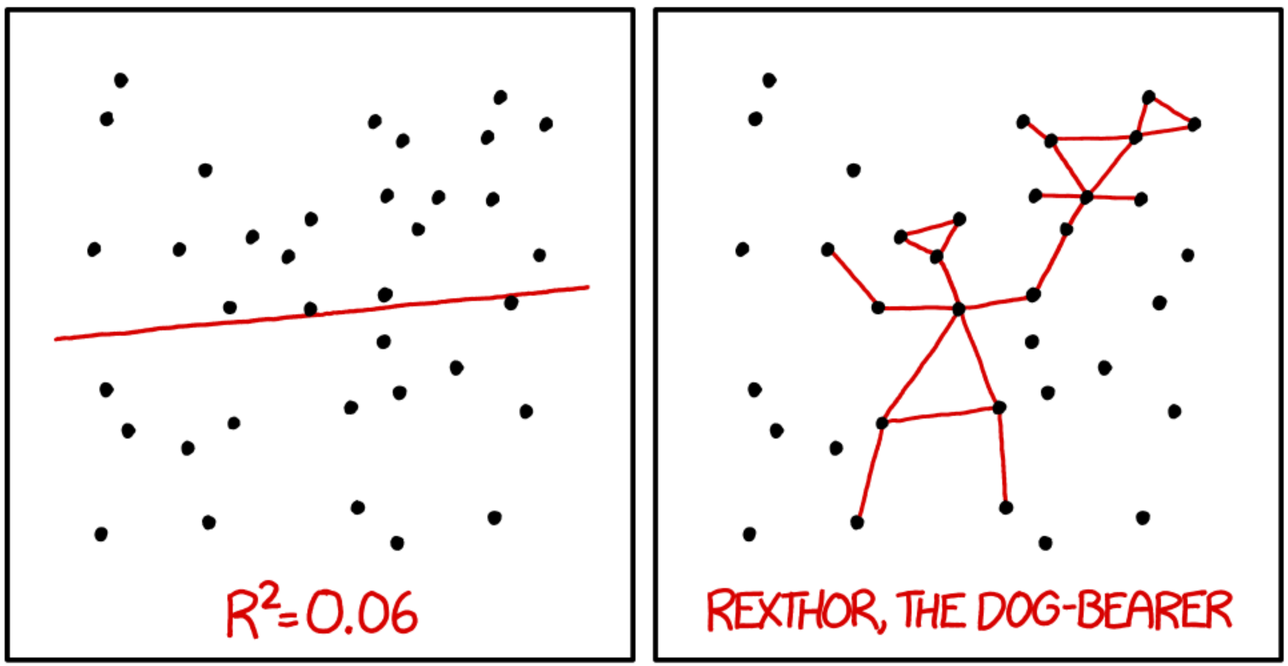
Linear models

Announcements

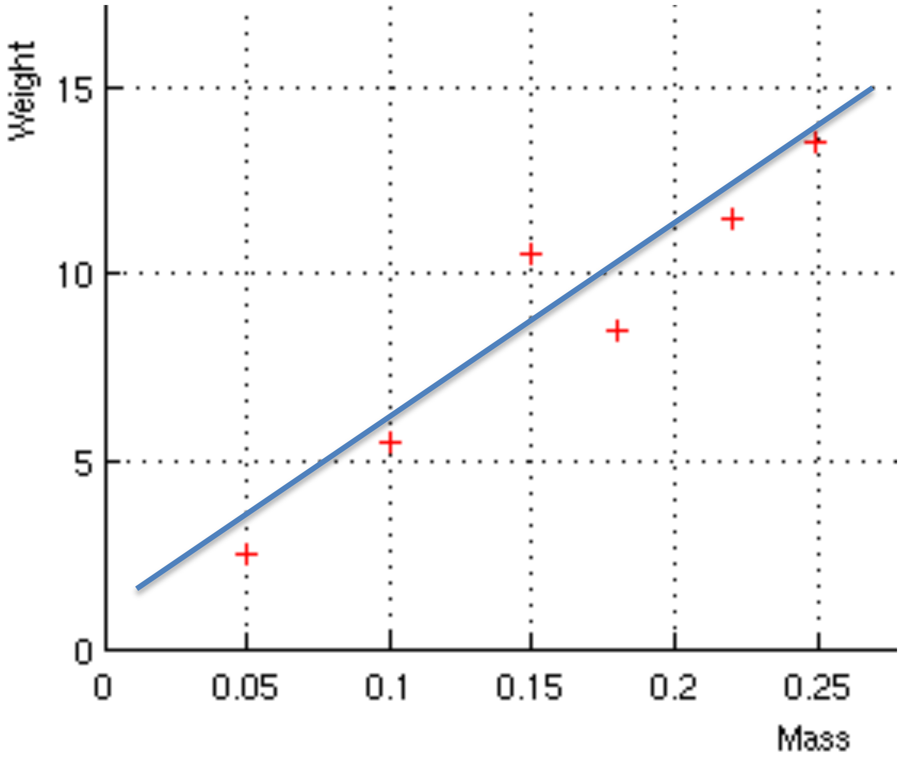
- A2 due date extended to November 11 (you can test your agents by playing against others – see Q&A post)
- A3 and optional A4 to follow (A4 will have a deadline during the last week of classes)
- Grades for A1 will be posted later tonight/tomorrow.
- Survey to follow



I DON'T TRUST LINEAR REGRESSIONS WHEN IT'S HARDER TO GUESS THE DIRECTION OF THE CORRELATION FROM THE SCATTER PLOT THAN TO FIND NEW CONSTELLATIONS ON IT.

