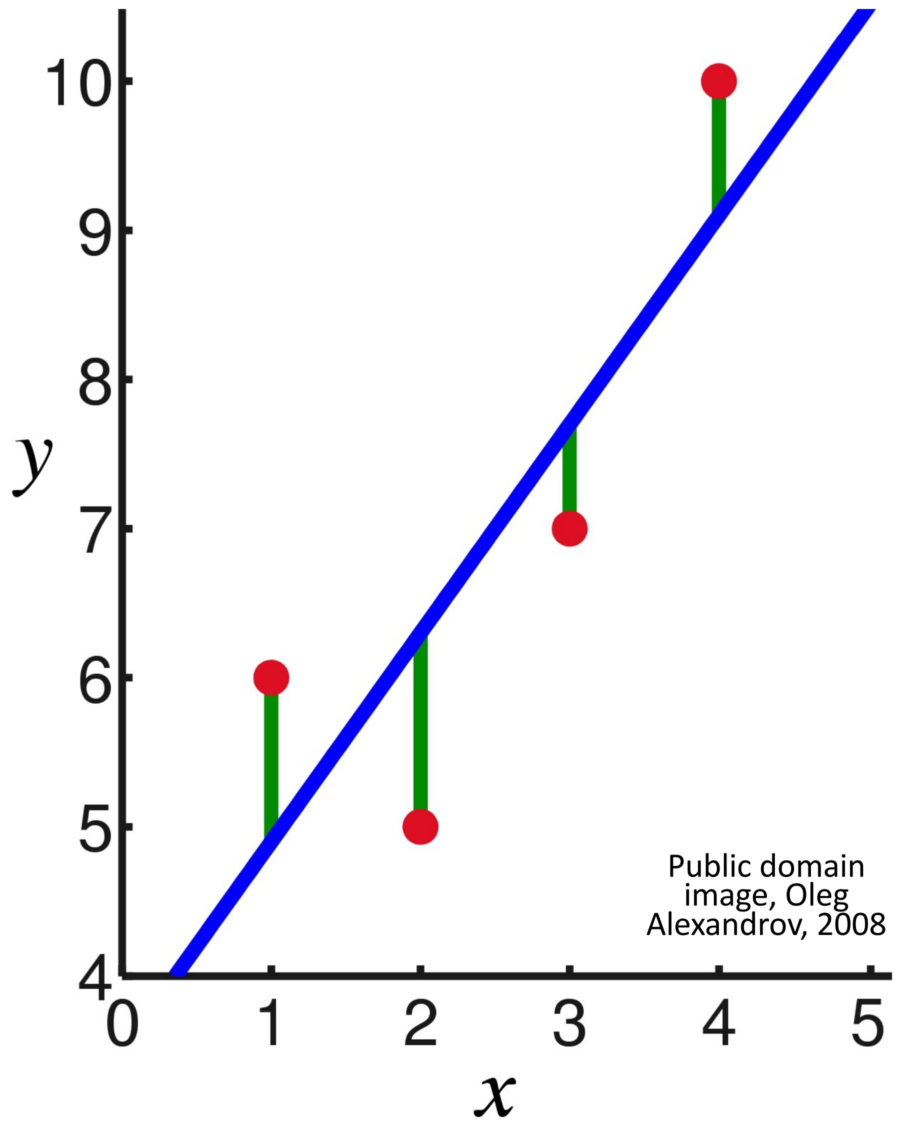
CS440/ECE448 Lecture 7: Linear Regression

Mark Hasegawa-Johnson, 1/2022



Outline

- Review: perceptron
- Linear regression: a neuron without the nonlinearity
- Mean-squared error
- Learning the solution: stochastic gradient descent
- Multiple linear regression

Review: Perceptron Learning Algorithm

For each training instance $\vec{x}$ with ground truth label $y \in \{-1,1\}$:

- Classify with current weights: $f(\vec{x}) = \mathrm{sign}(\vec{w}^T \vec{x})$
- Update weights:
  - If $f(\vec{x}) = y$ then do nothing
  - If $f(\vec{x}) \neq y$ then

$$\vec{w} = \vec{w} + y\vec{x} = \begin{cases} \vec{w} + \vec{x} & y = +1 \\ \vec{w} - \vec{x} & y = -1 \end{cases}$$

Example

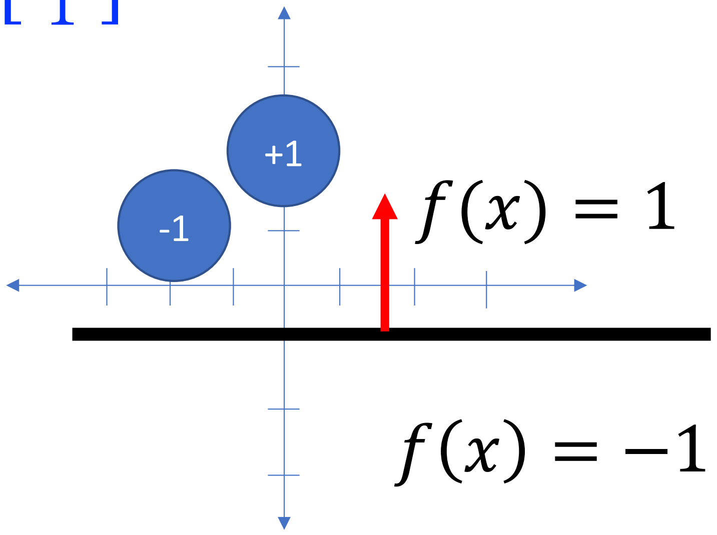
Now we have $\vec{w}^T = [0,2,1]$.

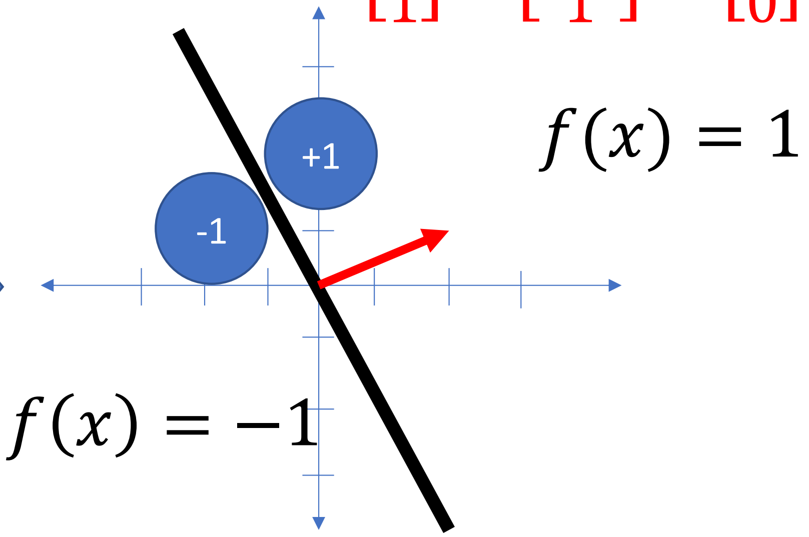
Suppose the next token is $\vec{x}^T = [-2,1,1]$, with the label $y = -1$. Since $f(x)$ is wrong, we update:

$$\vec{w} = \vec{w} - \vec{x}$$

$$\vec{x} = \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix}, y = -1$$

$$\vec{w} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} - \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$$





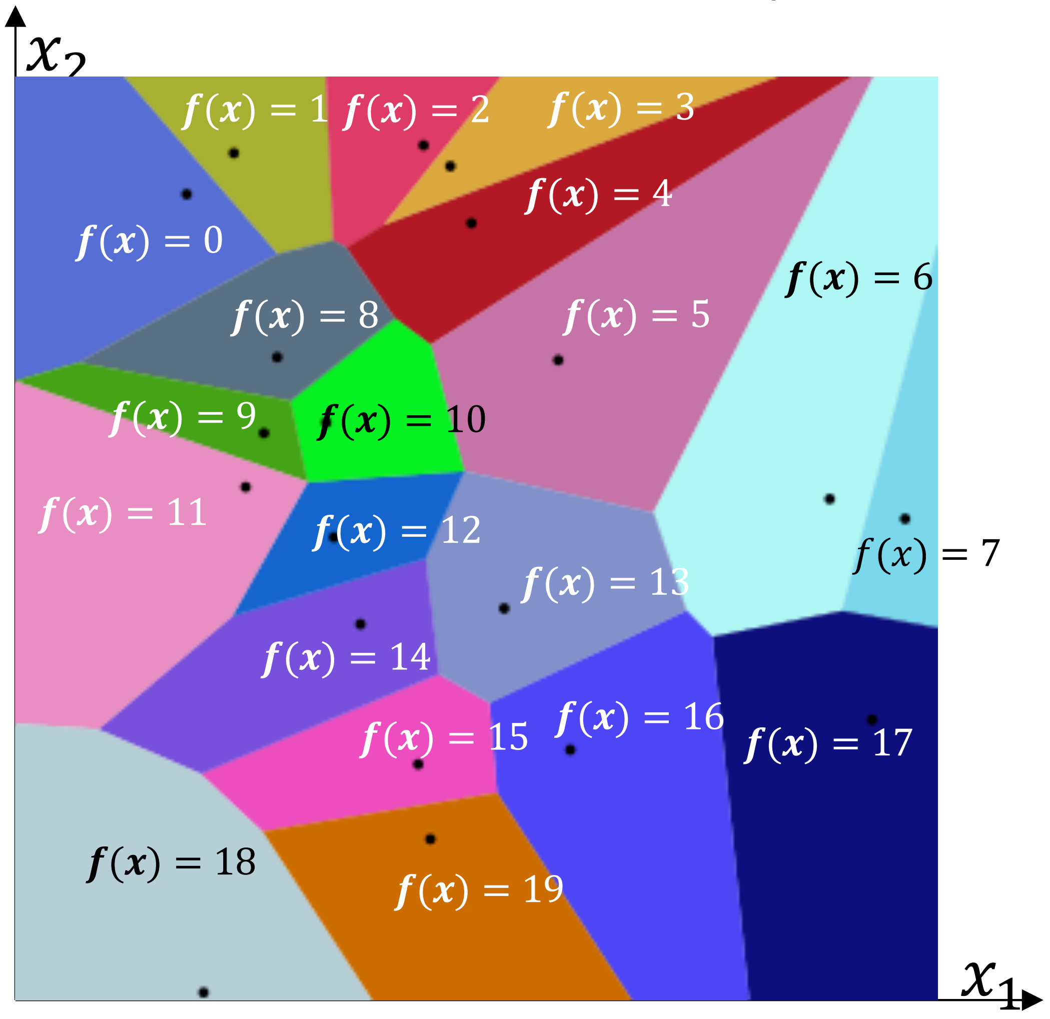
Review: Multi-Class Perceptron

For each training instance $\vec{x}$ with ground truth label $y \in \{-1,1\}$:

- Classify with current weights: $f(\vec{x}) = \underset{c}{\operatorname{argmax}}(\overrightarrow{w_c}^T \vec{x})$
- Update weights:
  - If $f(\vec{x}) = y$ then do nothing
  - If $f(\vec{x}) \neq y$ then

$$\overrightarrow{w}_y = \overrightarrow{w}_y + \eta\vec{x}$$
$$\overrightarrow{w}_{f(\vec{x})} = \overrightarrow{w}_{f(\vec{x})} - \eta\vec{x}$$

Notation: Vector dot product, with bias added



$$f(\vec{x}) = \underset{c}{\operatorname{argmax}}\left(\vec{w}_c^T \vec{x}\right)$$

$$\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{bmatrix}, \vec{w}_c = \begin{bmatrix} w_{c,1} \\ \vdots \\ w_{c,D} \\ b_c \end{bmatrix}$$

$$\vec{w}_c^T \vec{x} = b_c + \sum_{j=1}^{D} w_{c,j} x_j$$

Outline

- Review: perceptron
- Linear regression: a neuron without the nonlinearity
- Mean-squared error
- Learning the solution: stochastic gradient descent
- Multiple linear regression

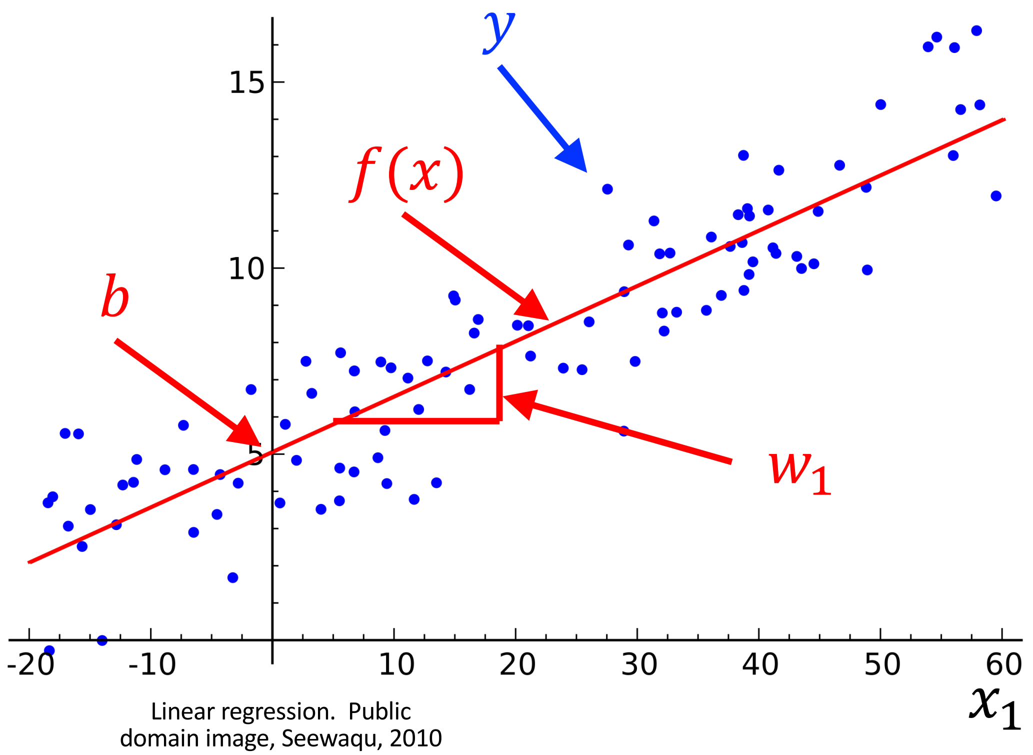
Linear regression

Linear regression is used to estimate a real-valued target variable, y, using a linear combination of real-valued input variables:

$$f(x) = \vec{w}^T \vec{x} = b + \sum_{j=1}^{D} w_j x_j$$

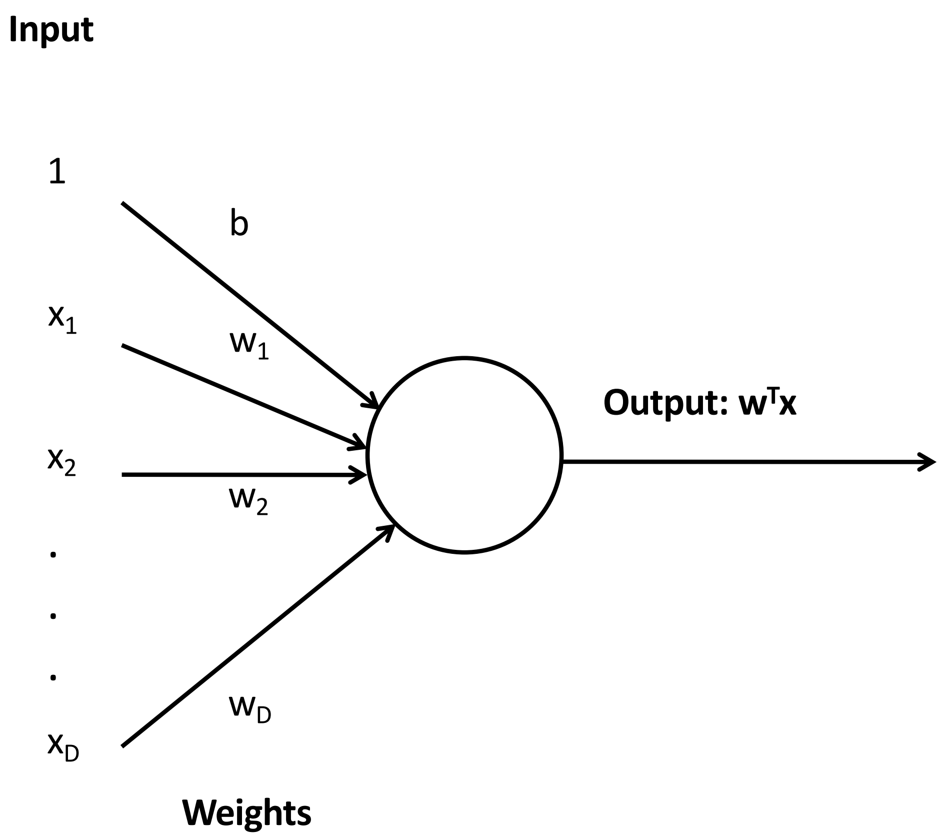
… so that …

$$f(x) \approx y$$



Linear regression. Public domain image, Seewaqu, 2010

Linear regression is like a neuron without a nonlinearity



- The neuron's **excitation** is

$$\vec{w}^T \vec{x} = b + \sum_{j=1}^{D} w_j x_j$$

- The neuron's **activation** is

$$f(x) = \vec{w}^T \vec{x}$$

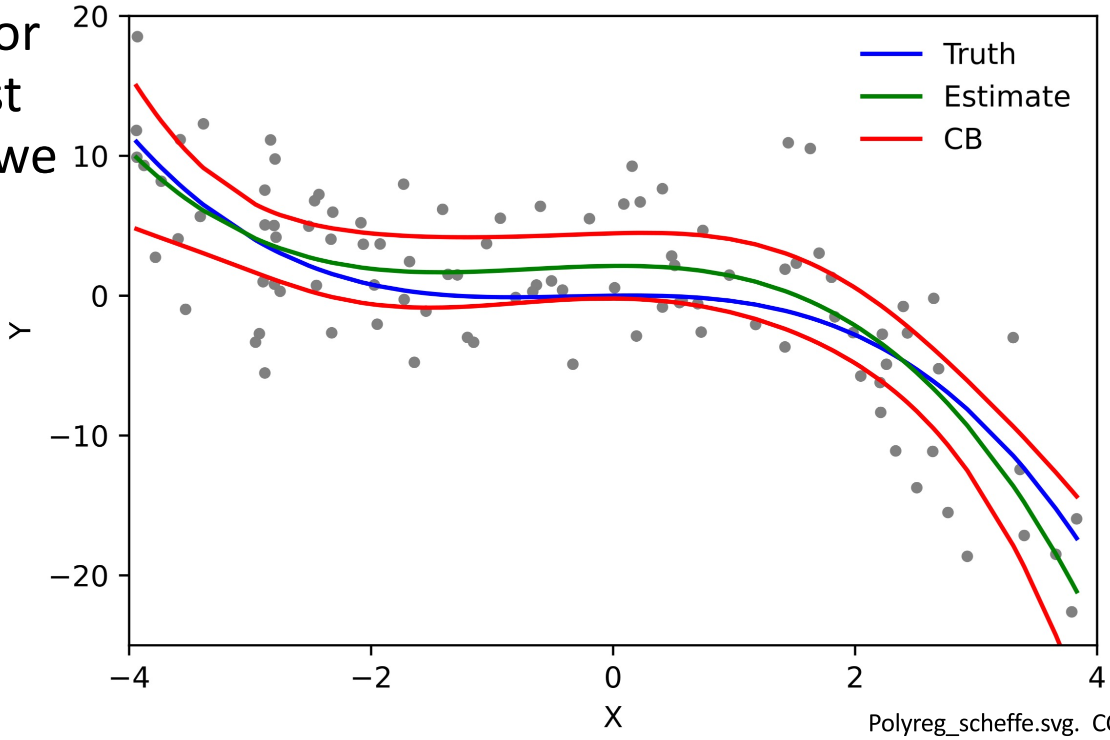
Polynomial regression = multivariate linear regression

