CPSC 340: Machine Learning and Data Mining

Linear Least Squares

Admin

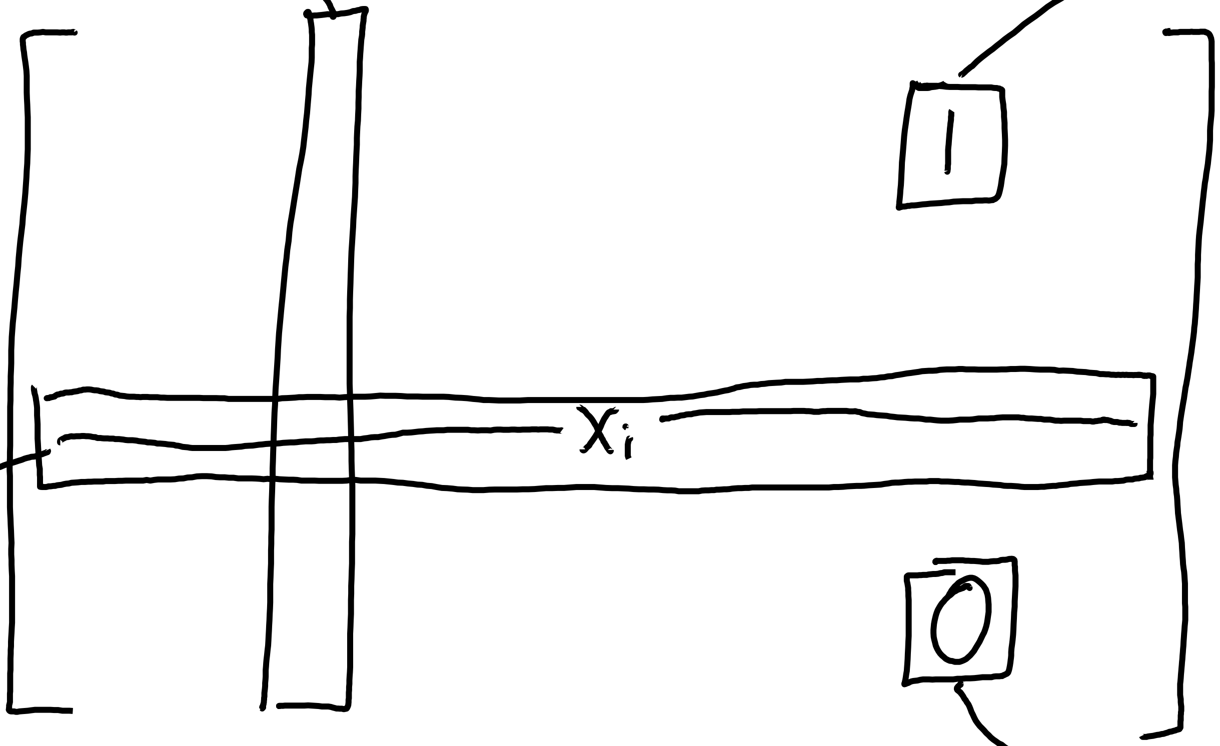
- Assignment 2 is due Sunday:
  - You should already be started!
- Regarding GitHub
  - I got a comment about late days:
  - We use push timestamps not commit timestamps

User-Product Matrix

Column gives <u>all users</u> that bought product.

$X_{ij} = 1$ means user 'i' bought item 'j'.

$X =$

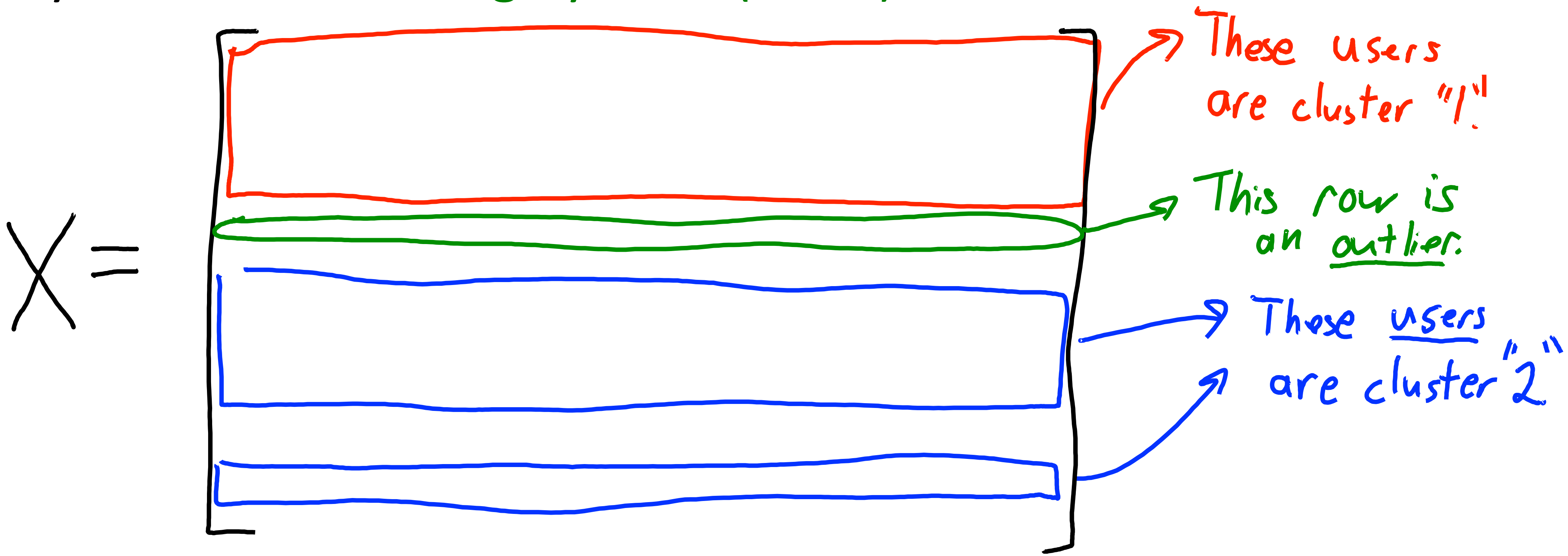


$X_{ij} = 0$ means user 'i' did <u>not</u> buy item 'j'

Row x_i gives <u>all items</u> bought by user 'i'. By convention, x_i is a $d \times 1$ column vector.

Clustering User-Product Matrix

- Normally think of clustering by rows (users):