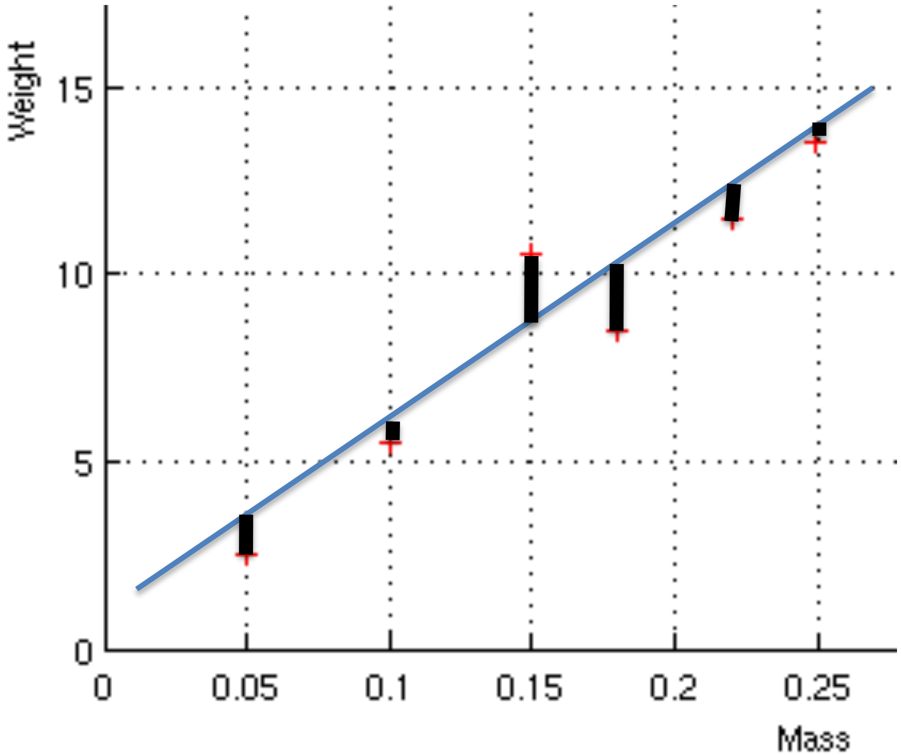
A step back: Line fitting

- Suppose you want to fit a line to some data points
  - I.e. find a line that minimizes the sum squared distances between each data point to the line



A step back: Line fitting

- Suppose you want to fit a line to some data points
  - I.e. find a line that minimizes the sum squared distances between each data point to the line



- **Linear regression** gives unique solution in closed form:

$$w_1 = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}$$

$$w_0 = (\sum y_j - w_1(\sum x_j))/N$$

Example

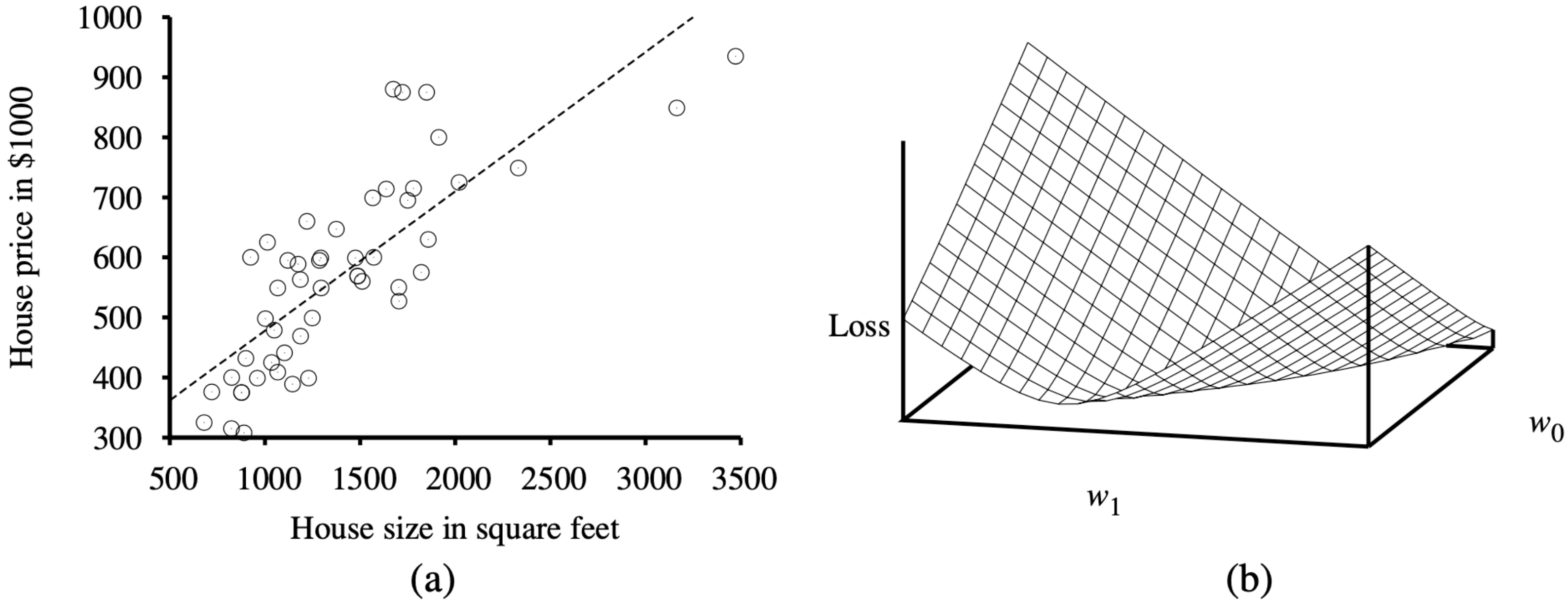
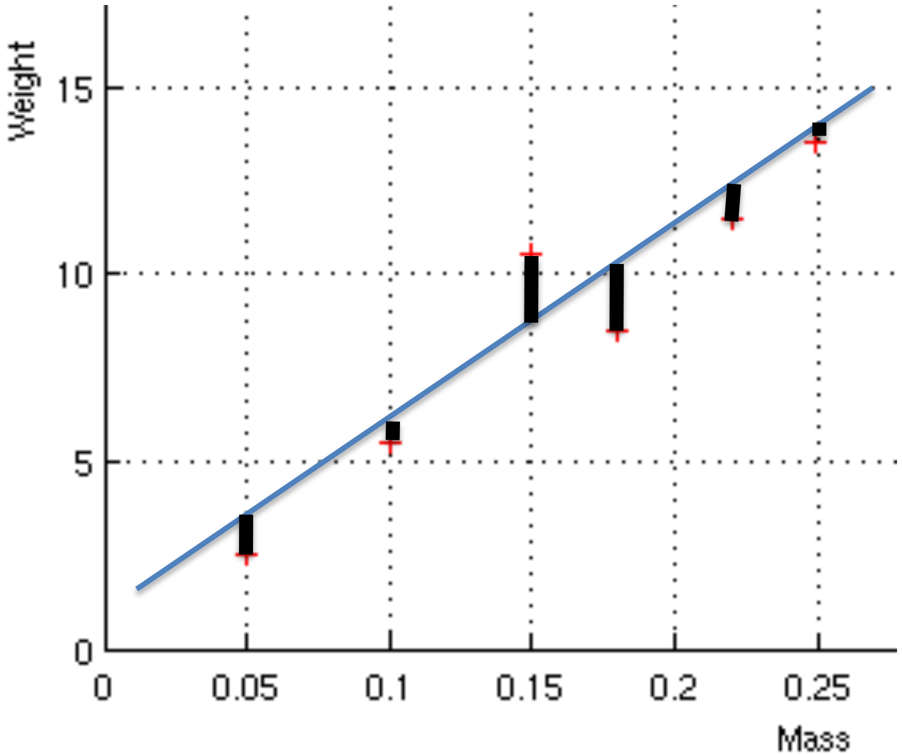