We can use linear regression to solve nonlinear regression problems by simply augmenting the features. For example, suppose we start with just one input variable, x, but suppose we expand it to four variables like this:

$$\vec{x} = \begin{bmatrix} x \\ x^2 \\ x^3 \\ 1 \end{bmatrix}$$

Then

$$f(x) = \vec{w}^T \vec{x}$$
$$= b + w_1 x + w_2 x^2 + w_3 x^3$$



Polyreg_scheffe.svg. CC-BY 3.0, Skbkekas, 2009

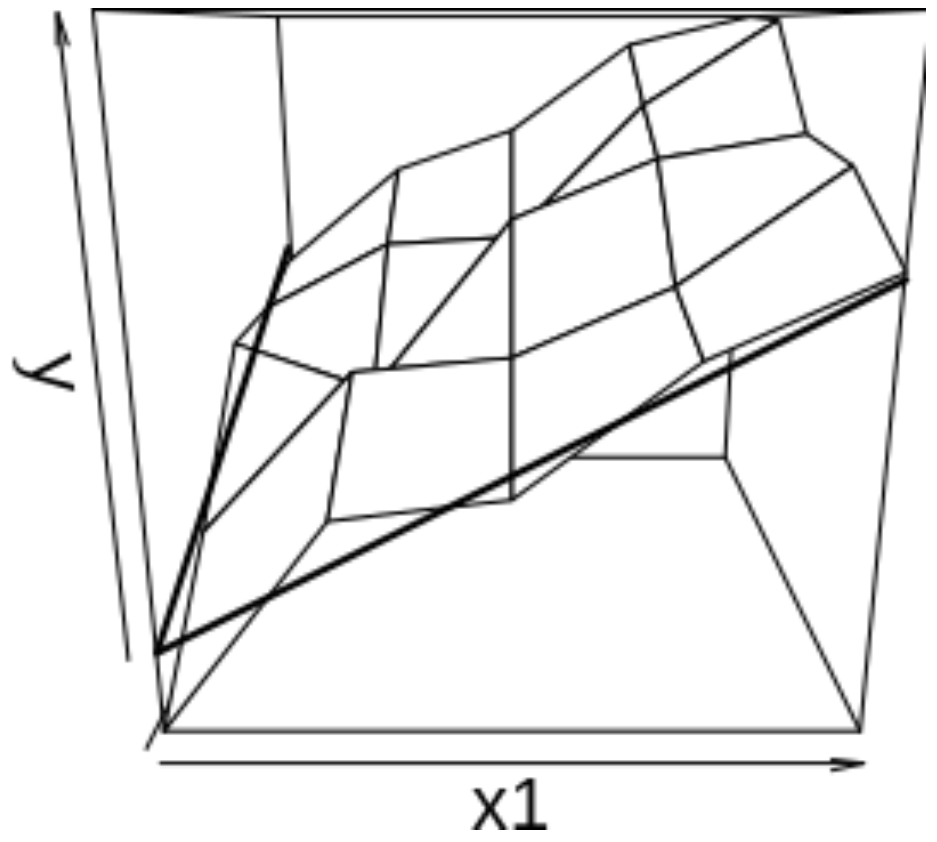
Multivariate linear regression in general

More generally, multivariate linear regression fits a D-dimensional hyperplane in the (D+1)-dimensional space $(x_1, \ldots, x_D, y)$:

$$\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{bmatrix}$$

Then

$$f(\vec{x}) = \vec{w}^T \vec{x} = b + \sum_{j=1}^{D} w_j x_j$$



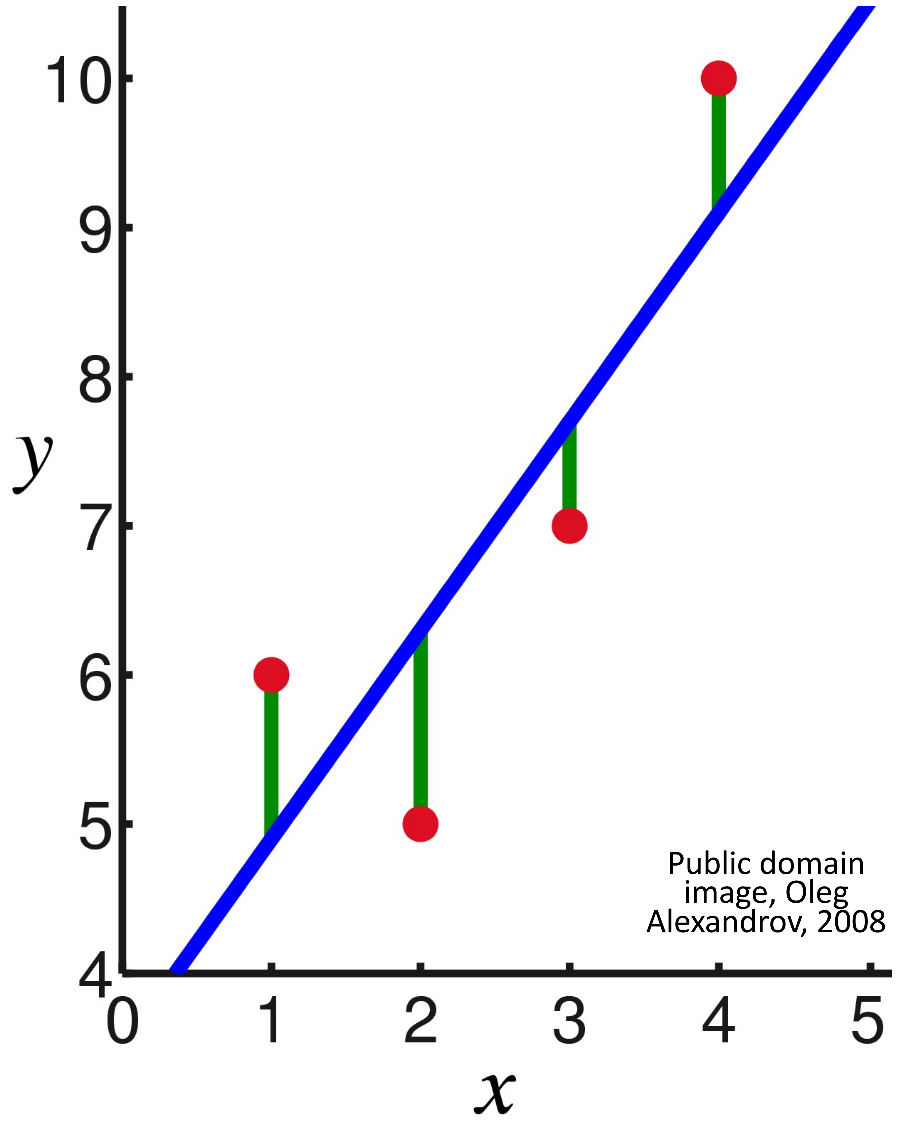
Regression_plane_avc_R.svg.
CC-SA 3.0, Cdang, 2013

Outline

- Review: perceptron
- Linear regression: a neuron without the nonlinearity
- Mean-squared error
- Learning the solution: stochastic gradient descent
- Multiple linear regression

What does it mean that $f(x) \approx y$?

- Generally, we want to choose the weights and bias, $\vec{w}$, in order to minimize the errors.
- The errors are the vertical green bars in the figure at right,
  $$\epsilon = f(\vec{x}) - y$$
- Some of them are positive, some are negative. What does it mean to "minimize" them?



