



ISTA 421/521

Introduction to Machine Learning

Lecture 2: Linear Models

Clay Morrison

clayton@sista.arizona.edu

Gould-Simpson 819

Phone 621-6609

28 August 2014



Three General Classes of ML

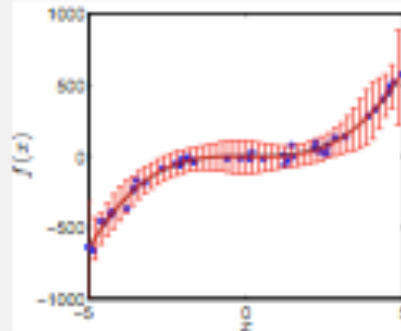
- **Supervised learning** – model $p(y|\mathbf{x})$
 - Given data and model, or data with correct output (label)
 - Regression, Classification, etc.
- **Unsupervised Learning** – model $p(\mathbf{x})$
 - Only given input data (no output)
 - Clustering, Latent Models, Projection methods, etc.
- **Reinforcement Learning** – model $p(\mathbf{s}_{t+1}|\mathbf{s},a)$
 - Given input data, *some* output, and *grade* for output
 - Learning to choose better actions
 - Markov decision processes, POMDPs, planning



Supervised Learning

Regression

Learning a continuous function from a set of examples.



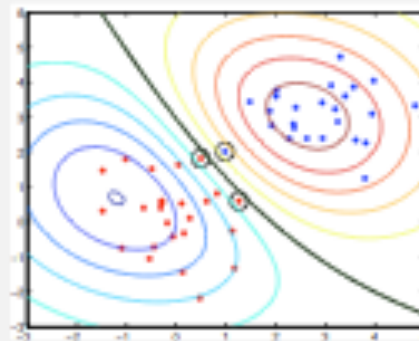
Example

Predicting stock prices (x might be time or some other variable of interest).

Supervised Learning

Classification

Learning a rule that can separate objects of different types from one another.



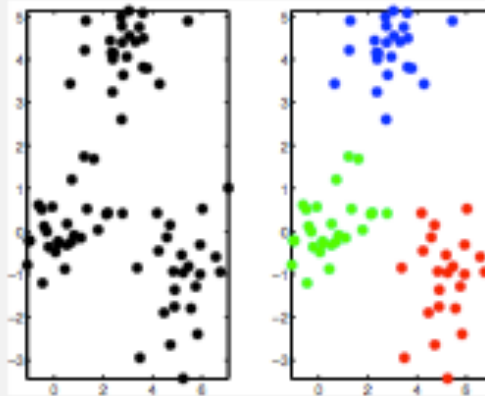
Examples

Disease diagnosis, spam email detection.

Unsupervised Learning

Clustering

Finding groups of similar objects.



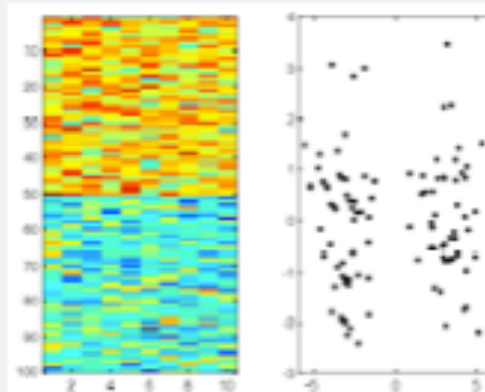
Examples

People with similar 'taste', genes with similar function.

Unsupervised Learning

Projection

Reducing the number of variables – e.g. from 10 to 2.

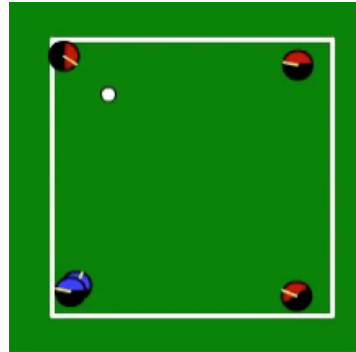


Examples

Visualising complex data.