Figure 18.13 (a) Data points of price versus floor space of houses for sale in Berkeley, CA, in July 2009, along with the linear function hypothesis that minimizes squared error loss: $y = 0.232x + 246$. (b) Plot of the loss function $\sum_j (w_1 x_j + w_0 - y_j)^2$ for various values of w_0, w_1. Note that the loss function is convex, with a single global minimum.

Russell & Norvig

A step back: Line fitting

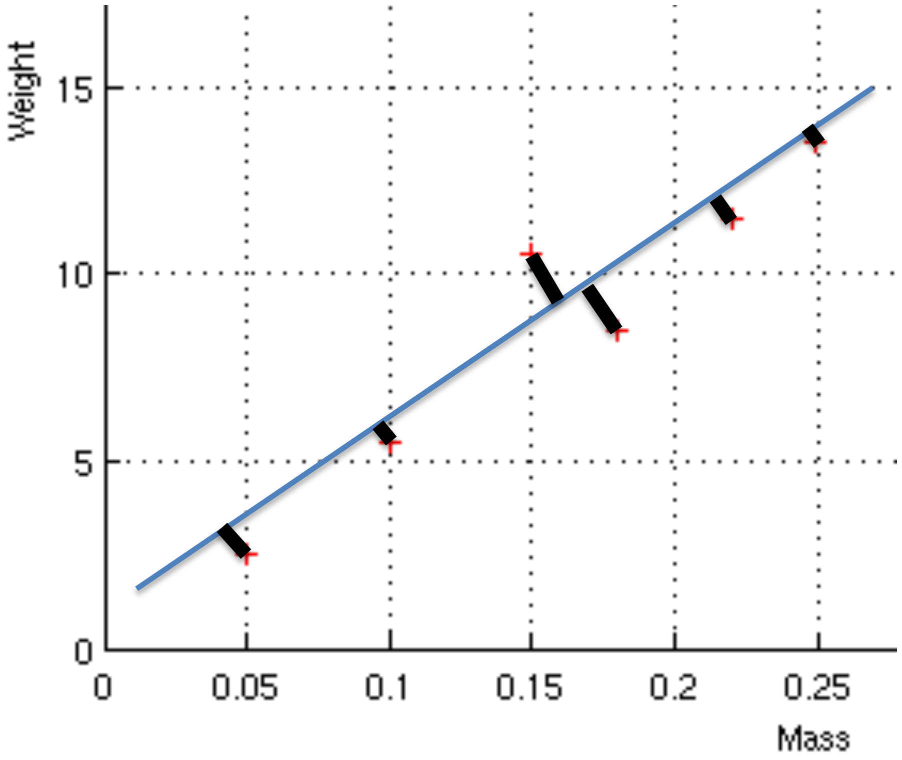
- Suppose you want to fit a line to some data points
  - I.e. find a line that minimizes the sum squared distances between each data point to the line



- Linear regression
  - aka "ordinary least squares"
  - Measures errors along y-axis only
  - Assumes observations on x-axis are error free

