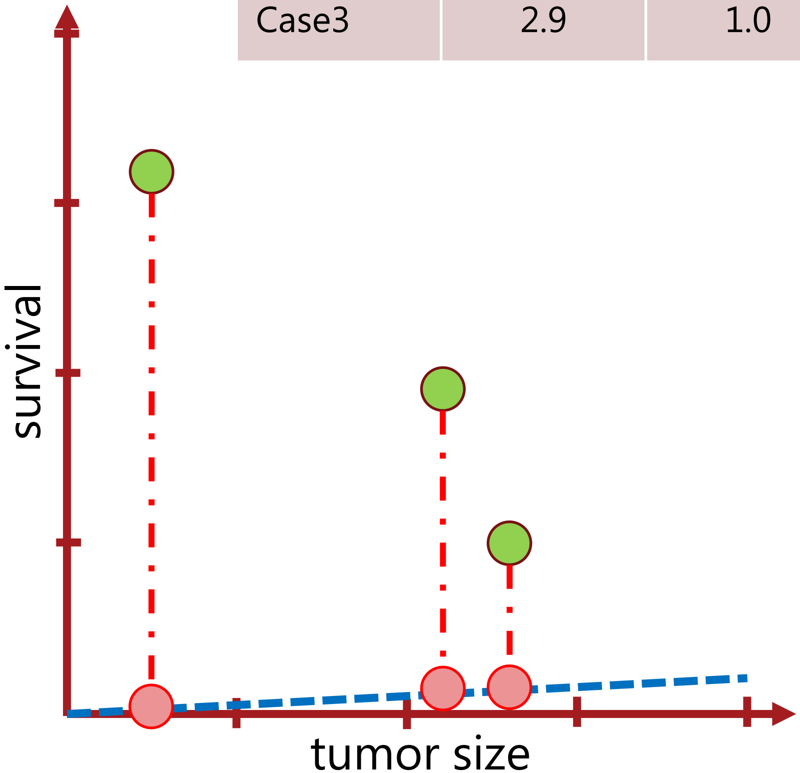
	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0

Let's initialize our model:

- $W = 0.01;\ b = 0.01$

The performance of the model is low:

- $y_1 = Wx_1 + b = 0.01 * 0.5 + 0.01 = 0.015$
- $y_2 = Wx_2 + b = 0.01 * 2.3 + 0.01 = 0.033$
- $y_3 = Wx_3 + b = 0.01 * 2.9 + 0.01 = 0.039$



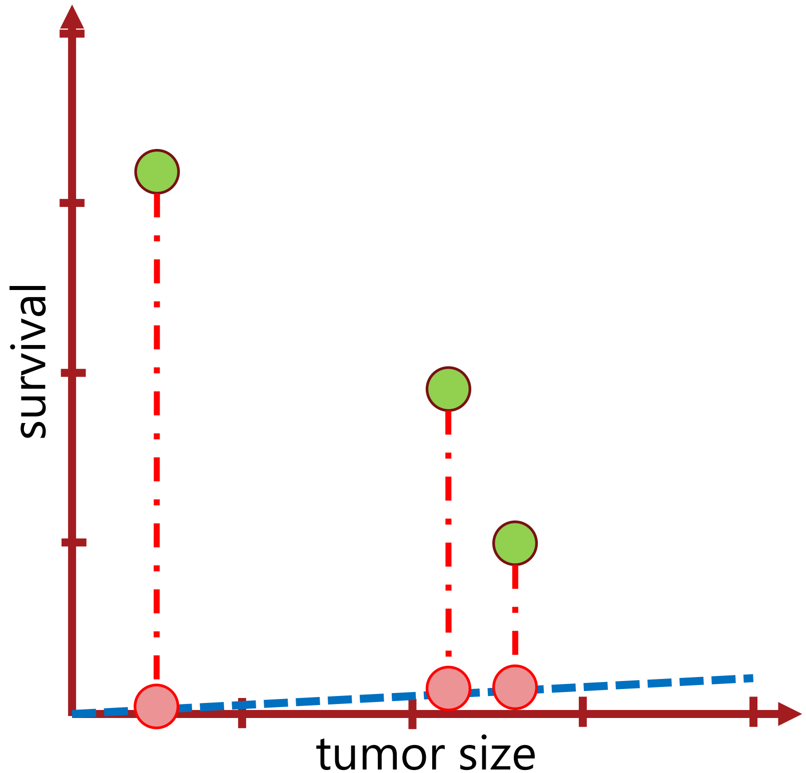
Iterative optimization: loss function

	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0

The performance of the model is low, but how low?