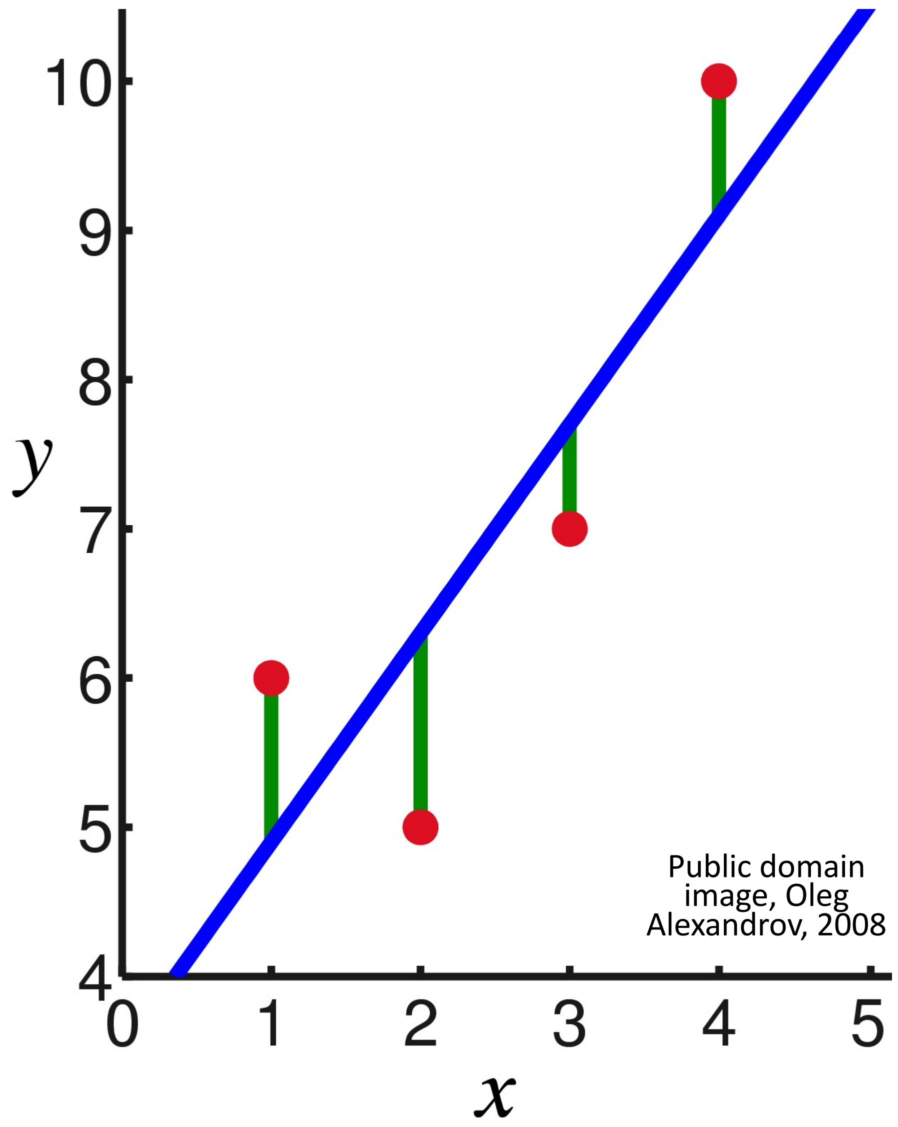
Public domain image, Oleg Alexandrov, 2008

First: count the training tokens

Let's introduce one more index variable. Let i=the index of the training token.

$$\vec{x}_i = \begin{bmatrix} x_{i,1} \\ \vdots \\ x_{i,D} \\ 1 \end{bmatrix}$$

$$f(\vec{x}_i) = \vec{x}_i^T \vec{w} = b + \sum_{j=1}^{D} x_{i,j} w_j$$

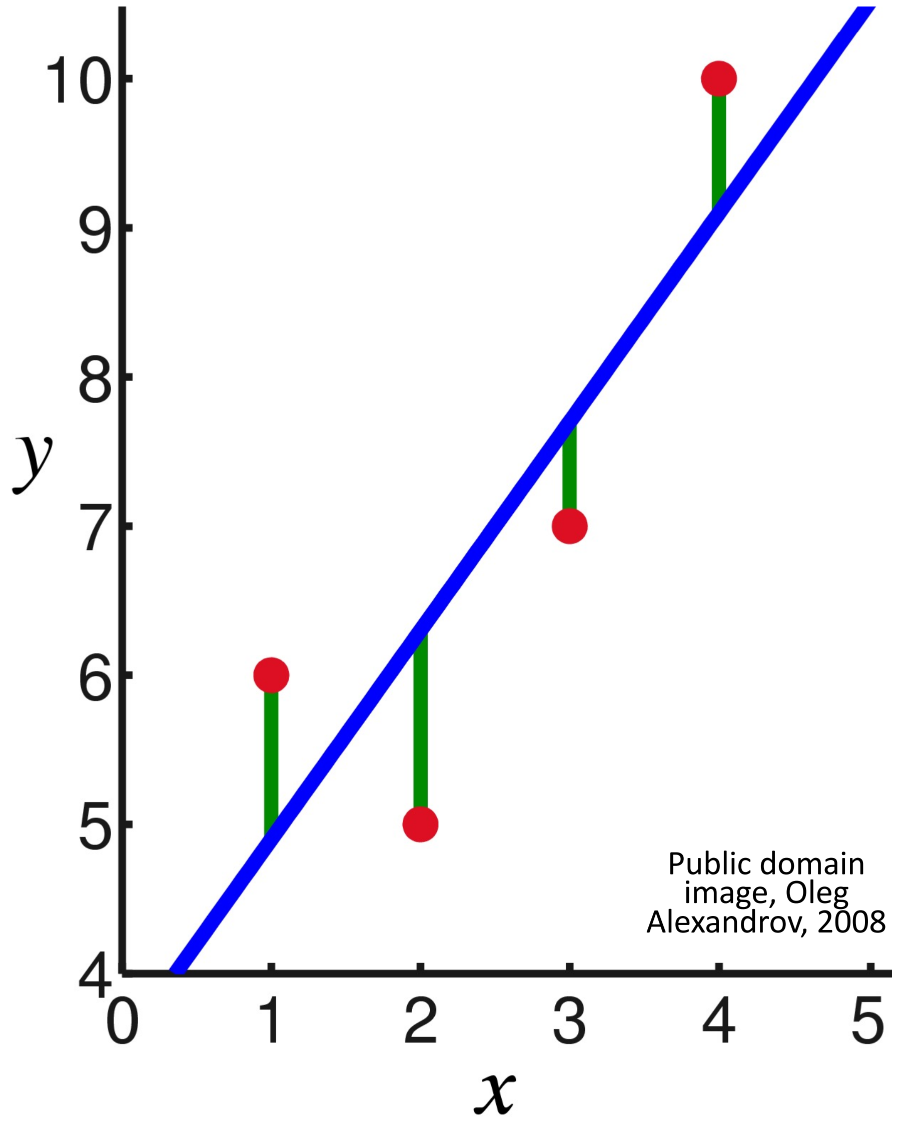


Training token errors

Using that notation, we can define a signed error term for every training token:

$$\epsilon_i = f(\vec{x}_i) - y_i$$

The error term is positive for some tokens, negative for other tokens. What does it mean to minimize it?



Public domain image, Oleg Alexandrov, 2008

Mean-squared error

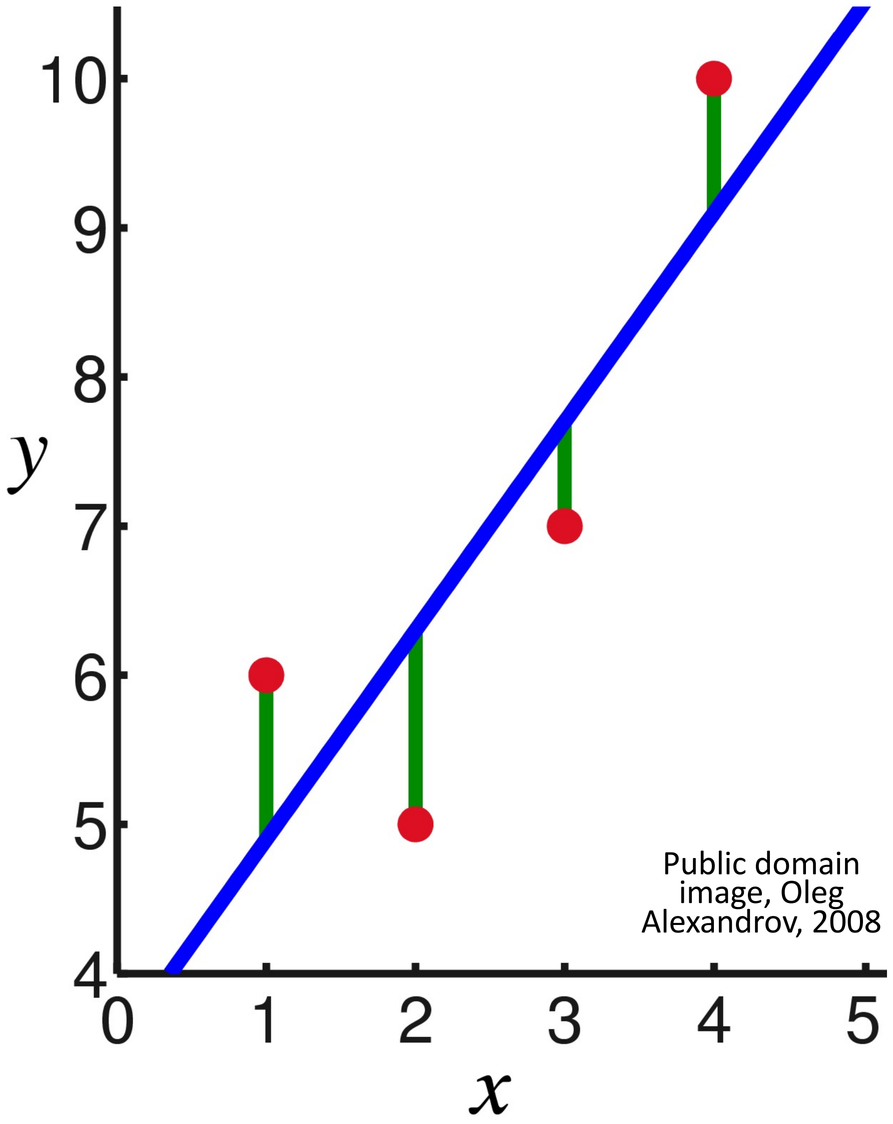
One useful criterion (not the only useful criterion, but perhaps the most common) of "minimizing the error" is to minimize the mean squared error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \epsilon_i^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} \left(\vec{x}_i^T \vec{w} - y_i\right)^2$$

Literally,

- … the mean …
- … of the square …
- … of the error terms.