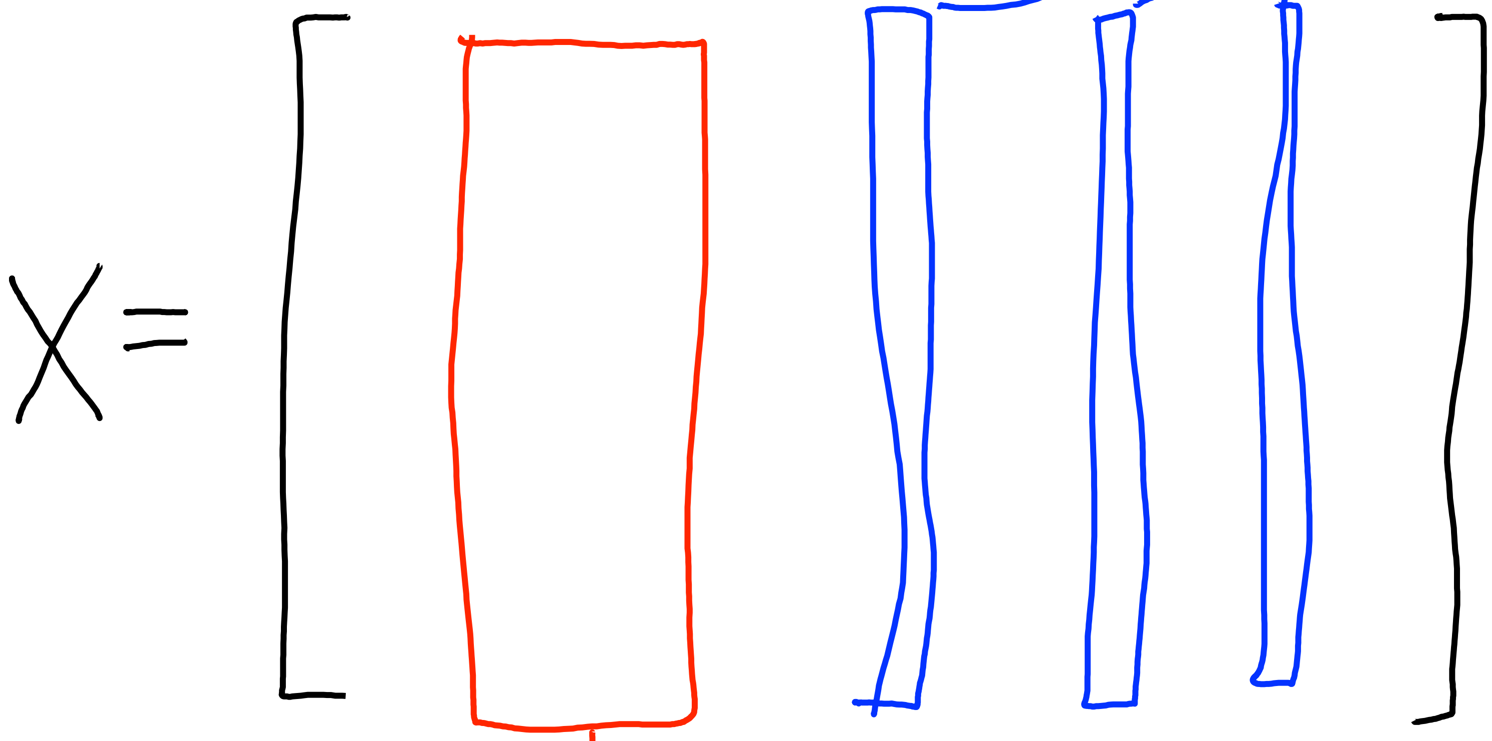


- We also find outliers by rows.

Clustering User-Product Matrix

- We could cluster by columns (products):

These products are cluster "2"

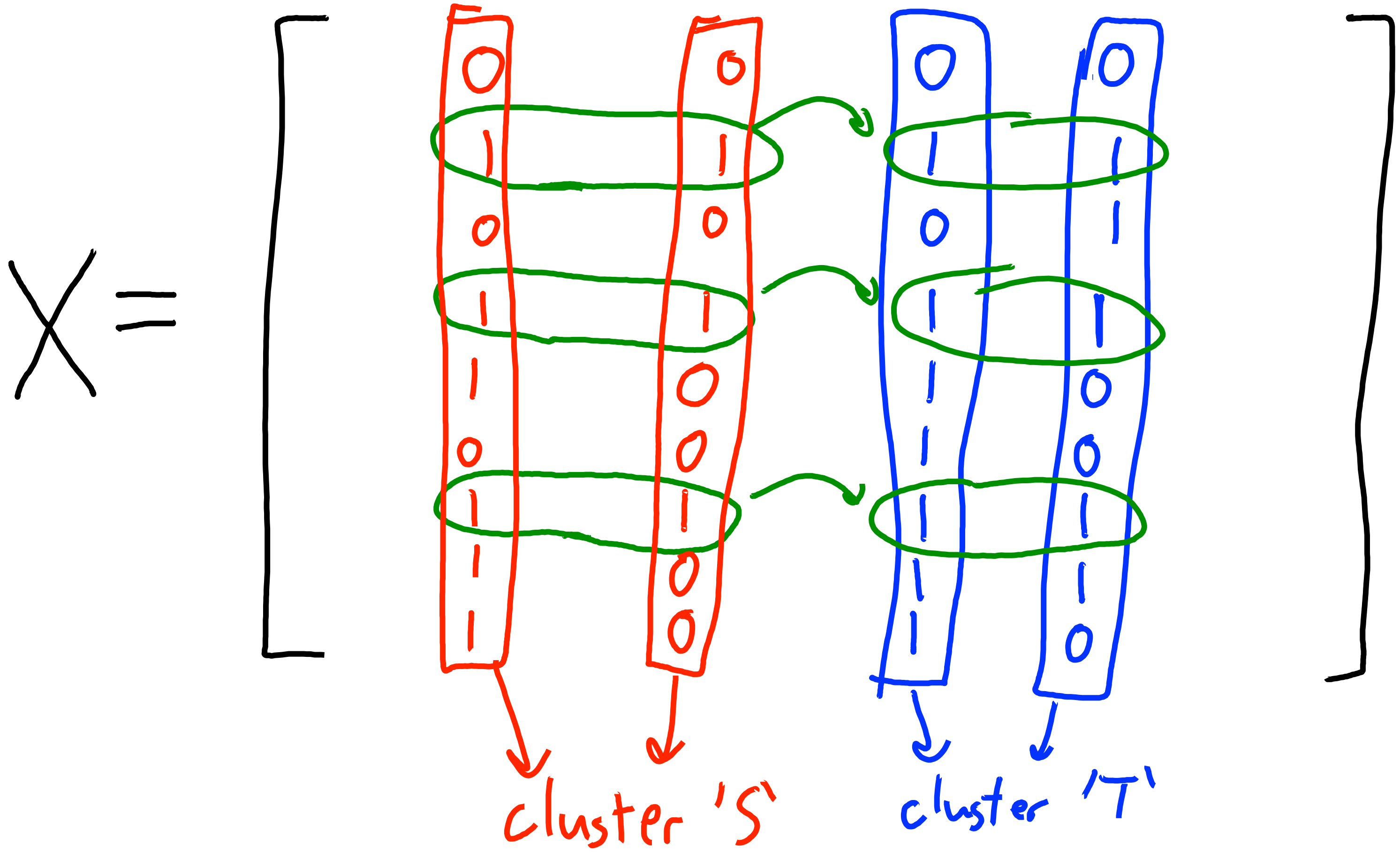


These products are cluster "1"

- Apply clustering to X^T.

Association Rules

- Association rules (S => T): all ‘1’ in cluster S => all ‘1’ in cluster T.



Amazon Product Recommendation

- Amazon product recommendation works by columns:
  - Conceptually, you take the user-product matrix:

$$X = \begin{bmatrix} & & & \\ & & & \\ & 1 & & \end{bmatrix}$$

user 'i' bought item 'j'

  - And transpose it to make a product-user matrix:

$$X^T = \begin{bmatrix} & & 1 & \\ & & & \\ & & & \end{bmatrix}$$

product 'i' was bought by user 'j'

  - Find similar products as nearest neighbours among products.
    - Cosine similarity used as "distance".

End of Part 2: Key Concepts

- We focused on 3 unsupervised learning tasks:
  - Clustering.
    - K-means algorithm (and using it for vector quantization).
    - Density-based clustering (and region-based pruning for finding close points).
    - Hierarchical clustering (and agglomerative algorithm for constructing trees).
  - Outlier Detection.
    - Surveyed common approaches (and said that problem is ill-defined).
  - Association rules.
    - A priori algorithm (for finding rules with high support and confidence).
    - Amazon product recommendation (for huge datasets).

Supervised Learning Round 2: Regression

- We're going to revisit supervised learning:

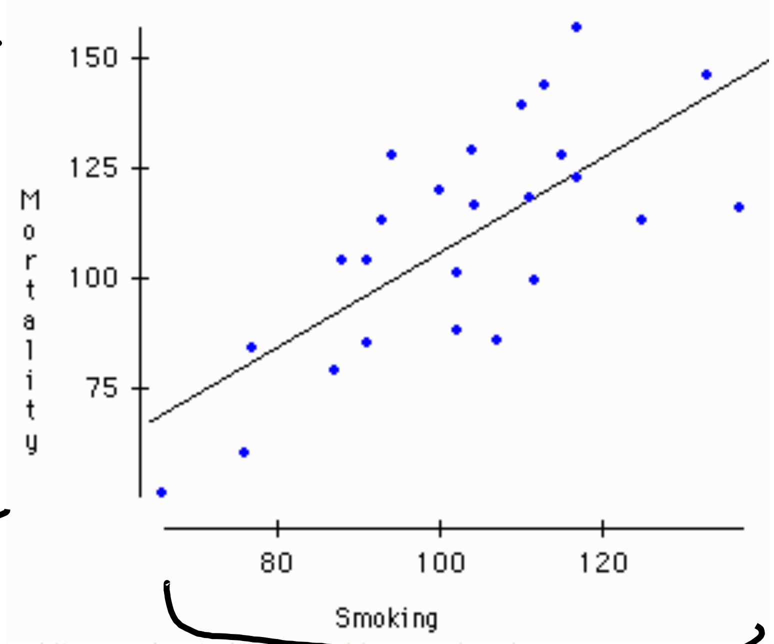
$$X = \begin{bmatrix} \quad \end{bmatrix} \qquad y = \begin{bmatrix} \quad \end{bmatrix}$$

- Previously, we considered classification:
  - We assumed y_i was discrete: y_i = 'spam' or y_i = 'not spam'.
- Now we're going to consider regression:
  - We allow y_i to be numerical: $y_i = 10.34$cm.