Sum of absolute differences (MAE):

- $Loss = \sum_i |y_i' - y_i| = \sum_i |(Wx_i + b_i) - y_i|$
- $Loss = |0.015 - 3.2| + |0.033 - 1.9| + |0.039 - 1.0| = 6.013$



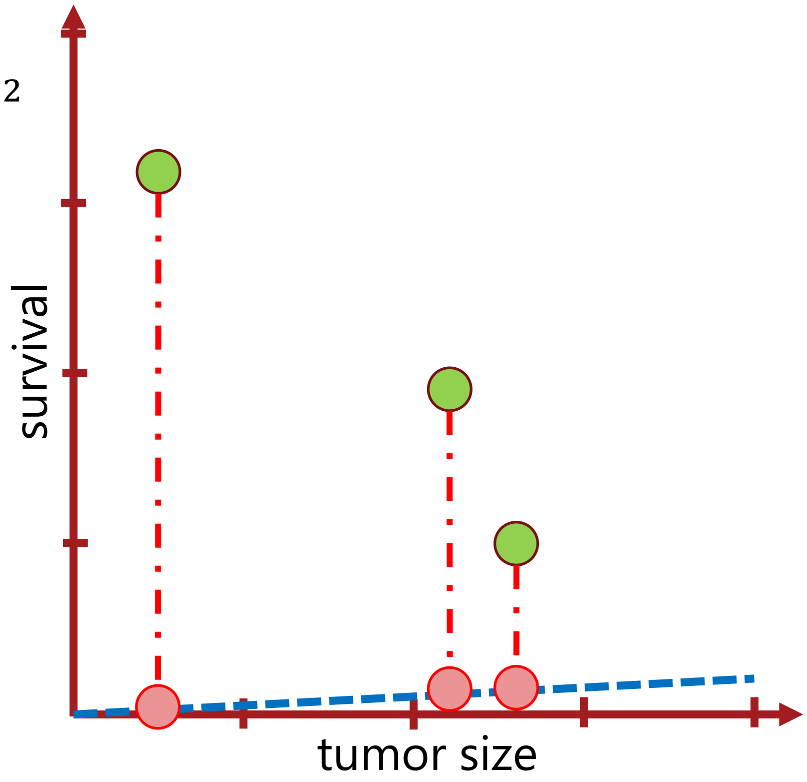
Iterative optimization: loss function

	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0

The performance of the model is low, but how low?

Mean squared error:

- $Loss = \sum_i (y_i' - y_i)^2 = \sum_i ((Wx + b) - y)^2$
- $Loss =$
  $(0.015 - 3.2)^2 +$
  $(0.033 - 1.9)^2 +$
  $(0.039 - 1.0)^2 = 14.55$