Public domain image, Oleg Alexandrov, 2008

Outline

- Review: perceptron
- Linear regression: a neuron without the nonlinearity
- Mean-squared error
- Learning the solution: stochastic gradient descent
- Multiple linear regression

Minimizing the MSE

Our goal is to find the coefficients $\vec{w} = [w_1, \dots, w_D, b]^T$ that minimize the MSE:

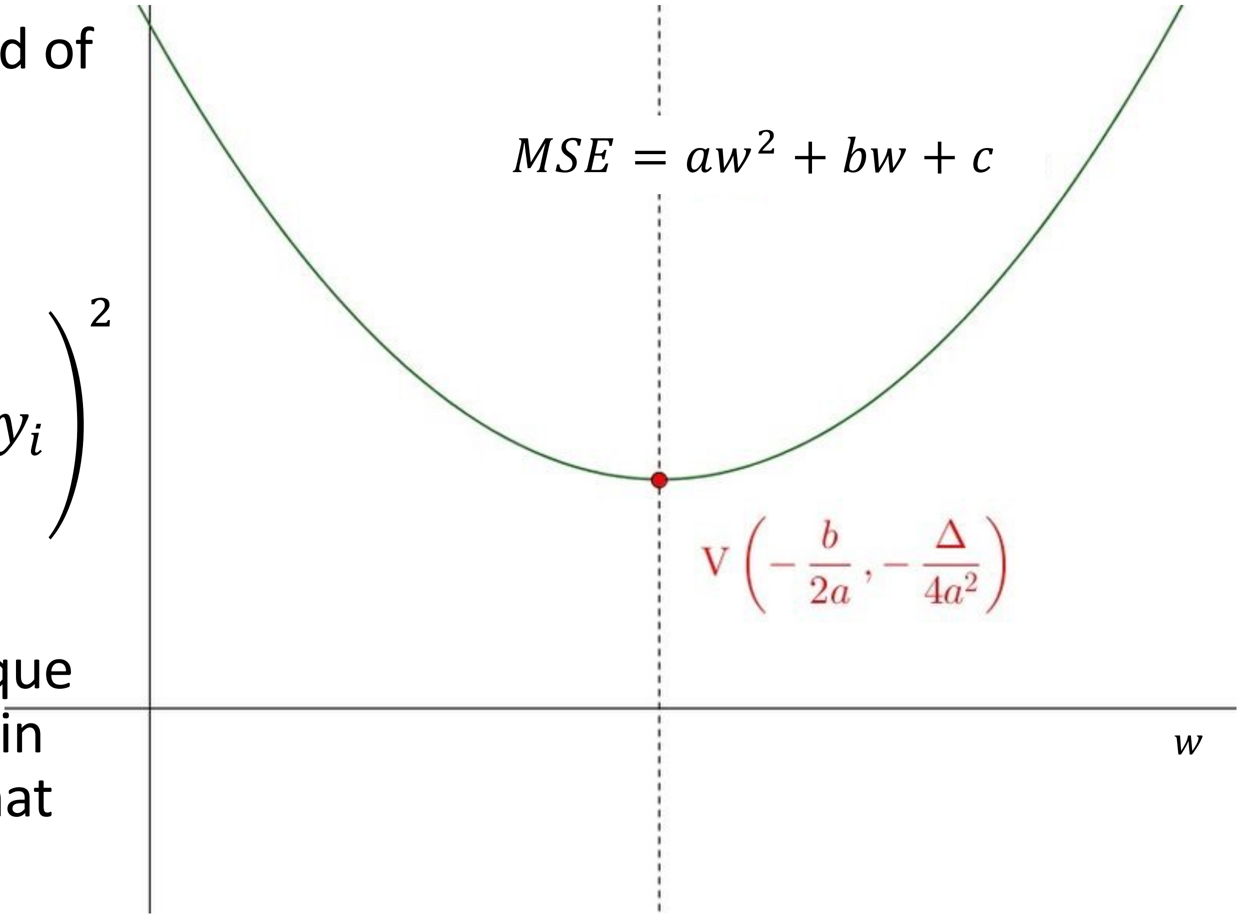
$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\vec{x}_i^T\vec{w} - y_i\right)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(b + \sum_{j=1}^{D} w_j x_{i,j} - y_i\right)^2$$

MSE = Parabola

Notice that, although it looks kind of complicated, the MSE is just a parabola in terms of b and w_j:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(b + \sum_{j=1}^{D} w_j x_{i,j} - y_i\right)^2$$

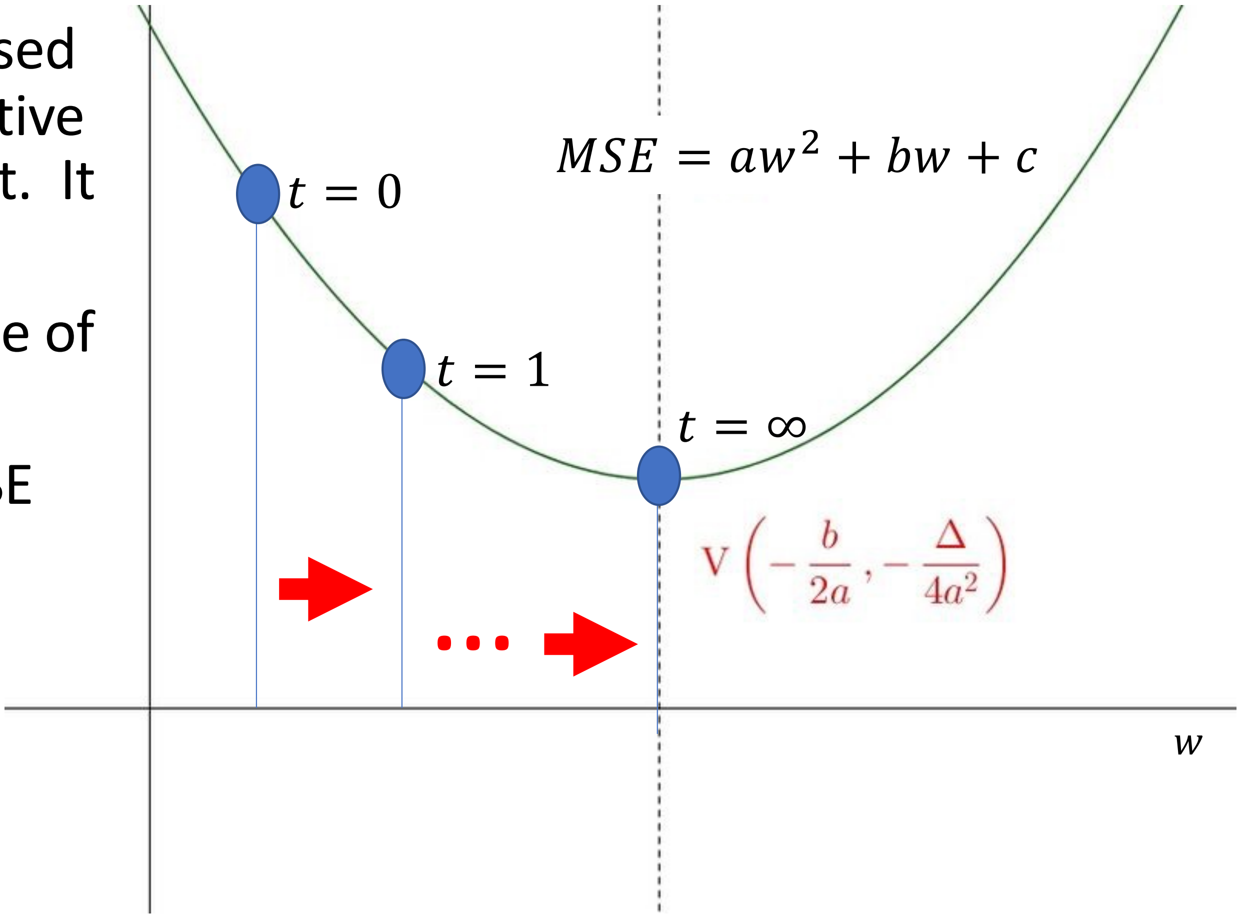


Since it's a parabola, it has a unique minimum that you can compute in closed form! But we won't do that today. Instead…

The iterative solution to linear regression

Instead of minimizing MSE in closed form, we're going to use an iterative algorithm called gradient descent. It works like this:

- Start from a random initial value of $\vec{w}$ (at $t = 0$).
- Adjust $\vec{w}$ in order to reduce MSE ($t = 1$).
- Repeat until you reach the optimum ($t = \infty$).