A step back: Line fitting

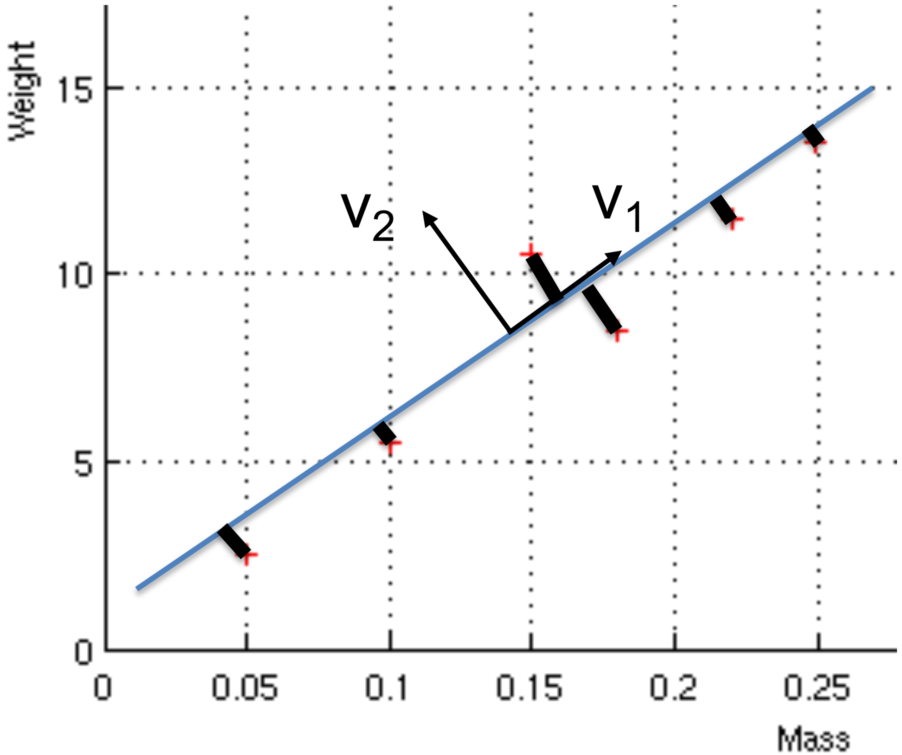
- Suppose you want to fit a line to some data points
  - I.e. find a line that minimizes the sum of distances between each data point to the line



- Total least squares
  - Measures Euclidean distance from point to line

A step back: Line fitting

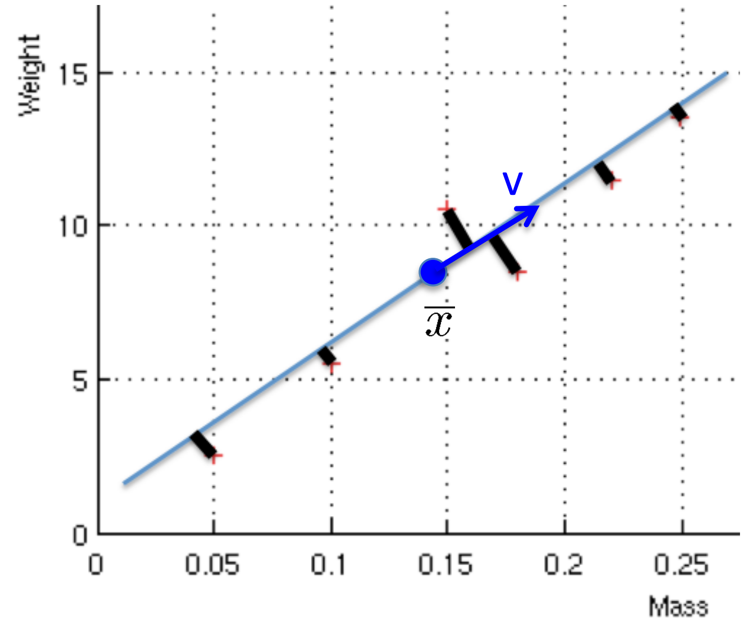
- Suppose you want to fit a line to some data points
  - I.e. find a line that minimizes the sum of distances between each data point to the line



- Total least squares
  - Measures Euclidean distance from point to line

Total least squares

- Can show that the best line passes through the centroid (mean) of the points, $\overline{x}$
- What about the direction?
  - Need to solve for vector *v* by minimizing error,



v that minimizes v^TAv is the largest eigenvector of A

$$
\begin{aligned}
\text{Total error} &= \sum_{x} \left((x-\bar{x})^T \cdot v\right)^2 \\
&= \sum_{\mathbf{x}} \mathbf{v}^{\mathbf{T}}(\mathbf{x}-\overline{\mathbf{x}})(\mathbf{x}-\overline{\mathbf{x}})^{\mathbf{T}}\mathbf{v} \\
&= \mathbf{v}^{\mathbf{T}}\left[\sum_{\mathbf{x}}(\mathbf{x}-\overline{\mathbf{x}})(\mathbf{x}-\overline{\mathbf{x}})^{\mathbf{T}}\right]\mathbf{v} \\
&= \mathbf{v}^{\mathbf{T}}\mathbf{A}\mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}}(\mathbf{x}-\overline{\mathbf{x}})(\mathbf{x}-\overline{\mathbf{x}})^{\mathbf{T}}
\end{aligned}
$$

Works in multivariate case too!

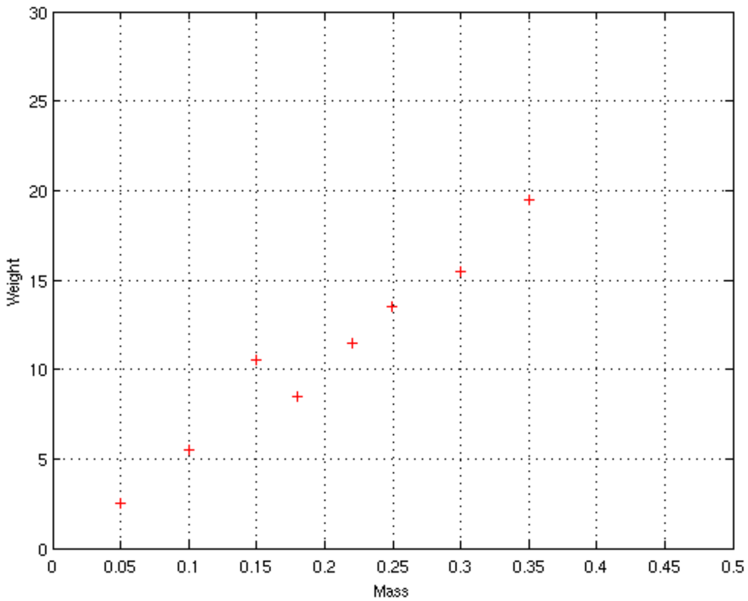
Total least squares

- A line minimizing the total least squares can be found using eigendecomposition
  1. Put the data in a matrix, X, where each row is a vector representing a data point
  2. Compute the mean of the columns of X, giving a vector $\mathbf{x}$
  3. Compute the covariance matrix, $A = (X-\mathbf{x})^T(X-\mathbf{x})$
  4. Find the eigenvectors and eigenvalues of A