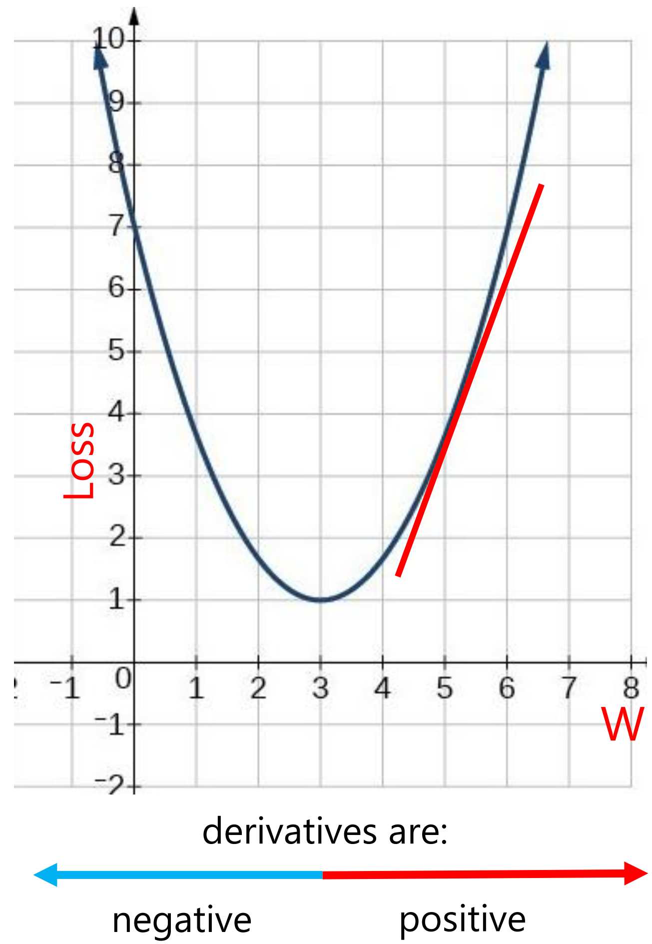
Iterative optimization: derivatives

We want the loss to be as small as possible, i.e. find its minimum.

We use derivatives to find minima/maxima of a function:

- How fast function changes
- Will it increase or decrease

- Chain rule:
  - $\frac{\partial}{\partial x} f\big(g(x)\big) = \frac{\partial}{\partial g} f\big(g(x)\big) \cdot \frac{\partial}{\partial x} g(x)$



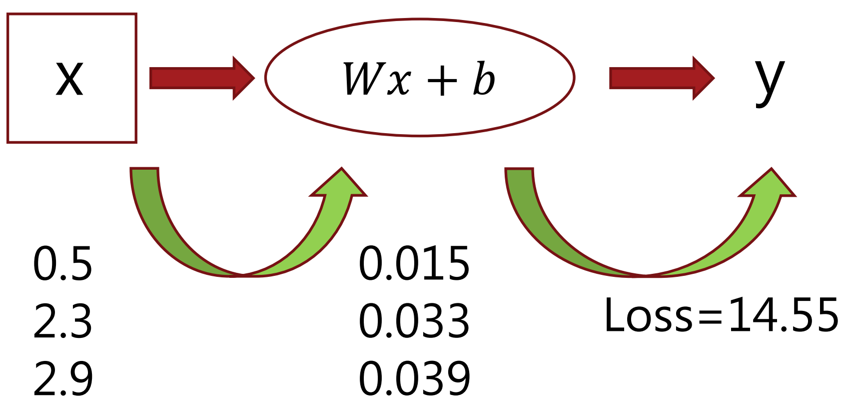
Iterative optimization: backpropagation

	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0

Initialization:

- $W = 0.01, b = 0.01$

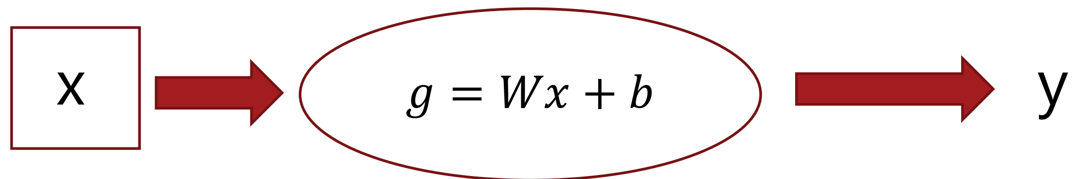
Forward pass:



Iterative optimization: backpropagation

Compute the derivatives:

$$Loss = \sum_i (y' - y)^2$$



- How changes in $Wx + b$ affect the loss

$$\frac{\partial g}{\partial W} = x$$

$$\frac{\partial Loss}{\partial g} = 2\sum_i (g(x_i) - y_i)$$

14.55

- How changes in W and b affect g

$$\frac{\partial g}{\partial b} = 1$$

- $\frac{\partial Loss}{\partial W}$ and $\frac{\partial Loss}{\partial b}$ will be computed using the chain rule:
  - $\frac{\partial Loss}{\partial W} = 2\sum_i \big((g(x_i) - y_i) \cdot x_i\big)$
  - $\frac{\partial Loss}{\partial b} = 2\sum_i \big((g(x_i) - y_i) \cdot 1\big)$