The gradient descent algorithm

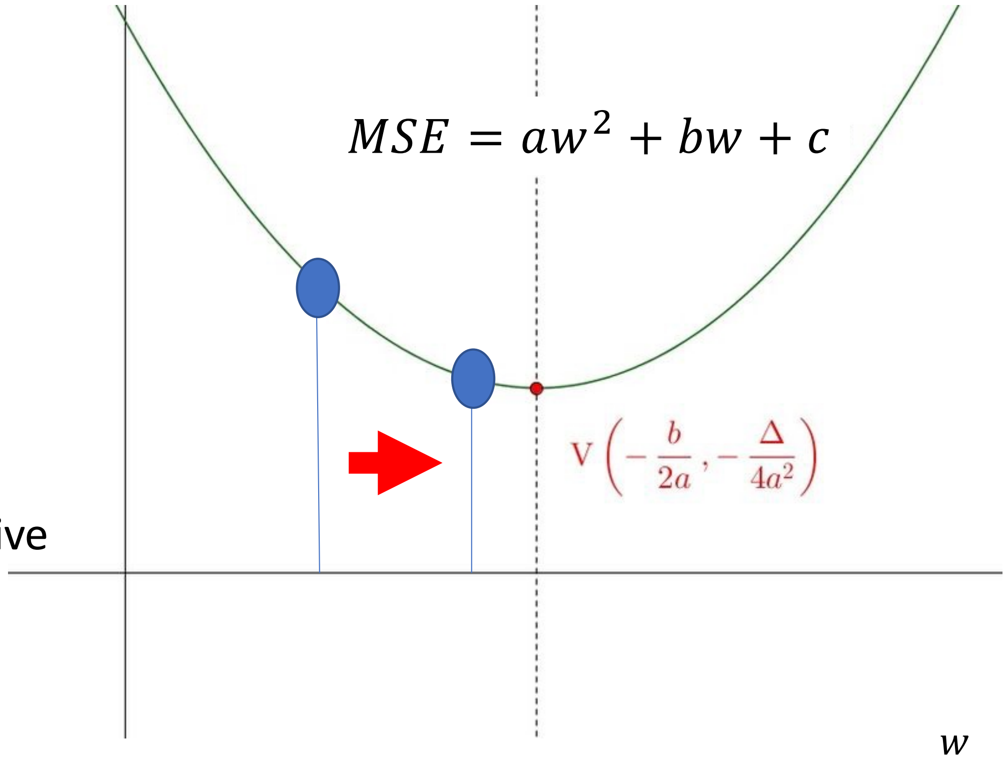
- Start from a random initial value of $\vec{w}$.
- Calculate the derivative of MSE with respect to $\vec{w}$:

$$\nabla_{\vec{w}} MSE = \begin{bmatrix} \frac{\partial MSE}{\partial w_1} \\ \vdots \\ \frac{\partial MSE}{\partial w_D} \\ \frac{\partial MSE}{\partial b} \end{bmatrix}$$

- Take a step "downhill" (in the direction of the negative gradient

$$\vec{w} \leftarrow \vec{w} - \frac{\eta}{2} \nabla_{\vec{w}} MSE$$

…where η is a constant called the "learning rate," that determines how big of a step you take. Usually, you need to adjust η in order to get optimum performance on a dev set, but often $\eta \approx 0.001$.



Stochastic gradient descent

- If n is large, computing or differentiating MSE can be expensive.
- The stochastic gradient descent algorithm picks one training token $(\vec{x}_i, y_i)$ at random (”stochastically”), and adjusts $\vec{w}$ in order to reduce the error a little bit for that one token:

$$\vec{w} \leftarrow \vec{w} - \frac{\eta}{2} \nabla_{\vec{w}} \epsilon_i^2$$

…where

$$\epsilon_i^2 = \left(\vec{x}_i^T \vec{w} - y_i\right)^2$$

Stochastic gradient descent

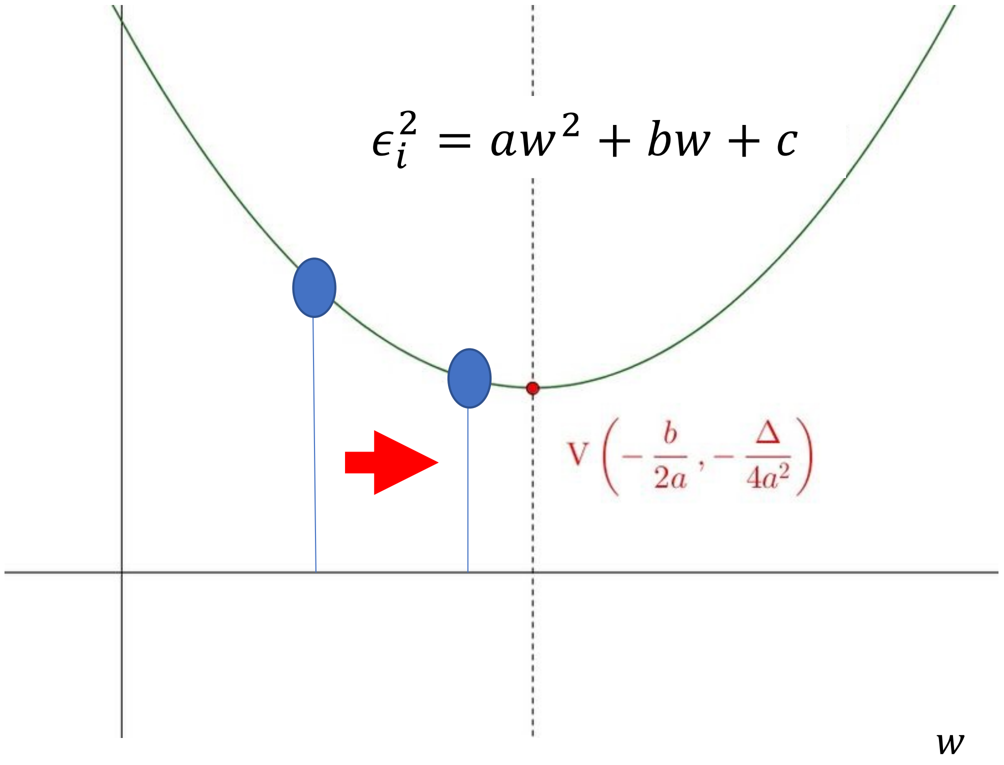
$$\epsilon_i^2 = \left(\vec{x}_i^T\vec{w} - y_i\right)^2$$

If we differentiate that, we discover that:

$$\nabla_{\vec{w}}\epsilon_i^2 = 2\epsilon_i\vec{x}_i$$

So the stochastic gradient descent algorithm is:

$$\vec{w} \leftarrow \vec{w} - \eta\epsilon_i\vec{x}_i$$



$\epsilon_i^2 = aw^2 + bw + c$
$V\left(-\frac{b}{2a}, -\frac{\Delta}{4a^2}\right)$
w

Comparison of perceptron and linear regression

Perceptron:

- If $f(\vec{x}) = y$ then do nothing
- If $f(\vec{x}) = 1$ but $y = -1$:
  $$\vec{w} \leftarrow \vec{w} - \eta\vec{x}$$
- If $f(\vec{x}) = -1$ but $y = 1$:
  $$\vec{w} \leftarrow \vec{w} + \eta\vec{x}$$

Linear regression:

- If $\epsilon_i = 0$ then do nothing
- If $\epsilon_i > 0$:
  $$\vec{w} \leftarrow \vec{w} - \eta\epsilon_i\vec{x}_i$$
- If $\epsilon_i < 0$:
  $$\vec{w} \leftarrow \vec{w} - \eta\epsilon_i\vec{x}_i$$

Outline

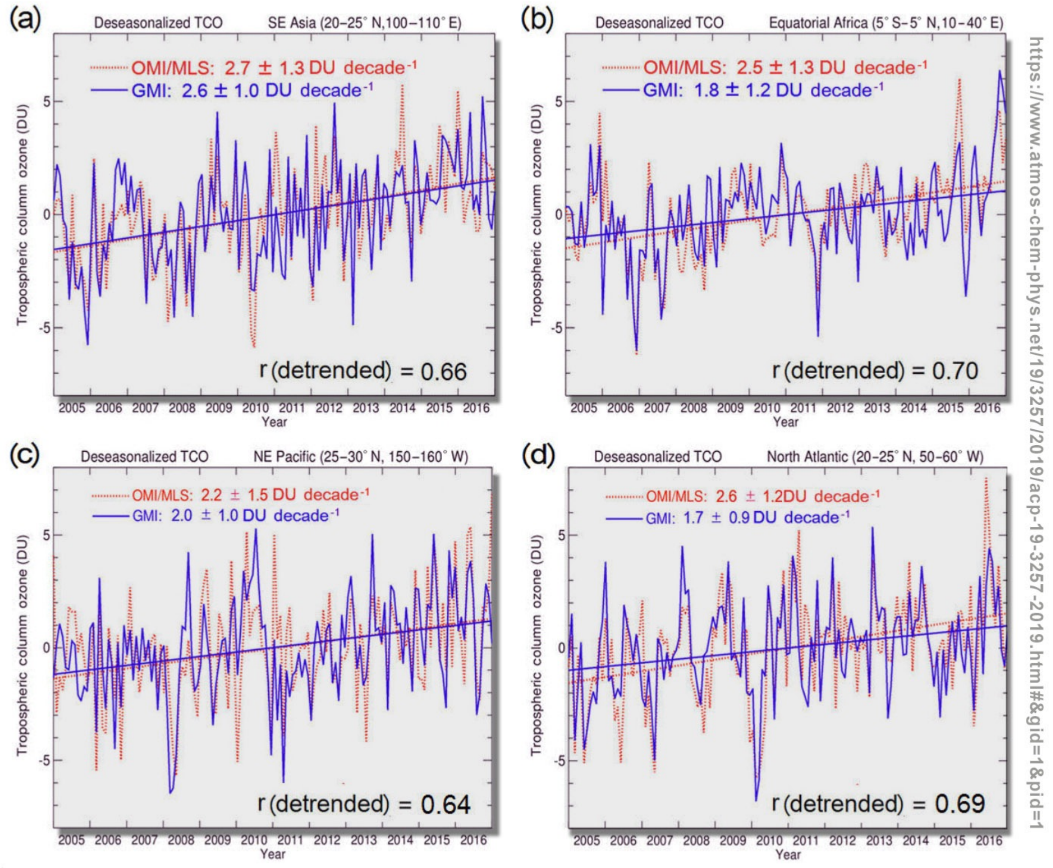
- Review: perceptron
- Linear regression: a neuron without the nonlinearity
- Mean-squared error
- Learning the solution: stochastic gradient descent
- Multiple linear regression

What if $\vec{y}$ is a vector?