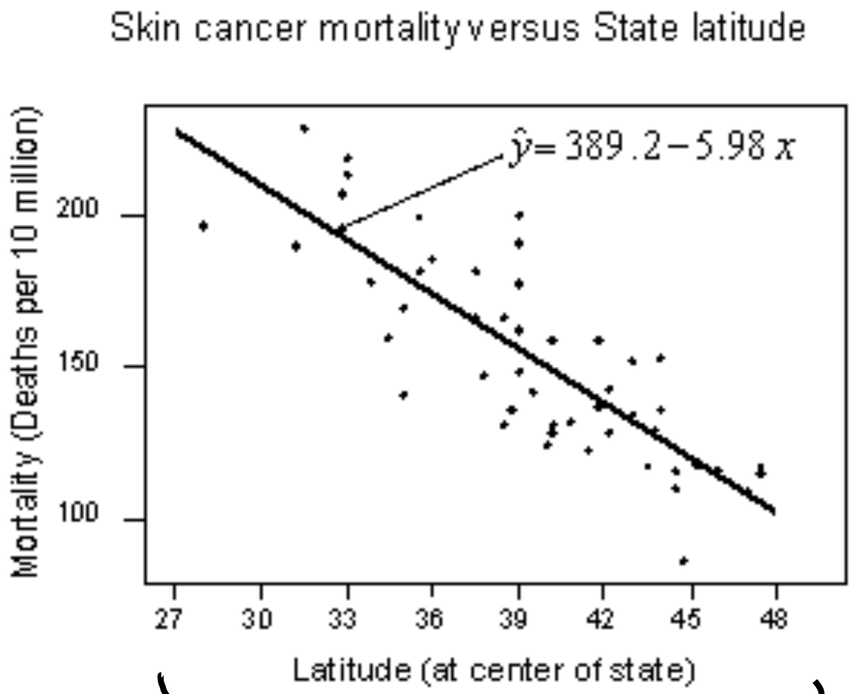
Example: Dependent vs. Explanatory Variables

- We want to discover relationship between numerical variables:
  - Does number of lung cancer deaths change with number of cigarettes?
  - Does number of skin cancer deaths change with latitude?

y_i



x_i



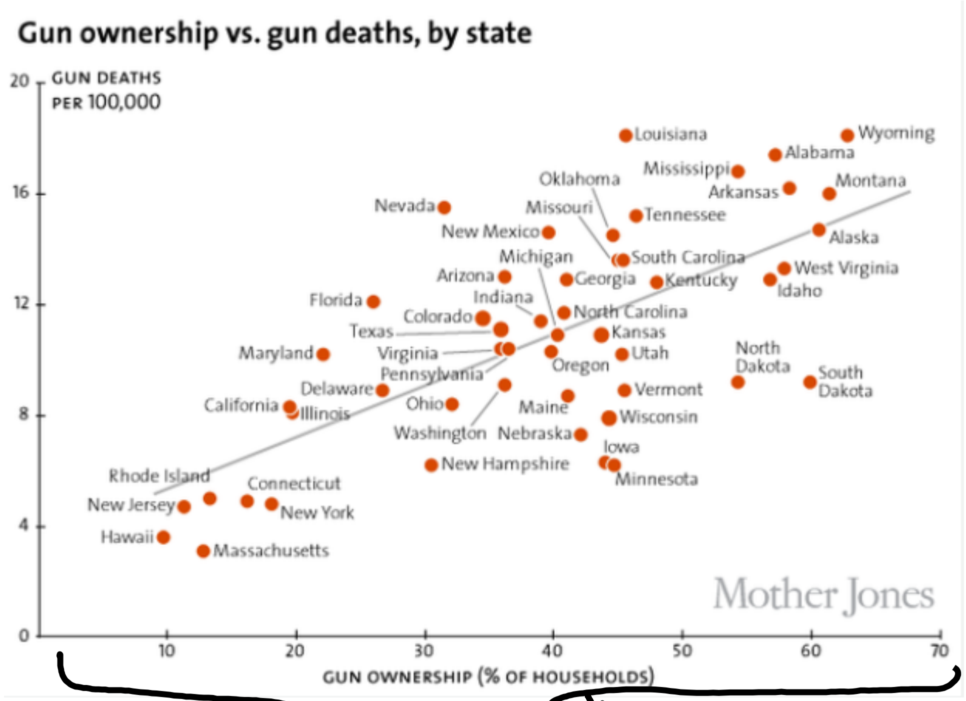
y_i

x_i

http://www.cvgs.k12.va.us:81/digstats/main/inferant/d_regrs.html
https://onlinecourses.science.psu.edu/stat501/node/11

Example: Dependent vs. Explanatory Variables

- We want to discover relationship between numerical variables:
  - Does number of lung cancer deaths change with number of cigarettes?
  - Does number of skin cancer deaths change with latitude?
  - Does number of gun deaths change with gun ownership?



y_i

x_i

http://www.vox.com/2015/10/3/9444417/gun-violence-united-states-america

Handling Numerical Labels

- One way to handle numerical y_i: discretize.
  - E.g., for ‘age’ could we use {‘age ≤ 20’, ‘20 < age ≤ 30’, ‘age > 30’}.
  - Now we can apply methods for classification to do regression.
  - But coarse discretization loses resolution.
  - And fine discretization requires lots of data.
  - We also discard ordering information.
- We could make regression versions of classification methods:
  - Next time: regression trees, generative models, non-parametric models.
- Today: one of oldest, but still most popular/important methods:
  - Linear regression based on squared error.
  - Very interpretable and the building block for more-complex methods.