Iterative optimization: optimization step

Update W and b according to the derivatives:

- $W \leftarrow W - \lambda \frac{\partial Loss}{\partial W}$; $b \leftarrow b - \lambda \frac{\partial Loss}{\partial b}$

Learning rate

The results of the optimization step:

- $W \leftarrow W - 0.1 \cdot 2 \sum_i \big((g(x_i) - y_i) \cdot x_i\big) = 0.01 - 0.1 \cdot 2 \cdot \big((0.015 - 3.2)0.5 + (0.033 - 1.9)2.3 + (0.039 - 1.0)2.9\big) = 0.01 - 0.1 \cdot -8.67 = 0.88$

W	b
0.01	0.01

- $b \leftarrow b - 0.1 \cdot 2 \sum_i \big((g(x_i) - y_i)\big) = 0.61$

	Tumor Size	Survival	First solution
Case1	0.5	3.2	0.015
Case2	2.3	1.9	0.033
Case3	2.9	1.0	0.039

Iterative optimization: optimization step

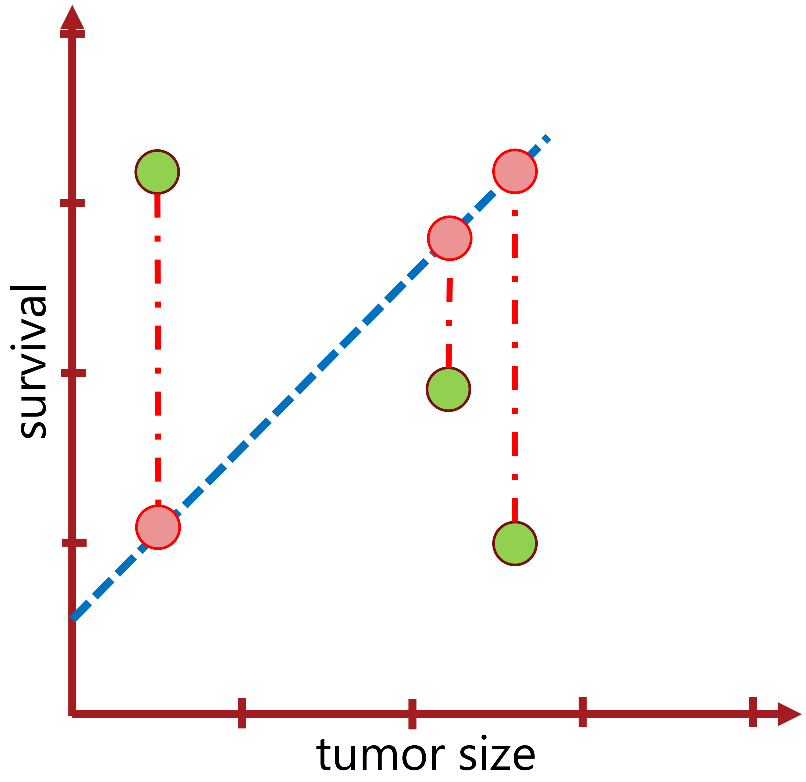
	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0

After parameter update:

- $W = 0.88;\ b = 0.61$

The performance of the model is better:

- $y_1 = Wx_1 + b = 0.70 * 0.5 + 0.61 = 1.05$
- $y_2 = Wx_2 + b = 0.70 * 2.3 + 0.61 = 2.63$
- $y_3 = Wx_3 + b = 0.70 * 2.9 + 0.61 = 3.16$