Reinforcement Learning

- Example: Simulated Robot Soccer Keepaway
- Input: angles, distances, clock



- What is the best next action to take given what I know now?
- Allows system to learn by interacting with environment with human only providing tips about “goodness”

Topics

- The linear model
 - Regression, Classification
- Classification
 - Probabilistic:
 - Bayes Classifier, Naïve Bayes
 - Logistic Regression
 - Other, *non*-probabilistic
 - K-nearest neighbors
 - Support Vector Machines and kernel methods
- Clustering
 - K-means
 - Mixture Models and EM
- Other Unsupervised methods:
 - Principle Components Analysis
 - Latent Variable Models
- Additional topics (time permitting)
 - Neural networks, Deep networks
 - Ensemble methods, Boosting
 - Gaussian processes
- Probability
 - Quantifying uncertainty
 - Bayesian Approach: Prior, Marginal Likelihood, MAP
- Inference Methods
 - Least Squares
 - Maximum Likelihood
 - Bayesian Inference: Direct and Sampling
- Machine Learning algorithm evaluation
- Learning theory
- Feature Selection and Model Selection

Today

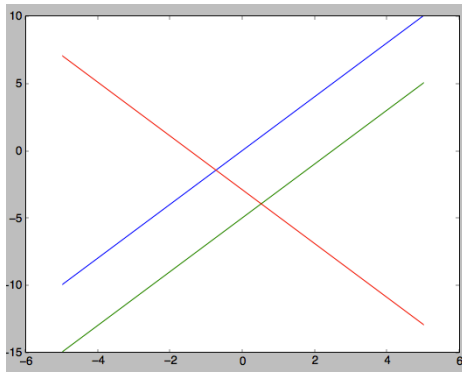
- Homework 1
- A simple learning problem
- Linear Model (what is the *model*)
- Loss Function (what is a *good* model)
- Least Squares (finding the *best* model)
- Prediction
- Moving to higher dimensions

Homework 1

- **Goal:** Get comfortable with your programming environment and some written exercises.
- Three parts:
 - Exercises: 1.1, 1.3, 1.4, 1.5
 - Use script, plotlinear[.m/.py], to plot some lines! Generate the plot and put it in your pdf submission with a caption.
 - I give you a matrix that I will ask you to use code to transpose and take the matrix inverse. You need to include in your pdf a neat representation of the original matrix, and the two resulting matrices, and include the code you used to generate the matrix.
- I will post these written instructions on D2L and let everyone know when it's available.
- **DUE:** Next Friday, Sept 5, through D2L dropbox
- Worth 8 points