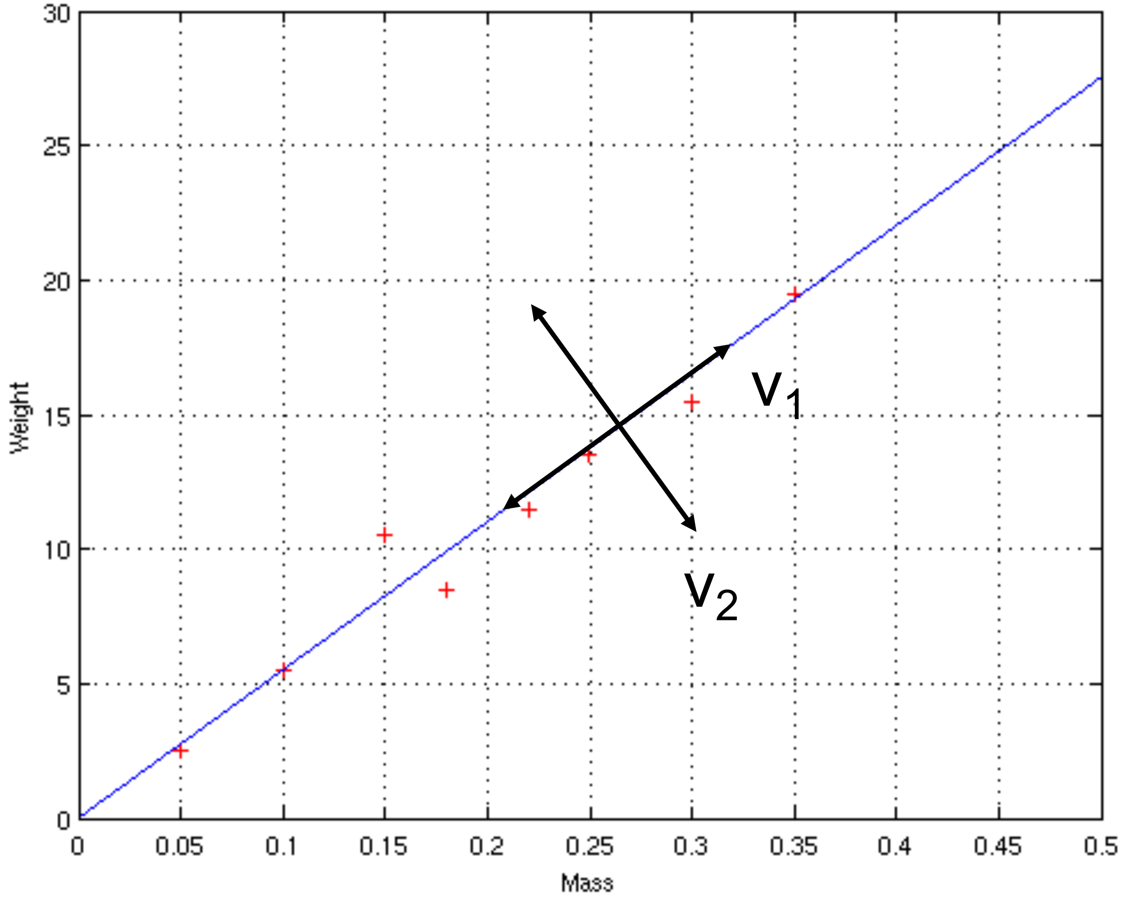
Example

$$X = \begin{bmatrix} 0.15 & 10.5 \\ 0.05 & 2.5 \\ 0.18 & 8.5 \\ 0.10 & 5.5 \\ 0.25 & 13.5 \\ 0.35 & 19.5 \\ 0.30 & 15.5 \\ 0.22 & 11.5 \end{bmatrix} \qquad Y = \begin{bmatrix} -0.05 & -0.375 \\ -0.15 & -8.375 \\ -0.02 & -2.375 \\ -0.10 & -5.375 \\ 0.05 & 2.625 \\ 0.15 & 8.625 \\ 0.10 & 4.625 \\ 0.02 & 0.625 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.0708 & 3.7600 \\ 3.7600 & 207.8750 \end{bmatrix}$$



- Eigenvectors and eigenvalues of A:
  - v1 = (0.0181, 0.999), λ1 = 208
  - v2 = (-0.999, 0.0181), λ2 = 0.0181