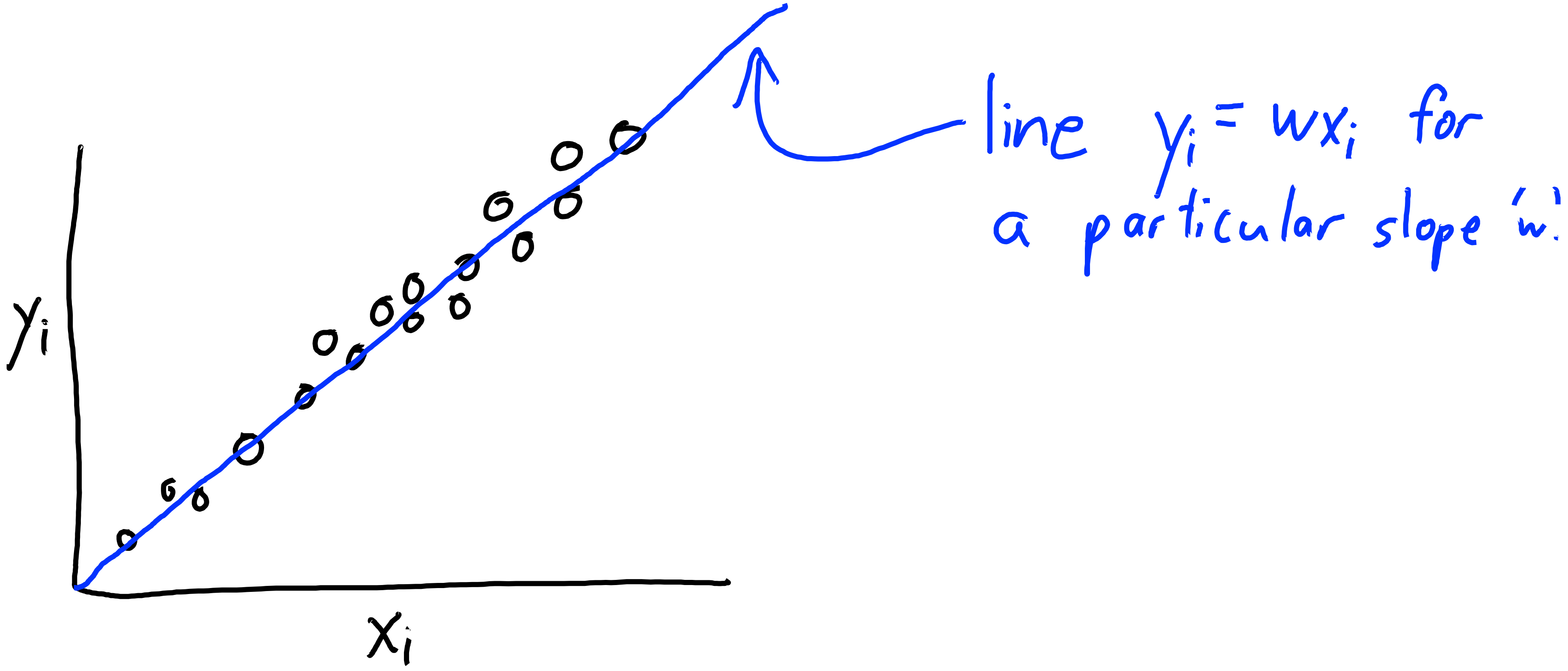
Linear Regression in 1 Dimension

- Assume we only have 1 feature (d = 1):
  - E.g., x_i is number of cigarettes and y_i is number of lung cancer deaths.
- Linear regression models y_i is a linear function of x_i:

$$y_i = w x_i$$

- The parameter 'w' is the weight or regression coefficient of x_i.
- As x_i changes, slope 'w' affects the rate that y_i increases/decreases:
  - Positive 'w': y_i increase as x_i increases.
  - Negative 'w': y_i decreases as x_i increases.

Linear Regression in 1 Dimension



Aside: terminology woes

- Different fields use different terminology and symbols.
  - “data points” = “**objects**” = “**examples**” = “rows”
  - "**inputs**" = "predictors" = "**features**" = "explanatory variables" = "regressors" = "independent variables" = "covariates” = “columns”
  - "**outputs**" = "outcomes" = "targets" = "response variables" = "dependent variables” (also called a "label" if it's categorical)
  - "regression coefficients" = "**weights**" = "parameters”
- With linear regression, the symbols are inconsistent too
  - In ML, the data is X and the weights are w
  - In Statistics, the data is X and the weights are β
  - In optimization, the data is A and the weights are x

Least Squares Objective

- Our linear model is given by:

$$y_i = w x_i$$

- So we make predictions for a new example by using:

$$\hat{y}_i = w \hat{x}_i$$

- But we can't use the same error as before:
  - Even if data <u>comes from a linear model</u> but has noise, we can have $\hat{y}_i \neq y_i$ for <u>all</u> training examples 'i' for the "best" model

Least Squares Objective

- We need a way to evaluate numerical error.
- Classic way to set slope ‘w’ is minimizing sum of squared errors:

$$f(w) = \sum_{i=1}^{n} (wx_i - y_i)^2$$

True value of y_i

Our prediction of y_i

Sum up the squared differences over all training examples.

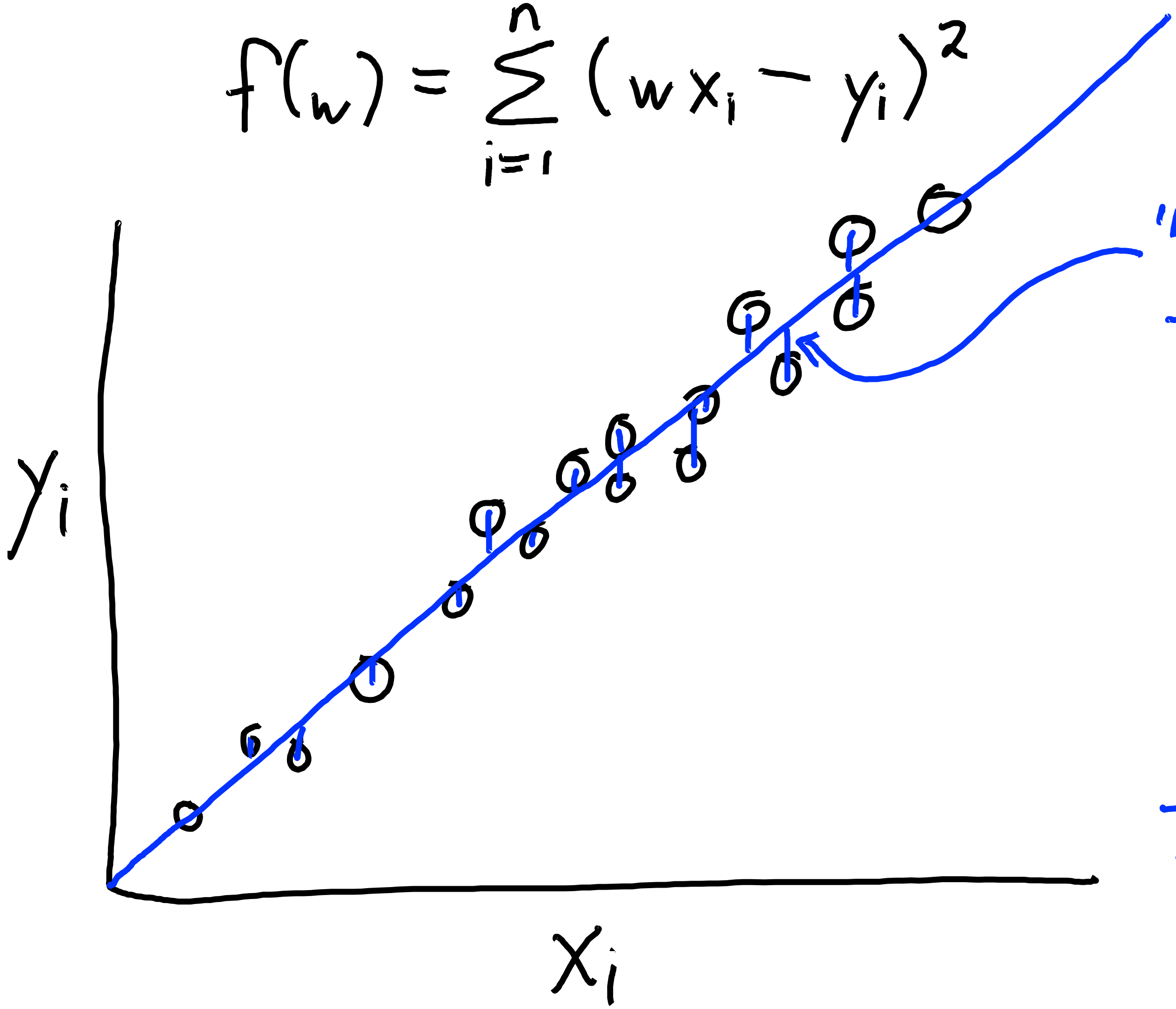
Difference between prediction and true value for example 'i'.

- There are some justifications for this choice.
  – Assuming errors are Gaussian and finding w by maximum likelihood.
- But usually, it is done because it is easy to minimize.