Sometimes we want to model a system with many inputs, and many outputs. In that case, y and f(x) both become vectors:

$$\vec{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_V \end{bmatrix}, \qquad f(\vec{x}) = \begin{bmatrix} f_1(\vec{x}) \\ \vdots \\ f_V(\vec{x}) \end{bmatrix}$$

…and our goal is to find a function $f(\vec{x})$ so that $f(\vec{x}) \approx \vec{y}$.



Jerry R. Ziemke et al. (2019) Trends in tropospheric ozone ; Aura record (2005–2016) ; in Trends in global tropospheric ozone inferred from a composite record of TOMS/OMI/MLS/OMPS satellite measurements and the MERRA-2 GMI simulation.

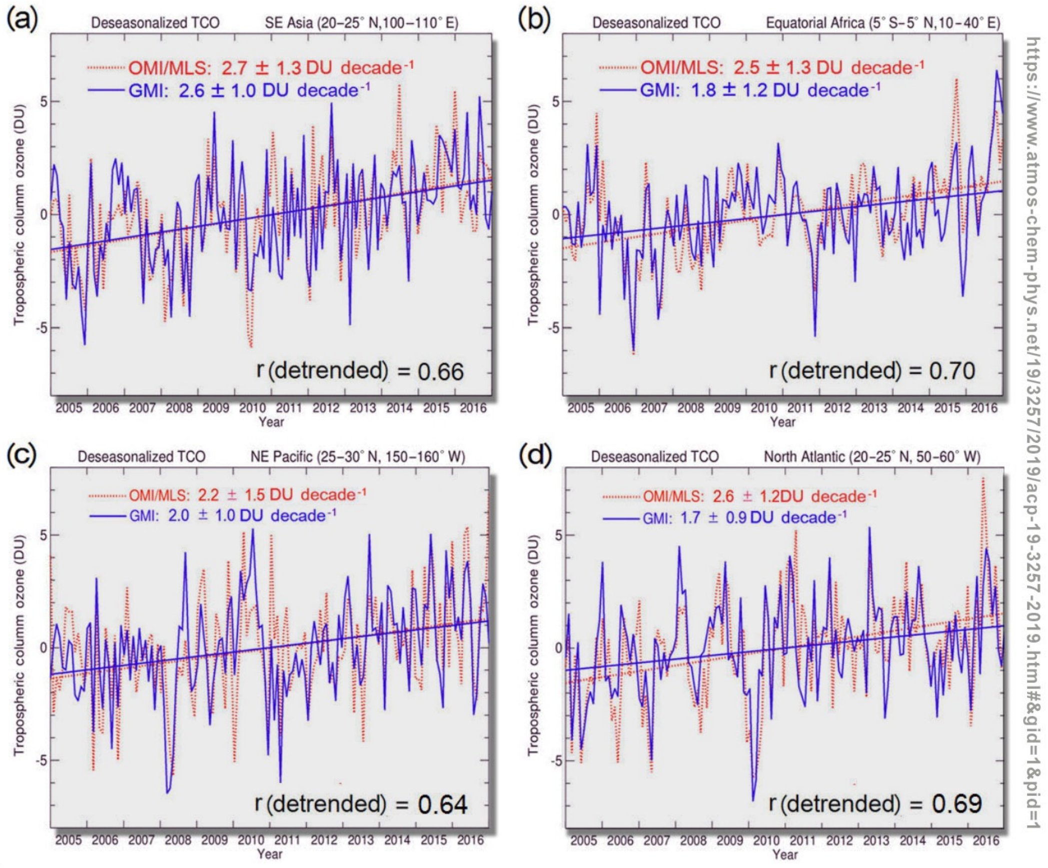
Atmos. Chem. Phys., 19, 3257-3269, 2019 ; https://doi.org/10.5194/acp-19-3257-2019
https://www.atmos-chem-phys.net/19/3257/2019/acp-19-3257-2019.html CC-BY-SA 4.0

CC-SA 4.0, Joseph Lamiot, 2019

What if $\vec{y}$ is a vector?

We can model this using multiple $\vec{w}_c$ vectors, a little like we did for multi-class perceptron:

$$f_c(\vec{x}) = \vec{w}_c^T \vec{x} = b_c + \sum_{j=1}^{D} w_{c,j} x_j$$



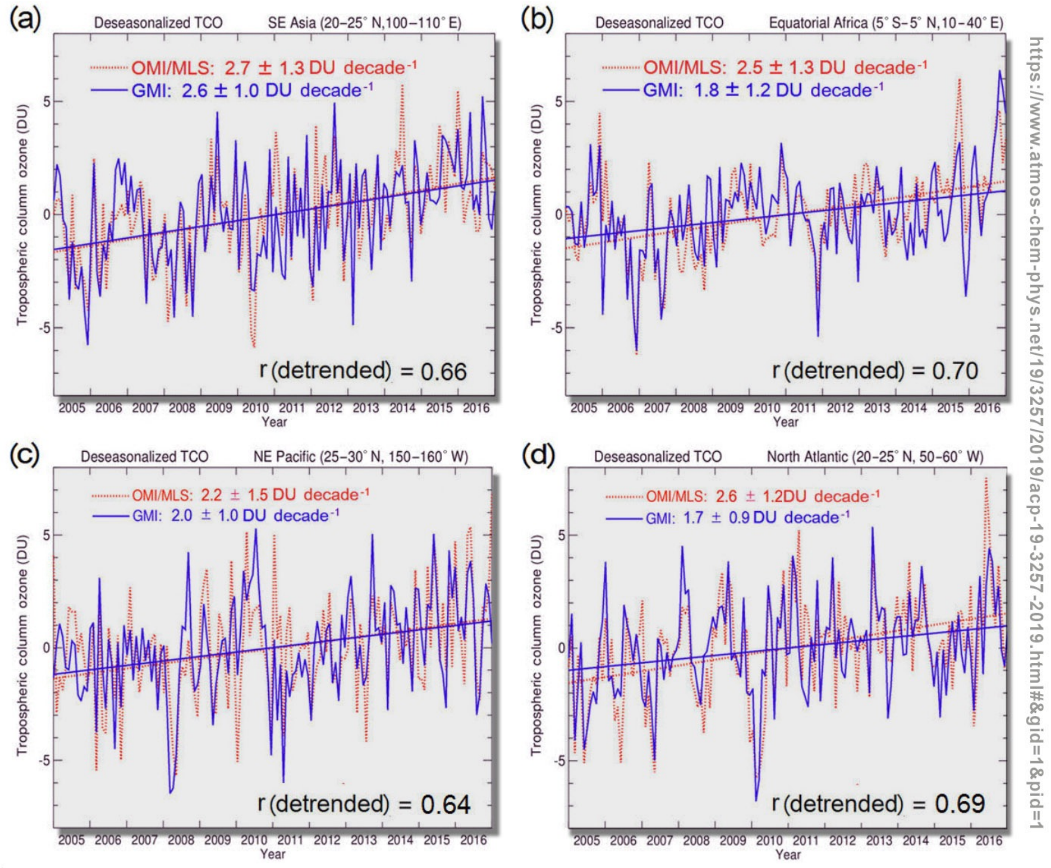
Jerry R. Ziemke et al. (2019) Trends in tropospheric ozone ; Aura record (2005–2016) ; in Trends in global tropospheric ozone inferred from a composite record of TOMS/OMI/MLS/OMPS satellite measurements and the MERRA-2 GMI simulation.

Atmos. Chem. Phys., 19, 3257-3269, 2019 ; https://doi.org/10.5194/acp-19-3257-2019
https://www.atmos-chem-phys.net/19/3257/2019/acp-19-3257-2019.html CC-BY-SA 4.0

… it means $\vec{\epsilon}_i$ is a vector!

The gradient of the error with respect to each of the weights is:

$$\epsilon_{c,i} = f_c(\vec{x}_i) - y_{c,i}$$



Jerry R. Ziemke et al. (2019) Trends in tropospheric ozone ; Aura record (2005–2016) ; in Trends in global tropospheric ozone inferred from a composite record of TOMS/OMI/MLS/OMPS satellite measurements and the MERRA-2 GMI simulation.