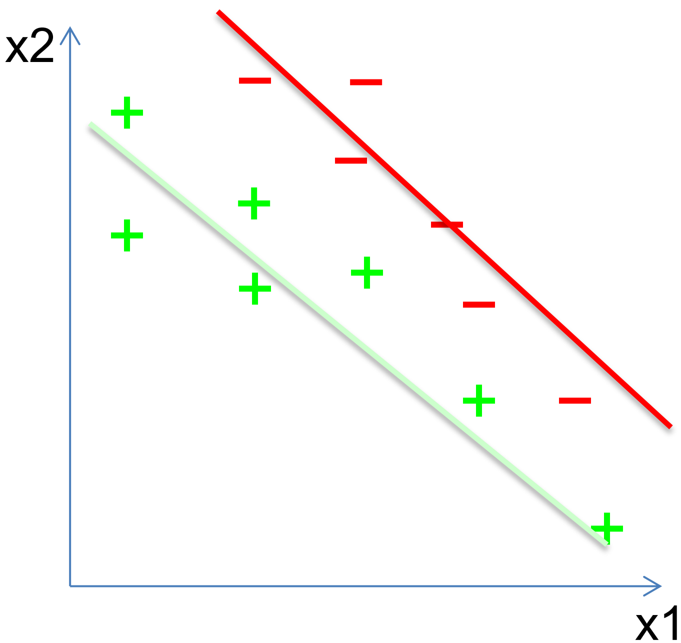


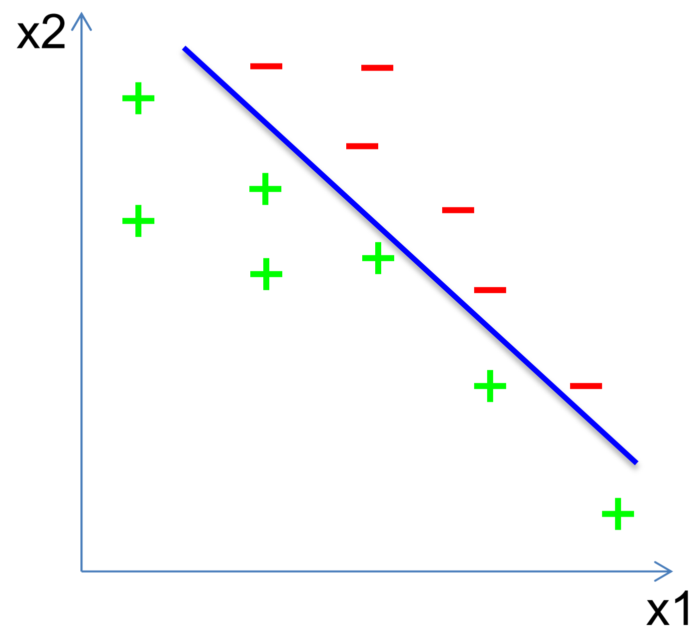
- Eigenvectors and eigenvalues of A:
  - $v_1 = (0.0181, 0.999)$, $\lambda_1 = 208$
  - $v_2 = (-0.999, 0.0181)$, $\lambda_2 = 0.0181$

Classification

- We can model the class itself, but often want to model the *difference* between two classes

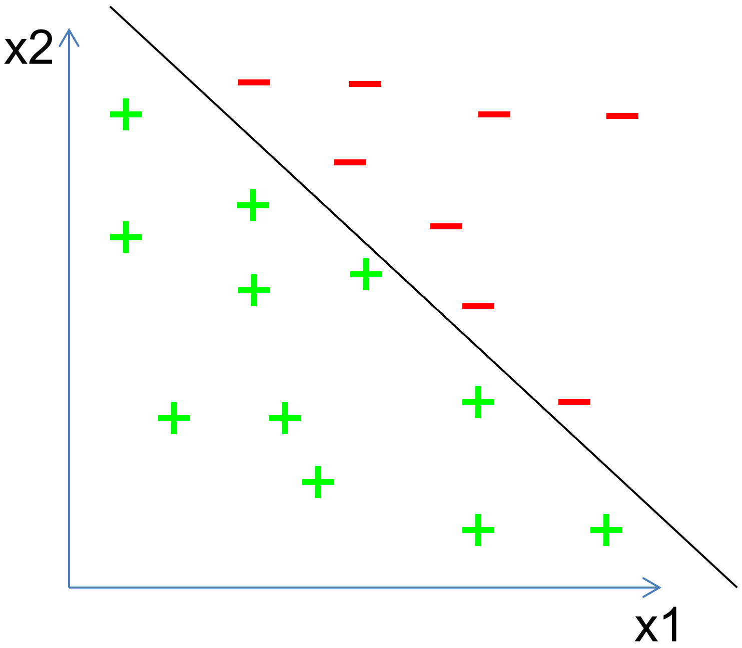
Generative vs **Discriminative**





Linear classifiers : Motivation

- Decision tree produces axis-aligned decision boundaries and can’t accurately classify data like this:

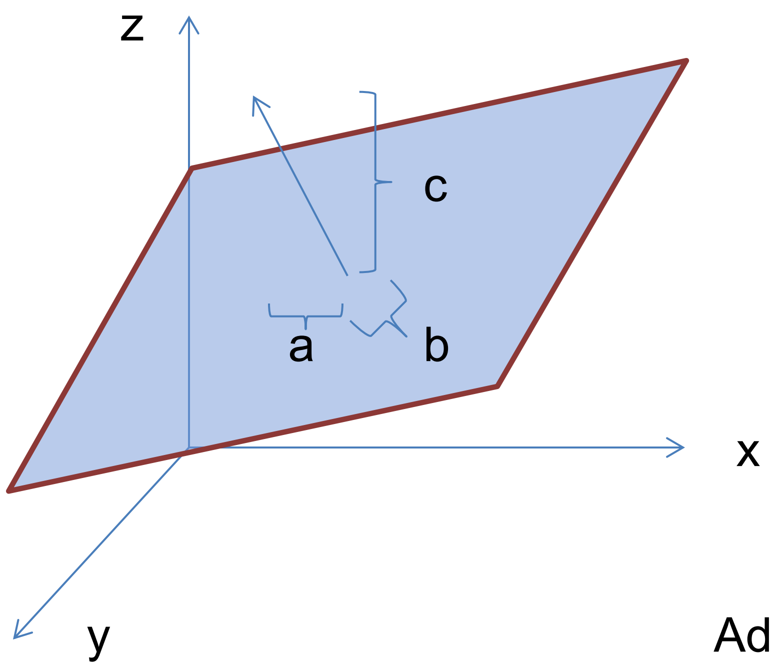


- Linear classifiers model this boundary directly

Adapted from K. Hauser’s slide

Plane Geometry

- In 3D, a plane can be expressed as the set of solutions (x,y,z) to the equation ax+by+cz+d=0
  - ax+by+cz+d > 0 is one side of the plane
  - ax+by+cz+d < 0 is the other side
  - ax+by+cz+d = 0 is the plane itself