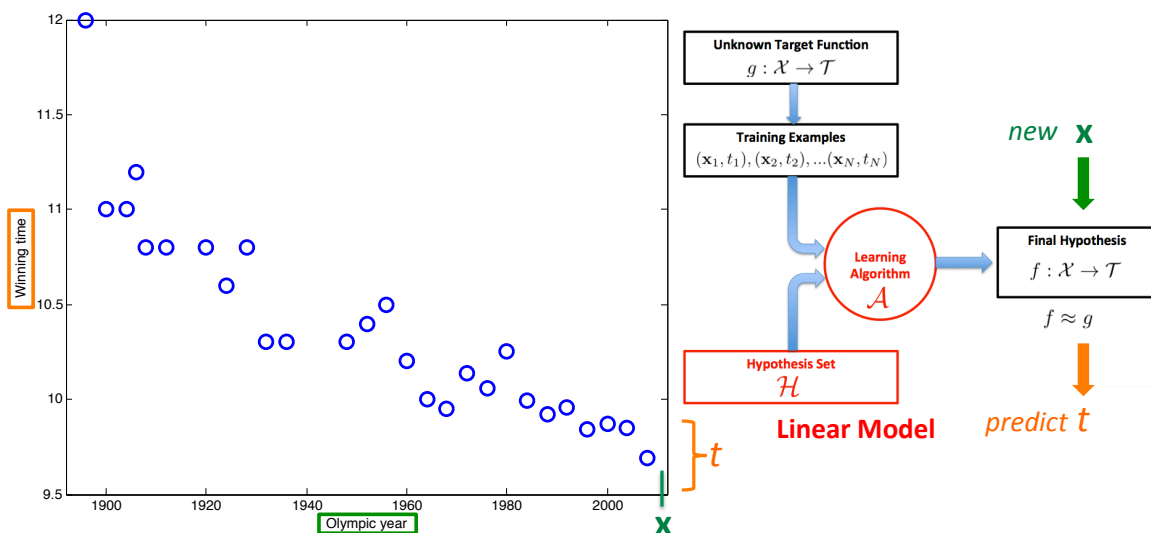
Plotlinear.py

- need matplotlib



```
15 Keeps plotting lines on the current plot until you quit (Ctrl-D)
16
17 Enter intercept: 0
18 Enter gradient (slope): 2
19
20 y = 0.0 + 2.0 x
21
22
23 Enter intercept: -5
24 Enter gradient (slope): 2
25
26 y = -5.0 + 2.0 x
27
28
29 Enter intercept: -3
30 Enter gradient (slope): -2
31
32 y = -3.0 + -2.0 x
33
```

A simple learning problem



Want to describe **Winning time (t)** as a function of **Olympic year (x)**

Defining a Model

- Define function that maps inputs (Olympics year, x_i) to output or target values (Winning times, t_i)

$$t = f(x)$$

- The model itself likely has **parameters**, which we'll generically refer to as ' α ' here. It is common to make them explicit within a function:

$$t = f(x; \alpha)$$

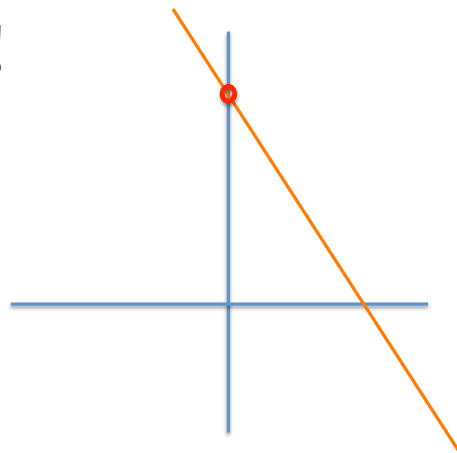
Lines!

- Slope-intercept form**

$$y = \underline{m}x + \underline{b}^*$$

- General (standard) form**

$$ax + by + c = 0$$



$$\text{slope } m = -\frac{a}{b}$$

$$\text{y-intercept } b^* = -\frac{c}{b}$$

$$\text{x-intercept} = -\frac{c}{a}$$

Lines!