Loss improves $Loss\ = 9.8$

Iterative optimization: optimization step

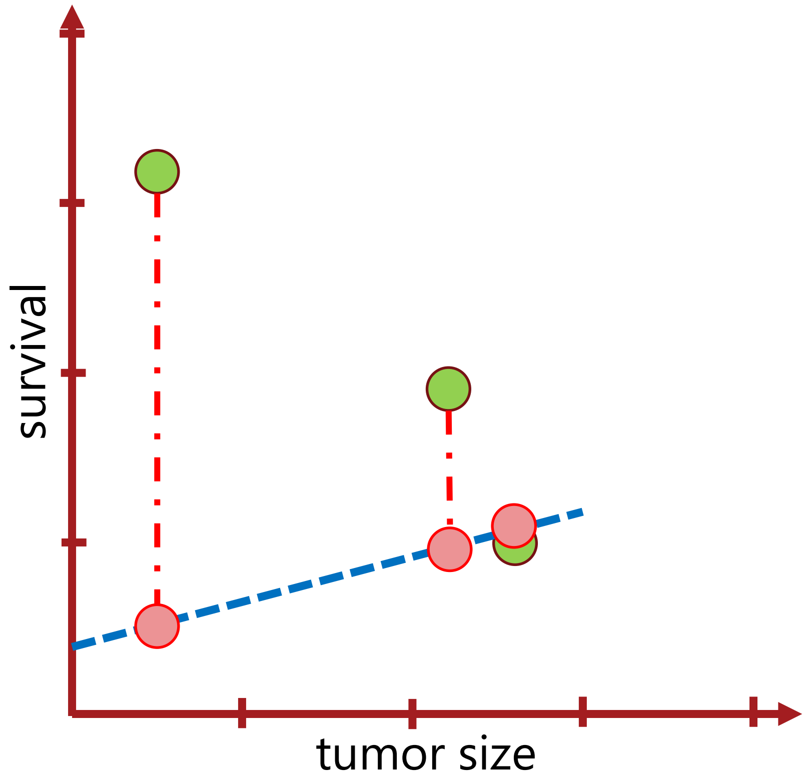
	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0

After next parameter update:

- $W = 0.19;\ b = 0.54$

The performance of the model is better:

- $y_1 = Wx_1 + b = 0.45 * 0.5 + 0.38 = 0.63$
- $y_2 = Wx_2 + b = 0.45 * 2.3 + 0.38 = 0.98$
- $y_3 = Wx_3 + b = 0.45 * 2.9 + 0.38 = 1.10$



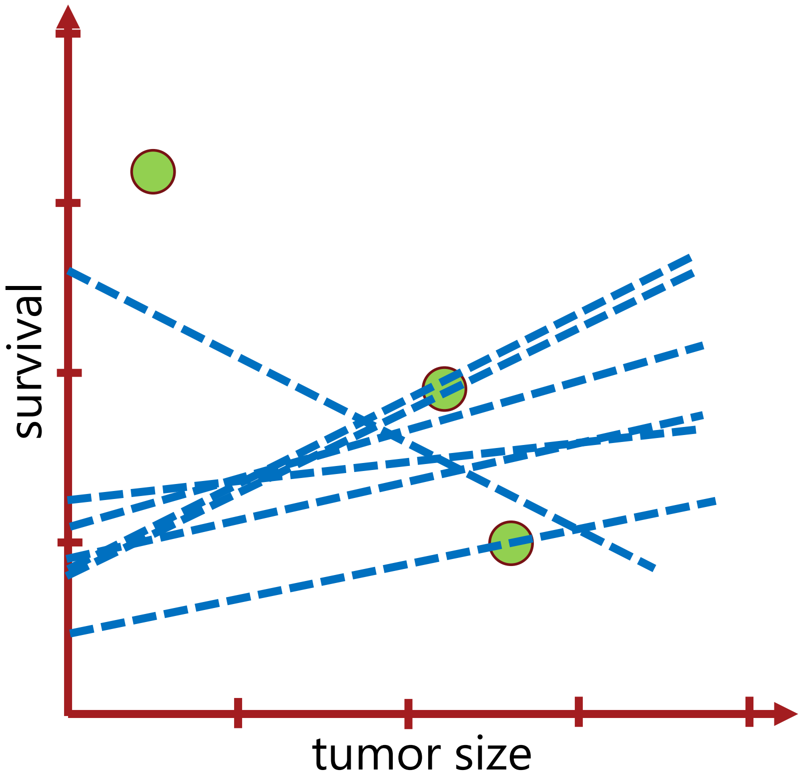
Loss improves $Loss\ = 7.46$

Iterative optimization: optimization step

After next parameter update:

- $W = 0.50;\ b = 0.88, Loss = 6.07$
- $W = 0.19;\ b = 0.50, Loss = 7.75$
- $W = 0.53;\ b = 0.85, Loss = 6.29$
- $W = 0.19;\ b = 0.90, Loss = 5.37$
- $W = 0.29;\ b = 1.13, Loss = 4.67$
- $W = 0.12;\ b = 1.23, Loss = 4.14$
- …
- $W = 0.04;\ b = 1.51, Loss = 3.27$
- $W = -0.15;\ b = 1.98, Loss = 2.06$
- $W = -0.22;\ b = 2.18, Loss = 1.63$
- $W = -0.29;\ b = 2.35, Loss = 1.31$
- $W = -0.36;\ b = 2.50, Loss = 1.04$
- $W = -0.43;\ b = 2.63, Loss = 0.82$

	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0



Iterative optimization

Instead of explicitly calculating the global derivative, we can calculate derivatives step by step:

- Eventually, we will calculate how each node is affects the next one
- By using the chain rule, we can estimate how change of b_0 and b_1 will affect the loss L
- We can then slightly modify values of b_0 and b_1 to reduce the loss L, and then recompute the new loss and the repeat the procedure

Coding

jupyter Multivariate Linear Regression Last Checkpoint: a few seconds ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

copy selected cells

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy
```

We shall work with the dataset found in the file 'murderdata.txt', which is a 20 x 5 data matrix where the columns correspond to

Index (not for use in analysis)

Number of inhabitants

Percent with incomes below $5000

Percent unemployed

Murders per annum per 1,000,000 inhabitants

Reference:

Helmut Spaeth, Mathematical Algorithms for Linear Regression, Academic Press, 1991, ISBN 0-12-656460-4.

D G Kleinbaum and L L Kupper, Applied Regression Analysis and Other Multivariable Methods, Duxbury Press, 1978, page 150.

http://people.sc.fsu.edu/~jburkardt/datasets/regression

What to do?

We start by loading the data; today we will study how the number of murders relates to the percentage of unemployment.

In [ ]:
```
data = numpy.loadtxt('murderdata.txt')
N, d = data.shape
```

We consider all both features simulaneously.

In [ ]:
```
t = data[:,4]
X = data[:,2:4]
print("Number of training instances: %i" % X.shape[0])
print("Number of features: %i" % X.shape[1])
```

In [ ]:
```
# NOTE: This template makes use of Python classes. If
# you are not yet familiar with this concept, you can
# find a short introduction here:
# http://introtopython.org/classes.html

class LinearRegression():
    """
    Linear regression implementation.
    """

    def __init__(self):
```

Coding

Computation in Practice

1. Definition of data matrix $\mathbf{X} \in \mathbb{R}^{N \times (D+1)}$ (make use of Numpy arrays and functions!)
2. There are different ways to compute an optimal weight vector $\hat{\mathbf{w}}$:
   1. Compute $\hat{\mathbf{w}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{t}$ (e.g., via `numpy.linalg.inv`)
   2. Directly solve the system of equations (e.g., via `numpy.linalg.solve`):

$$\begin{aligned} \nabla \mathcal{L}(\mathbf{w}) &= \mathbf{0} \\ \Leftrightarrow \quad \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w} &= \boldsymbol{X}^T \boldsymbol{t} \end{aligned}$$

   3. ...
3. For new point $\mathbf{x}_{new} \in \mathbb{R}^D$: Compute $t_{new} = [1, \mathbf{x}_{new}^T]\hat{\mathbf{w}}$

```
In [ ]: t = data[:,4]
        X = data[:,2:4]
        print("Number of training instances: %i" % X.shape[0])
        print("Number of features: %i" % X.shape[1])
```

```
In [ ]: # NOTE: This template makes use of Python classes. If
        # you are not yet familiar with this concept, you can
        # find a short introduction here:
        # http://introtopython.org/classes.html

        class LinearRegression():
```

Questions?