Least Squares Objective

- Classic way to set slope 'w' is minimizing sum of squared errors:

$$f(w) = \sum_{i=1}^{n} (w x_i - y_i)^2$$



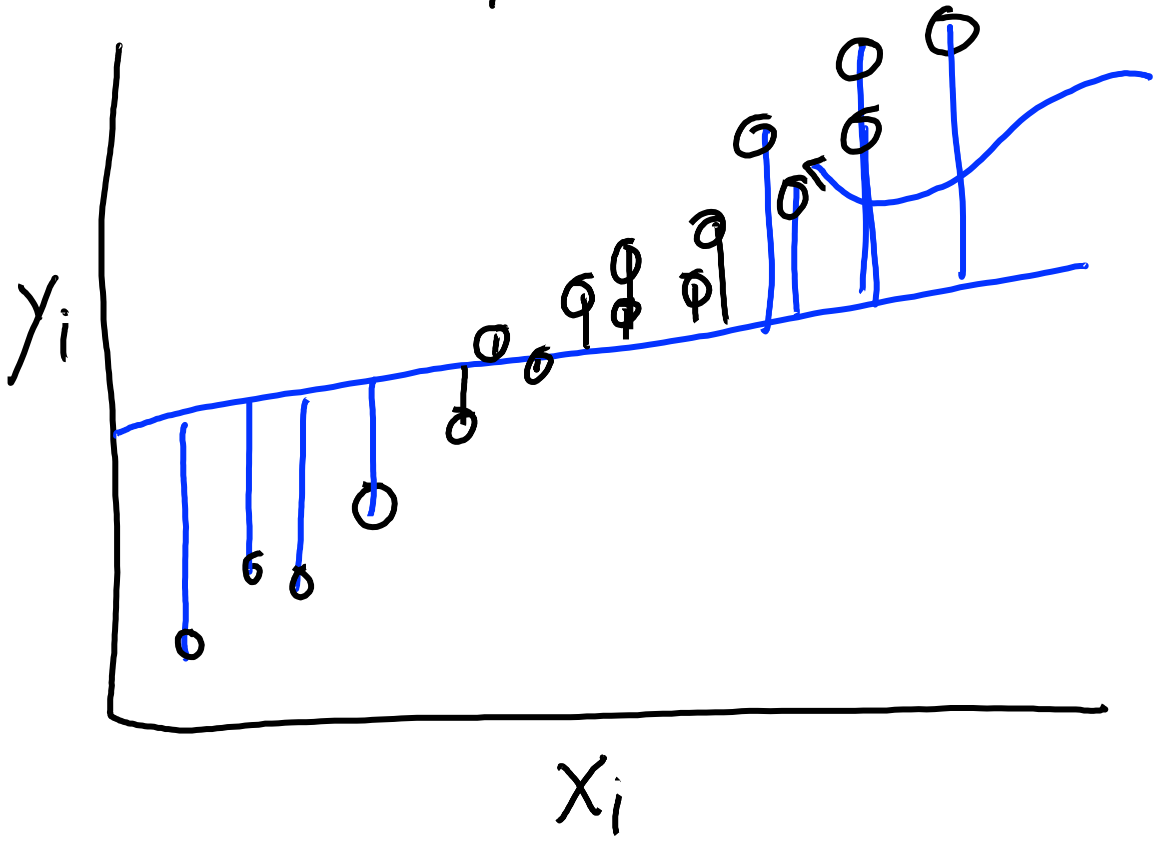
"Error" is the sum of the squared values of these vertical distances between the line (wx_i) and the targets (y_i)

If this error is small, then our predictions are close to the targets.

Least Squares Objective

- Classic way to set slope 'w' is minimizing sum of squared errors:

$$f(w) = \sum_{i=1}^{n} (w x_i - y_i)^2$$

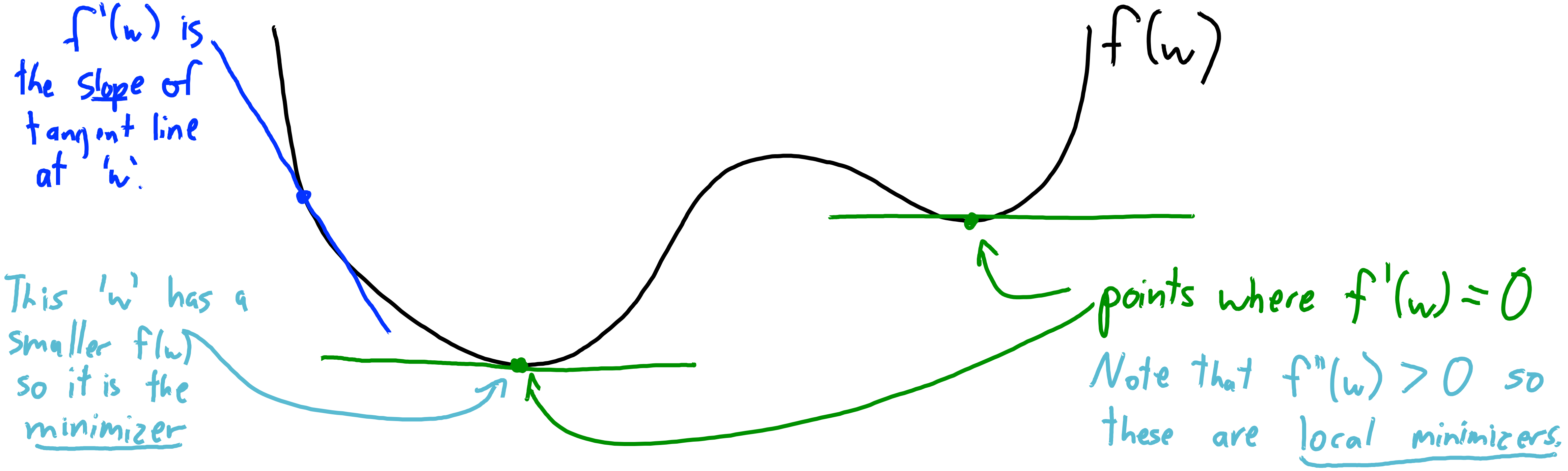


"Error" is the sum of the squared values of these vertical distances between the line (wx_i) and the targets (y_i)

If this error is large, then our predictions are far from the targets.

Minimizing a differentiable function

- Math 101 approach to minimizing a differentiable function ‘f’:
  1. Take the derivative of ‘f’.
  2. Find points ‘w’ where the derivative f’(w) is equal to 0.
  3. Choose the smallest one (but check that f’’(w) is positive).



Terminology (Take 2)

- “Minimum” : the value of f when f(x) is minimized
  - written as min f(x)
- “Minimizer” : the value of x when f(x) is minimized
  - written as arg min f(x)
- “Minima” : plural of minimum
- And vice versa…
  - Maximum, maximizer, maxima

Finding Least Squares Solution

- Finding 'w' that minimizes sum of squared errors:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 = \frac{1}{2}(wx_1 - y_1)^2 + \frac{1}{2}(wx_2 - y_2)^2 + \cdots + \frac{1}{2}(wx_n - y_n)^2$$

$$f'(w) = \sum_{i=1}^{n}(wx_i - y_i)x_i = (wx_1 - y_1)x_1 + (wx_2 - y_2)x_2 + \cdots + (wx_n - y_n)x_n$$

Set $f'(w) = 0$: $\sum_{i=1}^{n}(wx_i - y_i)x_i = 0$ or $\sum_{i=1}^{n}\left[wx_i^2 - y_i x_i\right] = 0$

or $\sum_{i=1}^{n} wx_i^2 = \sum_{i=1}^{n} y_i x_i$

or $w\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i x_i$

so $\boxed{w = \frac{\sum_{i=1}^{n} y_i x_i}{\sum_{i=1}^{n} x_i^2}}$

Is this a minimizer?

$$f''(w) = \sum_{i=1}^{n} x_i^2$$

Since $(\text{anything})^2$ is non-negative, $f''(w) \geq 0$.

If at least one $x_i \neq 0$ then $f''(w) > 0$ and this is a minimizer.