- General (standard) form

$$\underline{a}x + \underline{b}y + \underline{c} = 0$$

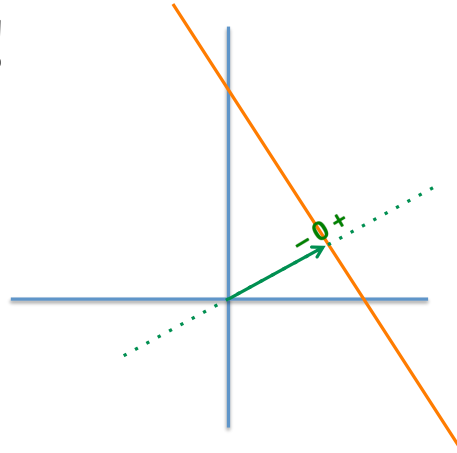
- Slope-intercept form

$$y = \underline{m}x + \underline{b}^*$$

$$\text{slope } m = -\frac{a}{b}$$

$$\text{y-intercept } b^* = -\frac{c}{b}$$

$$\text{x-intercept} = -\frac{c}{a}$$



Linear Models

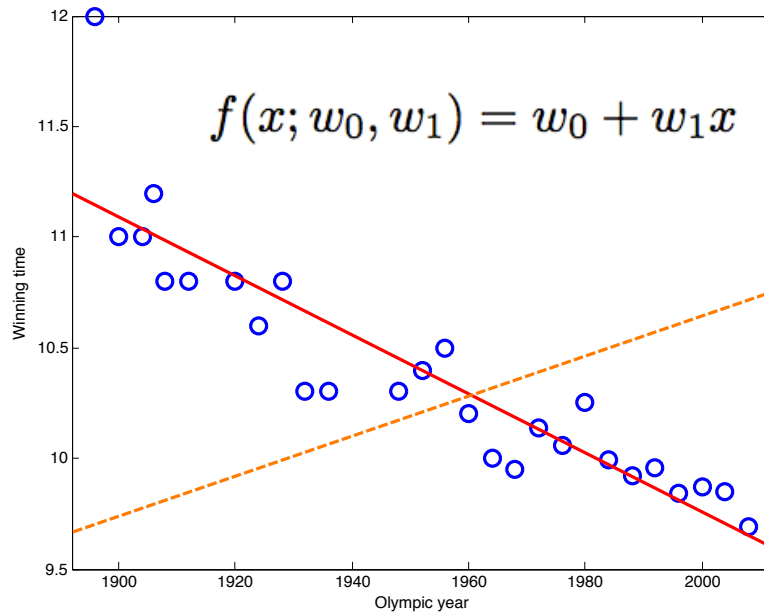
- $y = mx + b$ (or $t = w_1x + w_0$)
 - the classic line (in 2D space)
 - For a given line, m and b are the **parameters** and x is a **variable** in the relationship:

$$y = f(x; m, b)$$
 - When considering alternate lines, we are adjusting m and b
- Generally, as long as the variables are not themselves involved in anything more than **addition** and **scalar multiplication**, then the relationship is **linear**.

$$y = mx^2 + c \quad y = \sin(x) \quad \sqrt{y} = mx + c \quad \text{Not linear rel. btwn } x, y$$

$$y = mx + c^2 \quad y = x \sin(m) + c \quad \text{Is linear rel. btwn } x, y \text{ (but not parameters!)}$$

Data with line (particular w_0 & w_1)



(The red line happens to be a “best” fit)

(The dashed orange line does not describe the trend in the data very well; not a good fit)

17

Loss Function

$$\mathcal{L}_n()$$

Squared Error:

$$(t_n - f(x_n; w_0, w_1))^2$$

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

Mean Squared Error:

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

Goal: Find the “best” values for the parameters of model according to the loss fn

- If we’re lucky (i.e., the model and the loss fn are “well-behaved”) we can derive an **analytic** solution. Otherwise, we’ll pick some (iterative) optimization method that is appropriate.
- Our first example, using a **mean squared error loss function** with a **linear model** permits a nice analytic solution!
 - Here (and in the book) we’ll first look at the direct, analytic method.
 - Another method: gradient descent
 - Same loss function, but iterative algorithm and can be used in cases where we don’t have an analytic solution for the parameters

Least Mean Squares Solution

(for single variable, 2 parameter linear model)

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

$$\begin{aligned}\mathcal{L} &= \frac{1}{N} \sum_{n=1}^N \mathcal{L}_n(t_n, f(x_n; w_0, w_1)) \\ &= \frac{1}{N} \sum_{n=1}^N (t_n - f(x_n; w_0, w_1))^2 && \text{The specific loss fn we're working with here} \\ &= \frac{1}{N} \sum_{n=1}^N (t_n - (w_0 + w_1 x_n))^2 && \text{The specific model we're working with here} \\ &= \frac{1}{N} \sum_{n=1}^N (w_1^2 x_n^2 + 2w_1 x_n w_0 - 2w_1 x_n t_n + w_0^2 - 2w_0 t_n + t_n^2) && \text{Multiply out and re-arrange to put into an easier-to deal with form.} \\ &= \frac{1}{N} \sum_{n=1}^N (w_1^2 x_n^2 + 2w_1 x_n (w_0 - t_n) + w_0^2 - 2w_0 t_n + t_n^2)\end{aligned}$$

Least Mean Squares Solution

(for single variable, 2 parameter linear model)

$$f(x; w_0, w_1) = w_0 + w_1 x \quad \leftarrow \text{Our model family}$$

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^N (w_1^2 x_n^2 + 2w_1 x_n (w_0 - t_n) + w_0^2 - 2w_0 t_n + t_n^2) \quad \leftarrow \text{Our loss fn}$$

Our goal: We want values for w_0 and w_1 that will **minimize** this loss function

i.e., we seek values for w_0 and w_1 that will make the loss function be the smallest when we actually sum over all the values of x and t in the dataset.