Adapted from K. Hauser’s slide

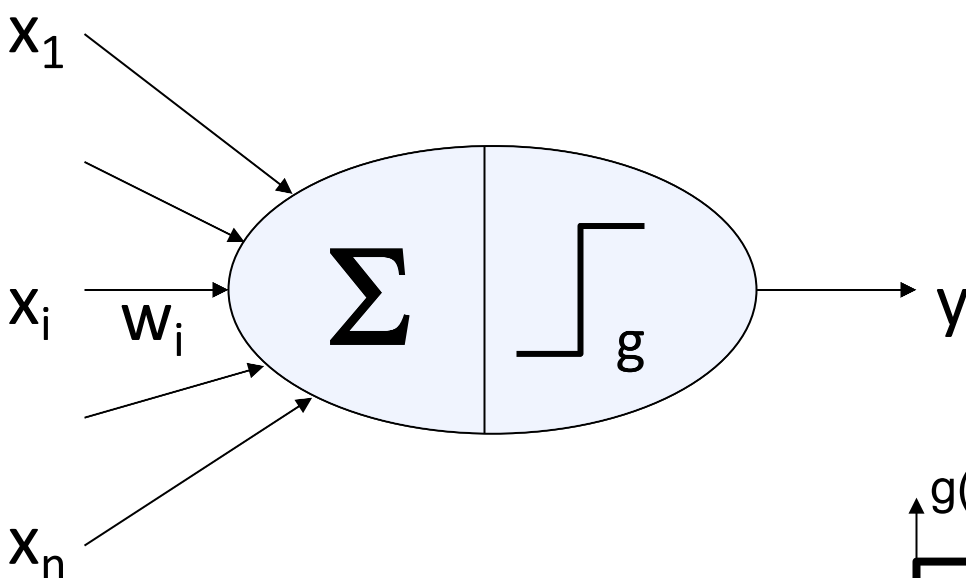
Linear Classifier

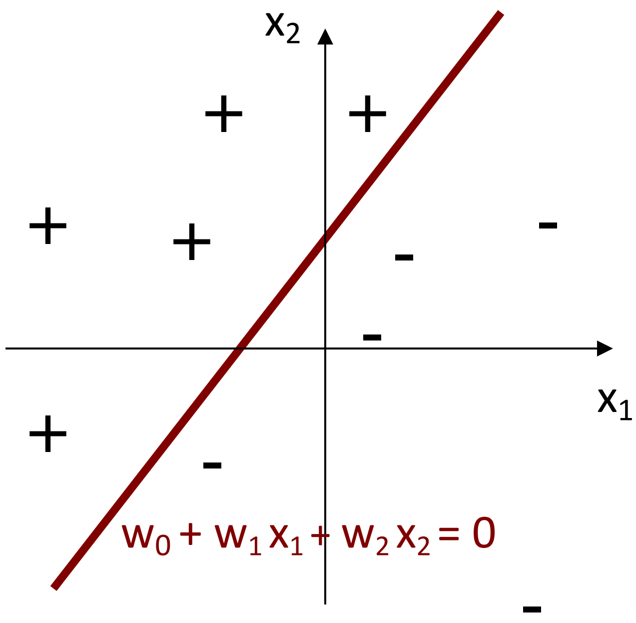
- In d dimensions,
  - $w_0+w_1*x_1+\ldots+w_d*x_d=0$

    is a hyperplane.
- Idea:
  - Use $w_0+w_1*x_1+\ldots+w_d*x_d \geq 0$ to denote positive classifications
  - Use $w_0+w_1*x_1+\ldots+w_d*x_d < 0$ to denote negative classifications

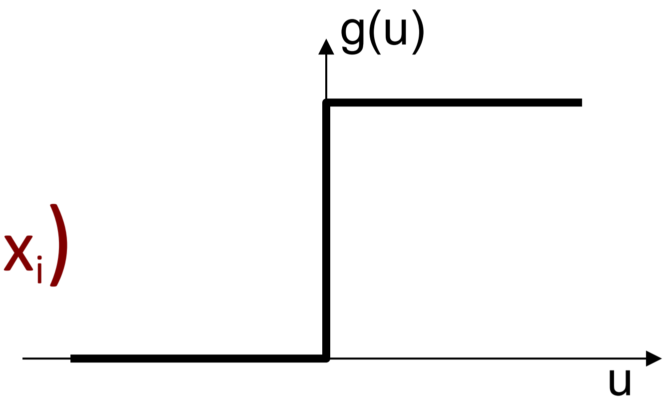
Adapted from K. Hauser's slide

Perceptron



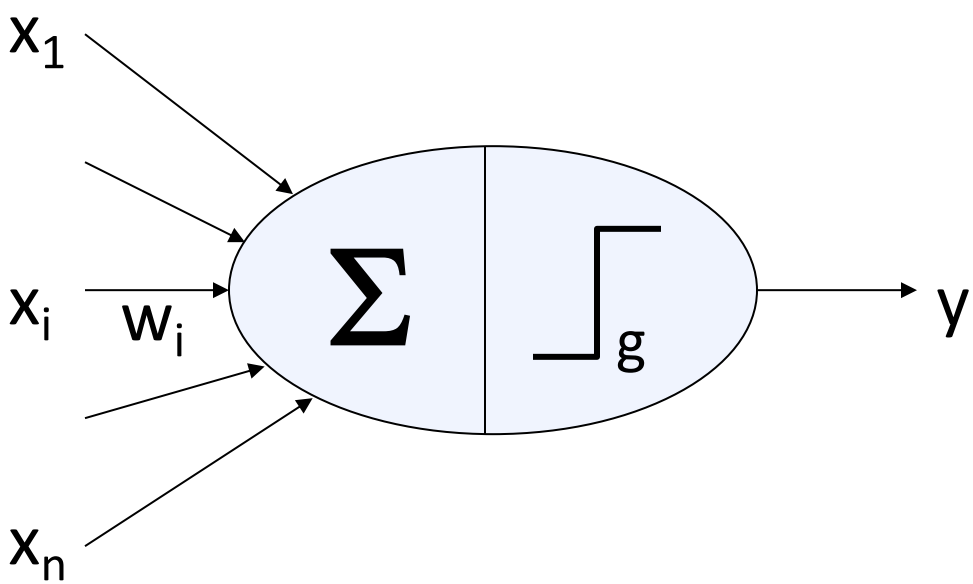


$$y = f(\mathbf{x},\mathbf{w}) = g\left(\Sigma_{i=1,\ldots,n}\, w_i\, x_i\right)$$
$$= g(w \bullet x)$$



Adapted from K. Hauser's slide

Perceptrons can model different functions



The function $x_1 \wedge x_2 \wedge \neg x_3$?