Multiple Explanatory Variables

- Smoking is not the only contributor to lung cancer.
  - For example, environmental factors like exposure to asbestos.
- How can we model the combined effect of smoking and asbestos?
- A simple way is with a 2-dimensional linear function:

$$y_i = w_1 x_{i1} + w_2 x_{i2}$$

Value of feature 2 in example 'i'

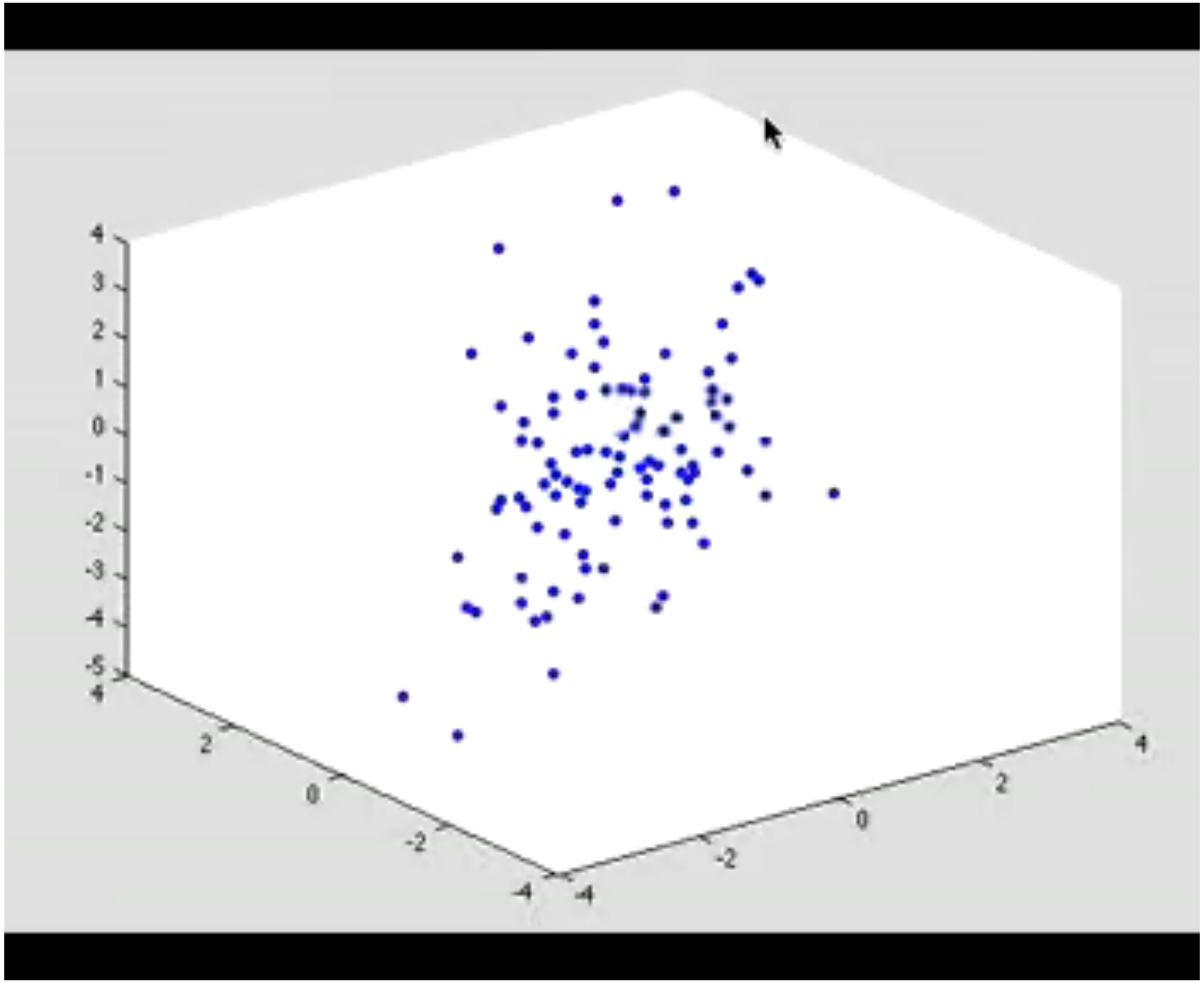
"weight" on feature 2.

"weight" of feature 1

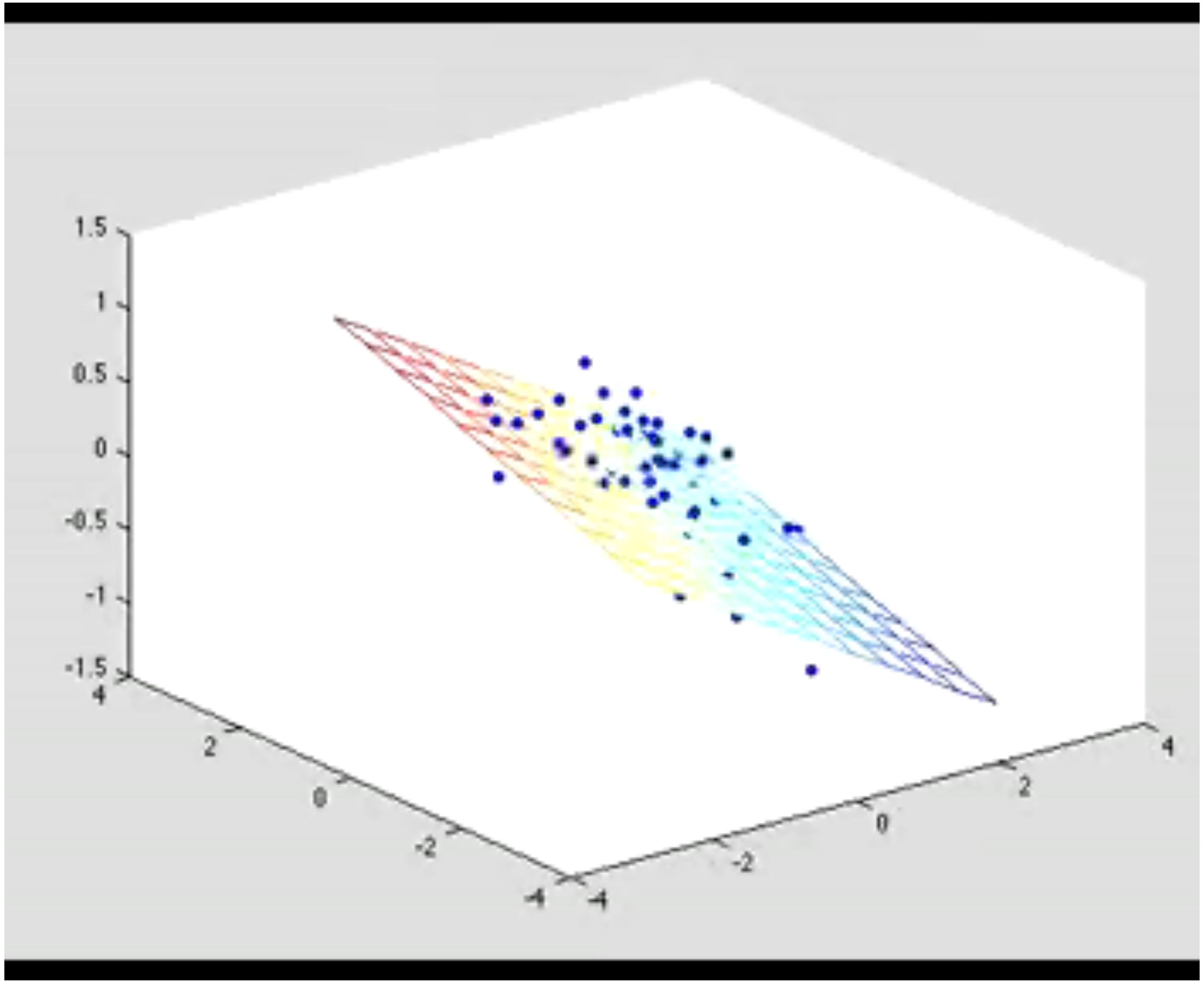
Value of feature 1 in example 'i'

- We have a weight w_1 for feature '1' and w_2 for feature '2'.