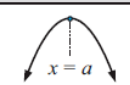
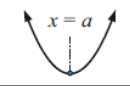
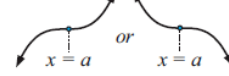
Because the loss function happens to be quadratic (in the two parameters) we can use a standard method from calculus for finding minima (maxima) directly: **taking the derivative of the function and setting it to zero**.

Our loss function has **two** parameters that we're trying set to minimize the loss fn, so we need to take the **partial derivative** (w.r.t. w_0 and w_1)

What we end up with are two functions, one for w_0 and one for w_1 , and both will work with **any** data and give the best **least mean square** (LMS) fit!

Side note: One way to tell we have a **unique** extreme

First and Second Derivative
around particular x value ($x=a$)

Stationary point	Sign diagram of $f'(x)$ near $x = a$	Shape of curve near $x = a$
local maximum	$\leftarrow + \mid - \rightarrow$ a	
local minimum	$\leftarrow - \mid + \rightarrow$ a	
horizontal inflection	$\leftarrow + \mid + \rightarrow$ or $\leftarrow - \mid - \rightarrow$ a a	

Second derivative

Constant negative
(if f is quadratic)

Constant positive
(if f is quadratic)

No longer a constant fn
(if f higher order than quad)

Least Mean Squares Solution

(for single variable, 2 parameter linear model)

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

- Partial derivative for w_0

First, since we're taking the partial w.r.t. w_0 , can drop any terms without w_0 .

$$\frac{1}{N} \sum_{n=1}^N [w_0^2 + 2w_1 x_n w_0 - 2w_0 t_n]$$

Next, move sums inward to put in easier form

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

Finally, take deriv. w.r.t w_0

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

Continued...

- Solve for $\frac{\partial \mathcal{L}}{\partial w_0} = 0$

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

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

$$w_0 = \frac{1}{N} \left(\sum_{n=1}^N t_n \right) - w_1 \frac{1}{N} \left(\sum_{n=1}^N x_n \right) = \bar{t} - w_1 \bar{x}$$

Least Mean Squares Solution

(for single variable, 2 parameter linear model)

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

- Partial derivative for w_1 Do the same for w_1 ...

$$\frac{1}{N} \sum_{n=1}^N [w_1^2 x_n^2 + 2w_1 x_n w_0 - 2w_1 x_n t_n] \quad \text{only keep terms with } w_1$$

$$w_1^2 \frac{1}{N} \left(\sum_{n=1}^N x_n^2 \right) + 2w_1 \frac{1}{N} \left(\sum_{n=1}^N x_n (w_0 - t_n) \right) \quad \text{move sums inside}$$

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

$$= w_1 \frac{2}{N} \left(\sum_{n=1}^N x_n^2 \right) + \frac{2}{N} \left(\sum_{n=1}^N x_n \left(\overbrace{\bar{t} - w_1 \bar{x}}^{\text{solution for } w_0} - t_n \right) \right) \quad \text{plug in solution for } w_0$$

$$= w_1 \frac{2}{N} \left(\sum_{n=1}^N x_n^2 \right) + \bar{t} \frac{2}{N} \left(\sum_{n=1}^N x_n \right) - w_1 \bar{x} \frac{2}{N} \left(\sum_{n=1}^N x_n \right) - \frac{2}{N} \left(\sum_{n=1}^N x_n t_n \right) \quad \dots \text{ and rearrange terms}$$



25

- Partial derivative for w_1 continued...

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

$$= 2w_1 \left[\left(\frac{1}{N} \sum_{n=1}^N x_n^2 \right) - \bar{x} \bar{x} \right] + 2\bar{t} \bar{x} - 2 \frac{1}{N} \left(\sum_{n=1}^N x_n t_n \right) \quad \text{replace remaining mean x with } \bar{x} \text{ and group } w_1 \text{ terms}$$

$$2w_1 \left[\left(\frac{1}{N} \sum_{n=1}^N x_n^2 \right) - \bar{x} \bar{x} \right] + 2\bar{t} \bar{x} - 2 \frac{1}{N} \left(\sum_{n=1}^N x_n t_n \right) = 0 \quad \text{Solve for } w_1 \text{ with } \frac{\partial \mathcal{L}}{\partial w_1} = 0 \dots$$