Atmos. Chem. Phys., 19, 3257-3269, 2019 ; https://doi.org/10.5194/acp-19-3257-2019
https://www.atmos-chem-phys.net/19/3257/2019/acp-19-3257-2019.html CC-BY-SA 4.0

CC-SA 4.0, Joseph Lamiot, 2019

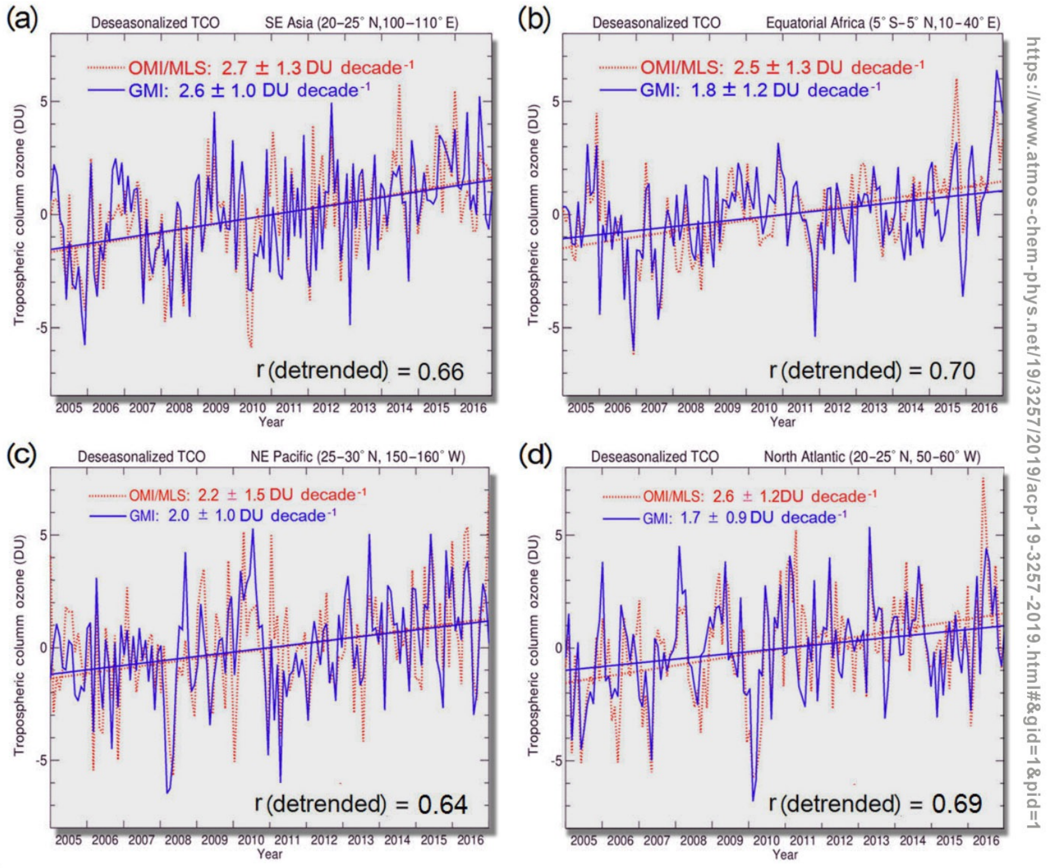
… and each $\vec{w}_c$ has its own gradient.

This means that we have a different error term for each of the weight vectors:

$$\nabla_{\vec{w}_c}\epsilon_{c,i}^2 = 2\epsilon_{c,i}\vec{x}_i$$

… so the stochastic gradient descent update step is

$$\vec{w}_c \leftarrow \vec{w}_c - \eta\epsilon_{c,i}\vec{x}_i$$



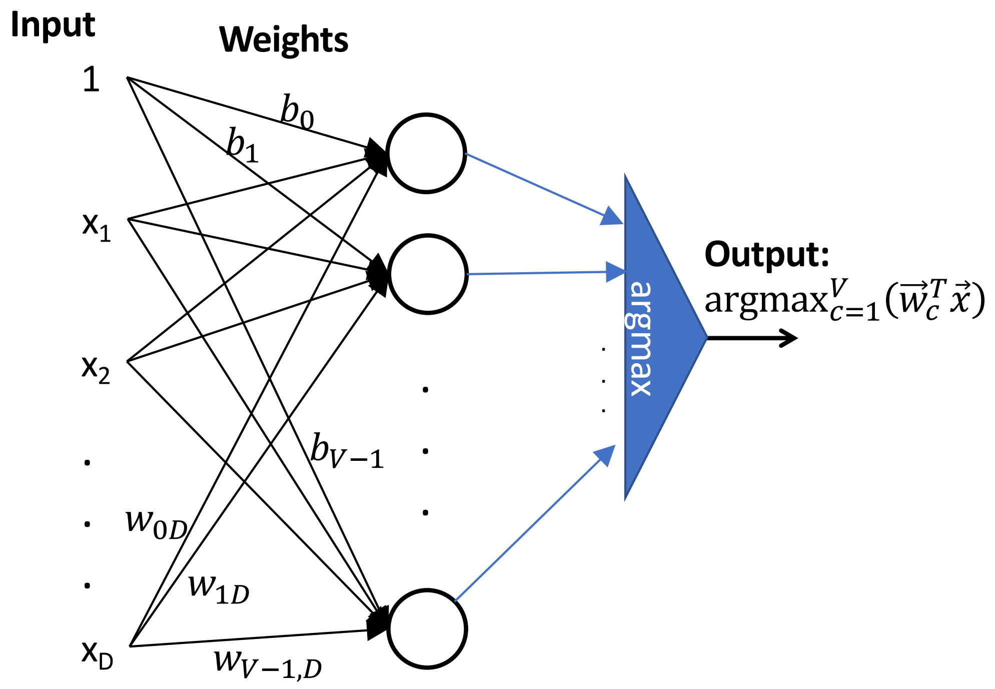
Jerry R. Ziemke et al. (2019) Trends in tropospheric ozone ; Aura record (2005–2016) ; in Trends in global tropospheric ozone inferred from a composite record of TOMS/OMI/MLS/OMPS satellite measurements and the MERRA-2 GMI simulation.

Atmos. Chem. Phys., 19, 3257-3269, 2019 ; https://doi.org/10.5194/acp-19-3257-2019
https://www.atmos-chem-phys.net/19/3257/2019/acp-19-3257-2019.html CC-BY-SA 4.0

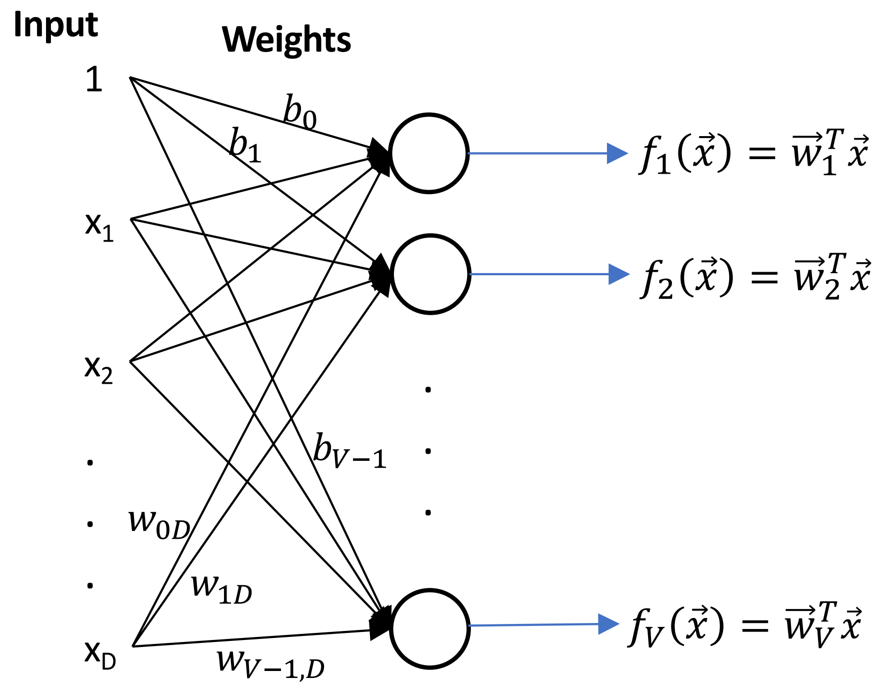
CC-SA 4.0, Joseph Lamiot, 2019

Comparison of Multi-Class Perceptron to Multiple Regression

Multi-Class Perceptron



Multiple Regression


Input
Weights
1
x1
x2
xD
b_0
b_1
b_{V-1}
w_{0D}
w_{1D}
$w_{V-1,D}$
$f_1(\vec{x}) = \vec{w}_1^T\vec{x}$
$f_2(\vec{x}) = \vec{w}_2^T\vec{x}$
$f_V(\vec{x}) = \vec{w}_V^T\vec{x}$

Summary

- Review: perceptron
- Linear regression: a neuron without the nonlinearity

$$f(\vec{x}) = \vec{w}^T \vec{x}$$

- Mean-squared error

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \epsilon_i^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\vec{x}_i^T \vec{w} - y_i\right)^2$$

- Learning the solution: stochastic gradient descent

$$\vec{w} \leftarrow \vec{w} - \frac{\eta}{2}\nabla_{\vec{w}}\epsilon_i^2 = \vec{w} - \eta\epsilon_i\vec{x}_i$$

- Multiple linear regression

$$\vec{w}_c \leftarrow \vec{w}_c - \frac{\eta}{2}\nabla_{\vec{w}_c}\epsilon_{c,i}^2 = \vec{w}_c - \eta\epsilon_{c,i}\vec{x}_i$$