Least Squares in 2-Dimensions



Least Squares in 2-Dimensions



Least Squares in d-Dimensions

- If we have 'd' features, the d-dimensional linear model is:

$$y_i = w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3} + \cdots + w_d x_{id}$$

- We can re-write this in summation notation:

$$y_i = \sum_{j=1}^{d} w_j x_{ij}$$

- We can also re-write this in vector notation:

$$y_i = \underbrace{w^T x_i}$$

"inner product" between vectors

$$w^T x = \overbrace{\begin{bmatrix} w_1 & w_2 & \cdots & w_d \end{bmatrix}}^{w^T} \overbrace{\begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix}}^{x_i} = \sum_{j=1}^{d} w_j x_{ij}$$

Notation Alert (again)

- In this course, all vectors are assumed to be column-vectors:

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \qquad x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix}$$

- So $w^T x_i$ is a scalar:

$$w^T x_i = \begin{bmatrix} w_1 & w_2 & \cdots & w_d \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix} = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id}$$

$$= \sum_{j=1}^{d} w_j x_{id}$$

- So rows of 'X' are actually transpose of column-vector x_i:

$$X = \begin{bmatrix} \text{---} & x_1^T & \text{---} \\ \text{---} & x_2^T & \text{---} \\ & \vdots & \\ \text{---} & x_n^T & \text{---} \end{bmatrix}$$

Least Squares in d-Dimensions

- The linear least squares model in d-dimensions minimizes:

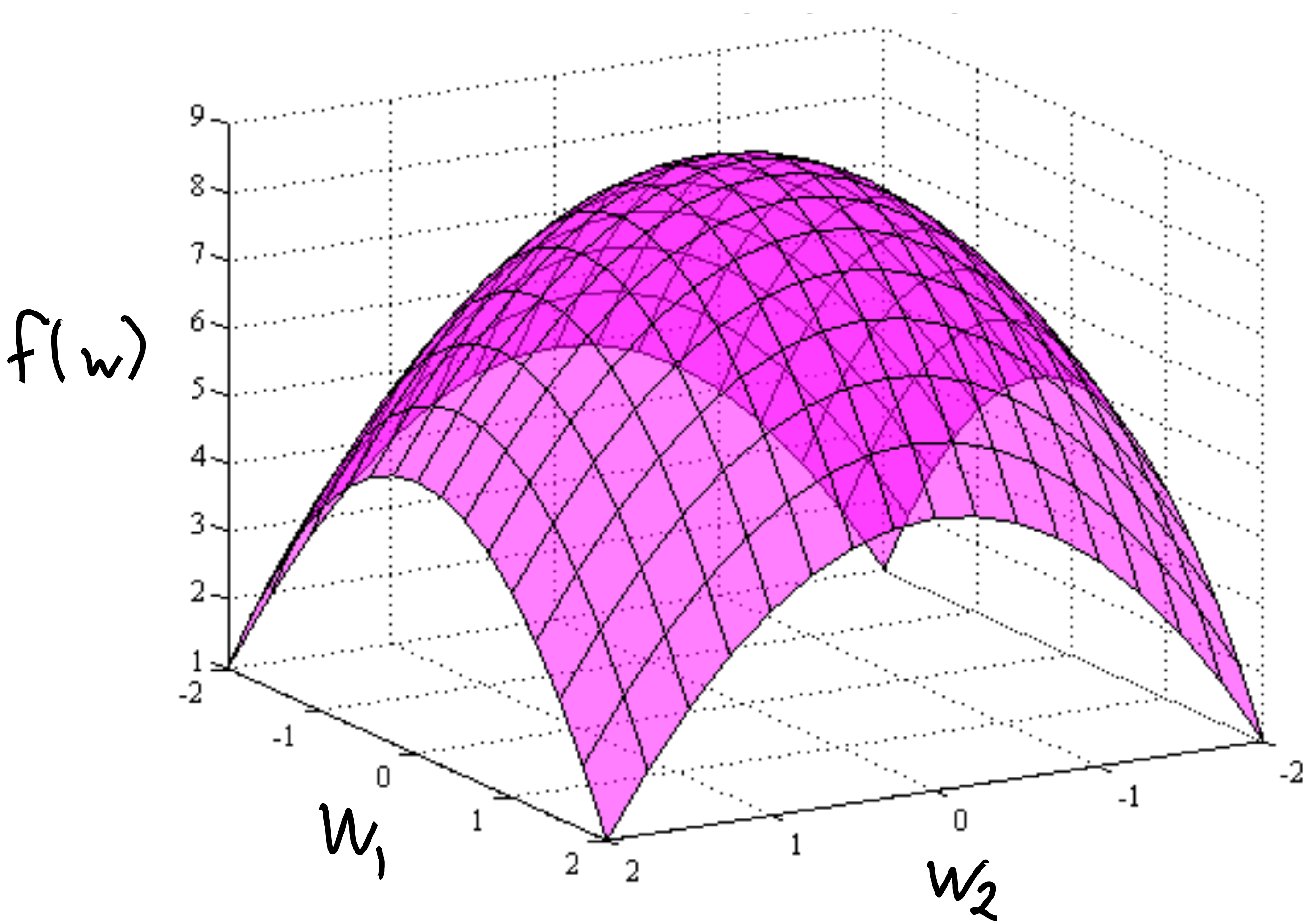
$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(\underbrace{w^Tx_i}_{\text{prediction is inner product of 'w' and 'x}_i\text{' (linear combination of features)}} - y_i)^2$$

'w' is now a vector

"Error" is still the sum of squared differences between "true" y_i and our "prediction" w^Tx_i

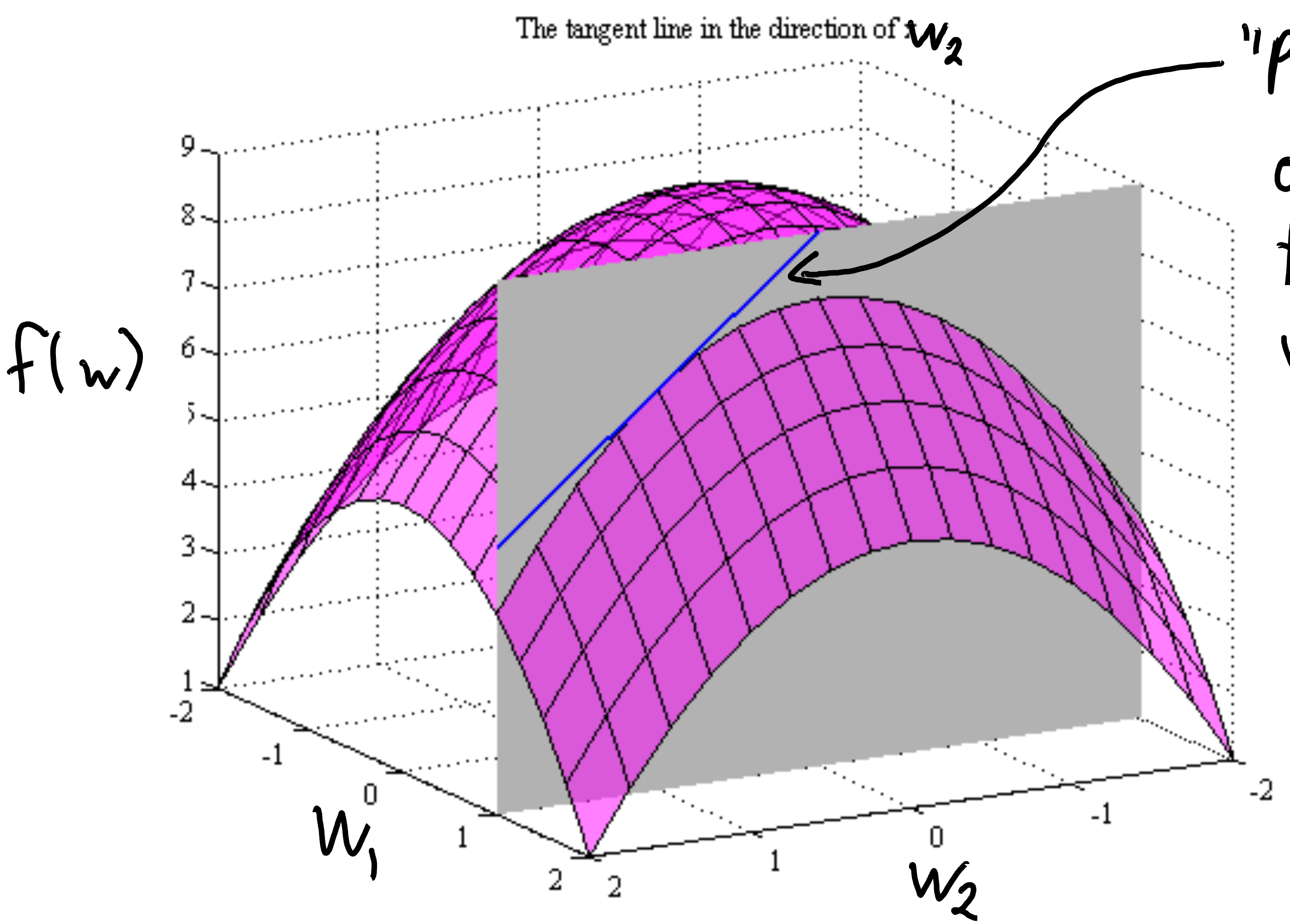
- How do we find the best vector 'w'?
  - Set the derivative of each variable ("partial derivative") to 0?