$$2w_1 \left[\left(\frac{1}{N} \sum_{n=1}^N x_n^2 \right) - \bar{x} \bar{x} \right] = 2 \frac{1}{N} \left(\sum_{n=1}^N x_n t_n \right) - 2\bar{t} \bar{x}$$

$$w_1 = \frac{\frac{1}{N} \left(\sum_{n=1}^N x_n t_n \right) + \bar{t} \bar{x}}{\left(\frac{1}{N} \sum_{n=1}^N x_n^2 \right) - \bar{x} \bar{x}} = \frac{\left(\frac{1}{N} \sum_{n=1}^N x_n t_n \right) - \left(\frac{1}{N} \sum_{m=1}^N t_n \right) \left(\frac{1}{N} \sum_{m=1}^N x_n \right)}{\left(\frac{1}{N} \sum_{n=1}^N x_n^2 \right) - \left(\frac{1}{N} \sum_{n=1}^N x_n \right)^2}$$

Solving LMS: Method 1 (analytic)

(for single variable, 2 parameter linear model)

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

- Partial derivative for w_0 :

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

- Set $\frac{\partial \mathcal{L}}{\partial w_0} = 0$ and solve for w_0 :

$$w_0 = \frac{1}{N} \left(\sum_{n=1}^N t_n \right) - w_1 \frac{1}{N} \left(\sum_{n=1}^N x_n \right) = \bar{t} - w_1 \bar{x}$$

The so-called
normal equations!

- Partial derivative for w_1 :

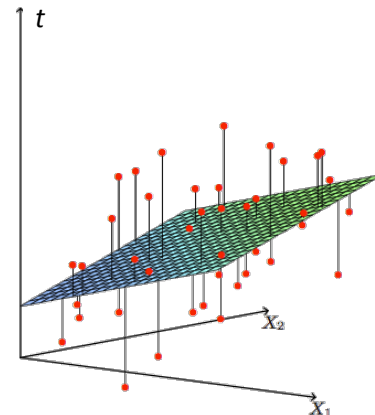
$$\frac{\partial \mathcal{L}}{\partial w_1} = 2w_1 \frac{1}{N} \left(\sum_{n=1}^N x_n^2 \right) + \frac{2}{N} \left(\sum_{n=1}^N x_n (w_0 - t_n) \right)$$

- Plug in w_0 , set $\frac{\partial \mathcal{L}}{\partial w_1} = 0$ and solve for w_1 :

$$w_1 = \frac{\left(\frac{1}{N} \sum_{n=1}^N x_n t_n \right) - \left(\frac{1}{N} \sum_{m=1}^N t_n \right) \left(\frac{1}{N} \sum_{m=1}^N x_n \right)}{\left(\frac{1}{N} \sum_{n=1}^N x_n^2 \right) - \left(\frac{1}{N} \sum_{n=1}^N x_n \right)^2} = \frac{\overline{xt} - \bar{x}\bar{t}}{\overline{x^2} - (\bar{x})^2}$$

How about more than 1 input?

- Most problems will involve more than just the relationship between 1 input attribute and a target.
- Extending our linear models to higher dimensions is desirable, and at least for 2 inputs (now the “line” is a plane in 3D) it is easy to visualize the geometry.
- In general, a linear model with n input variables and $n+1$ parameters (the w 's, with their values determined) is an n -dimensional “hyperplane” embedded in $n+1$ dimensions.



Things quickly get messy as we increase the dimensions...

- Suppose we want a richer predictive model for the Olympic data: not only include the best overall time for the gold, but also the best times of each sprinter that raced (s_1, \dots, s_8)
- This is a 9 dimensional hyperplane with 10 parameters:

$$\begin{aligned} t = f(x, s_1, \dots, s_8; w_0, \dots, w_9) = & w_0 + w_1x + w_2s_1 + w_3s_2 \\ & + w_4s_3 + w_5s_4 + w_6s_5 \\ & + w_7s_6 + w_8s_7 + w_9s_8 \end{aligned}$$

- The math is fundamentally the same, but to derive the normal equations, we need to take 10 partial derivatives, then have 10 equations to re-arrange and substitute back in...