Partial Derivatives



http://msemac.redwoods.edu/~darnold/math50c/matlab/pderiv/index.xhtml

Partial Derivatives



"Partial" derivative of 'f' with respect to w_2 is the derivative with respect to w when all other variables are held fixed.

Denoted by $\frac{\partial}{\partial w_2}$ for variable w_2

http://msemac.redwoods.edu/~darnold/math50c/matlab/pderiv/index.xhtml

Least Squares in d-Dimensions

- The linear least squares model in d-dimensions minimizes:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^Tx_i - y_i)^2$$

$$w^Tx_i = w_1x_{i1} + w_2x_{i2} + \cdots + w_dx_{id}$$

$$\frac{d}{dw_1}[w^Tx_i] = x_{i1} + 0 + \cdots + 0 = x_{i1}$$

- Computing the partial derivative:

$$\frac{\partial}{\partial w_1}\left[\frac{1}{2}\sum_{i=1}^{n}(w^Tx_i - y_i)^2\right] = \frac{1}{2}\sum_{i=1}^{n}\frac{\partial}{\partial w_1}\left[(w^Tx_i - y_i)^2\right]$$

$$= \frac{1}{2}\sum_{i=1}^{n} 2(w^Tx_i - y_i)\underbrace{\frac{\partial}{\partial w_1}[w^Tx_i]}_{\text{What is the derivative of } w^Tx_i \text{ with respect to } w_1?}$$

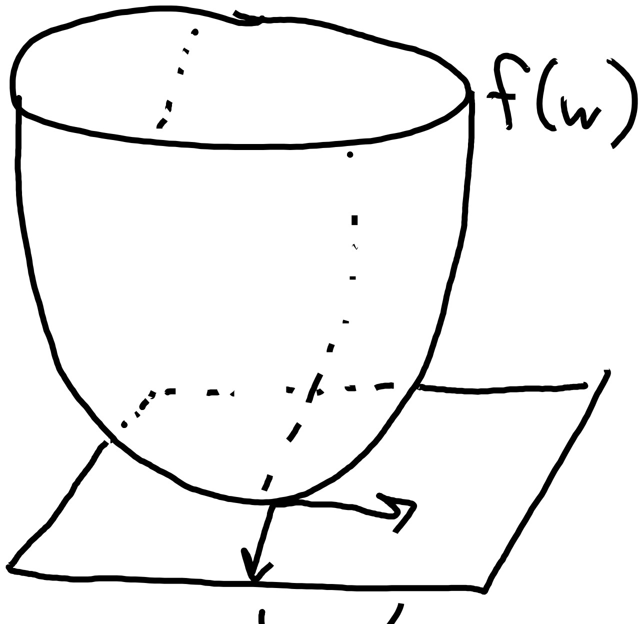
$$= \sum_{i=1}^{n}(w^Tx_i - y_i)x_{i1}$$

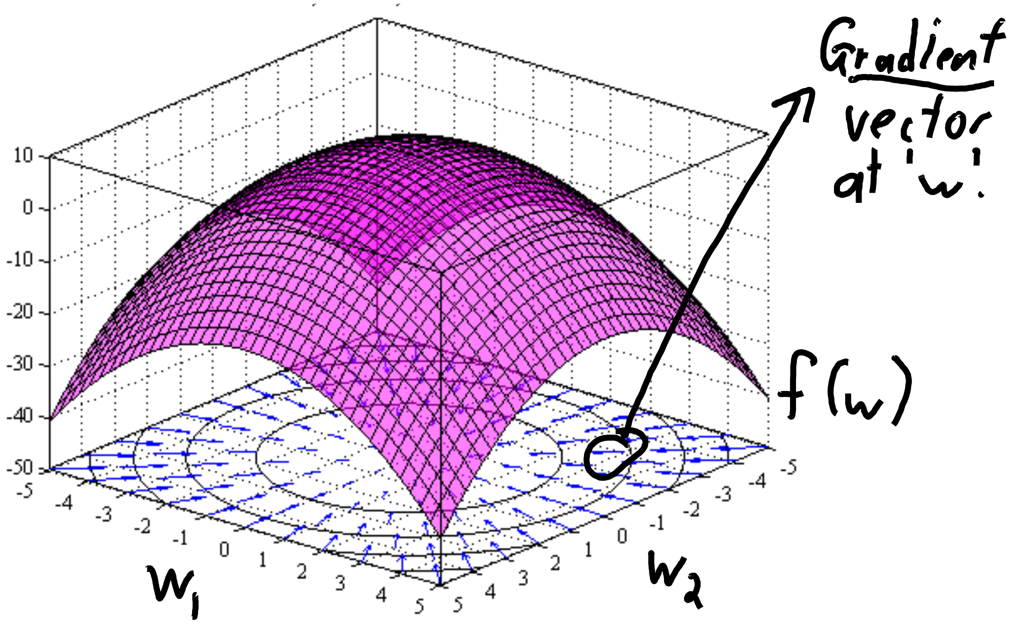
Problem: I can't just set to 0 and solve because it depends on $w_2, w_3, \ldots, w_d$

Gradient and Critical Points in d-Dimensions

- Generalizing “set the derivative to 0 and solve” in d-dimensions:
  - Find ‘w’ where the gradient vector equals the zero vector.
- Gradient is vector with partial derivative ‘j’ in position ‘j’:

$$\nabla f(w) = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \vdots \\ \frac{\partial f}{\partial w_d} \end{bmatrix}$$





Tangent slope is 0 in every direction at minimizer.

ac.redwoods.edu/~darnold/math50c/matlab/pderiv/index.xhtml

Gradient and Critical Points in d-Dimensions

- Generalizing "set the derivative to 0 and solve" in d-dimensions:
  - Find 'w' where the gradient vector equals the zero vector.
- Gradient is vector with partial derivative 'j' in position 'j':

$$\nabla f(w) = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \vdots \\ \frac{\partial f}{\partial w_d} \end{bmatrix}$$

For linear least squares:

$$\nabla f(w) = \begin{bmatrix} \sum_{i=1}^{n} (w^T x_i - y_i) x_{i1} \\ \sum_{i=1}^{n} (w^T x_i - y_i) x_{i2} \\ \vdots \\ \sum_{i=1}^{n} (w^T x_i - y_i) x_{id} \end{bmatrix}$$

Claims for linear least squares:

1. Finding a 'w' where $\nabla f(w) = 0$ can be done by solving a <u>system of linear equations.</u>
2. <u>All</u> 'w' where $\nabla f(w) = 0$ are <u>minimizers.</u>

There is a lot more to linear regression

- You can take an entire statistics course in linear regression
- Additional topics include
  - "interaction terms"
  - Feature selection
  - Model diagnostics (training/test error?)
  - Robust regression
  - Missing data
  - Multicollinearity
  - Computational issues
  - Connection to classification
- We will cover some of the above topics later in the course.

Summary

- Regression considers the case of a numerical y_i.
- Least squares is a classic method for fitting linear models.
  - With 1 feature, it has a simple closed-form solution.
- Gradient is vector containing partial derivatives of all variables.
- Linear system of equations gives least squares with 'd' features.

- Next time: *non-linear* regression.