Majority function (if most of inputs are 1 return 1, otherwise return 0)?

Adapted from K. Hauser's slide

Learning perceptrons

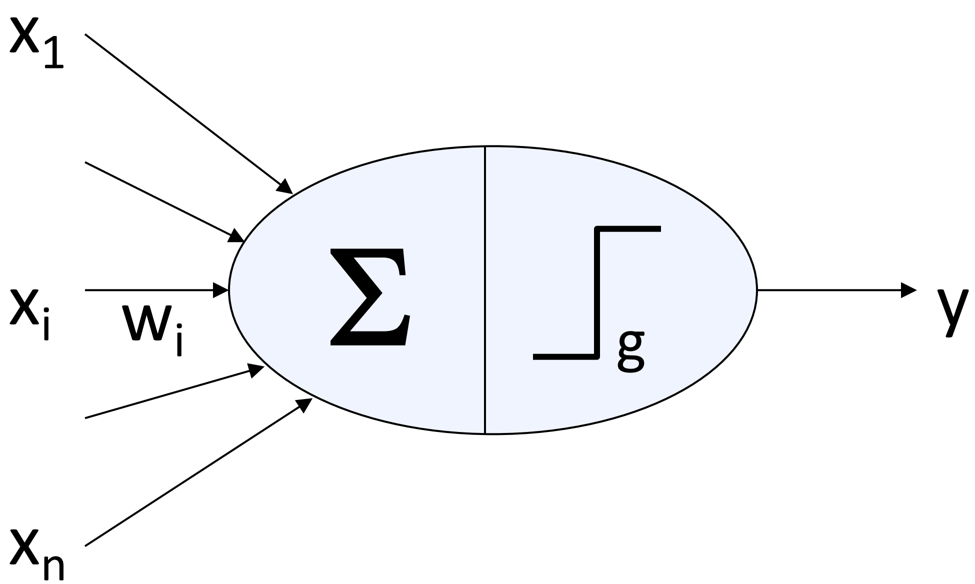
- How do we learn w?
  - Take derivative, set equal to 0, solve for w?
- Simple update rule:
  - Start with initial guess of w.
  - For each exemplar (x,y), and each weight w_i, update:

    $w_i \leftarrow w_i + \alpha\, x_i\, (y - g(w^T x))$

    (where y is either 0 or 1 and g() is either 0 or 1)
- Converges if data is linearly separable, but oscillates otherwise

Adapted from K. Hauser's slide

Perceptrons can model different functions



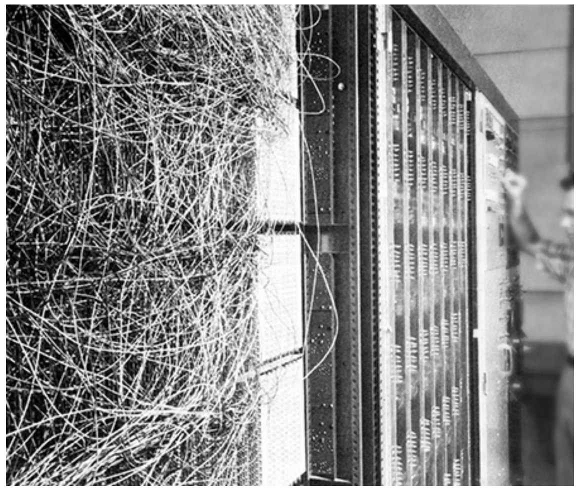
The function $x_1 \wedge x_2 \wedge \neg x_3$?

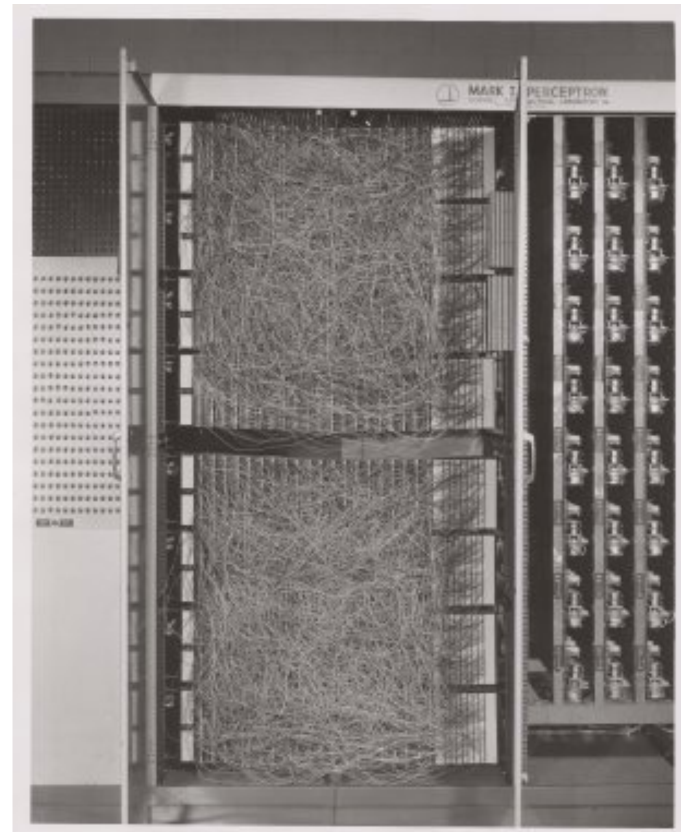
Majority function ?

XOR ?

Adapted from K. Hauser's slide

Perceptrons

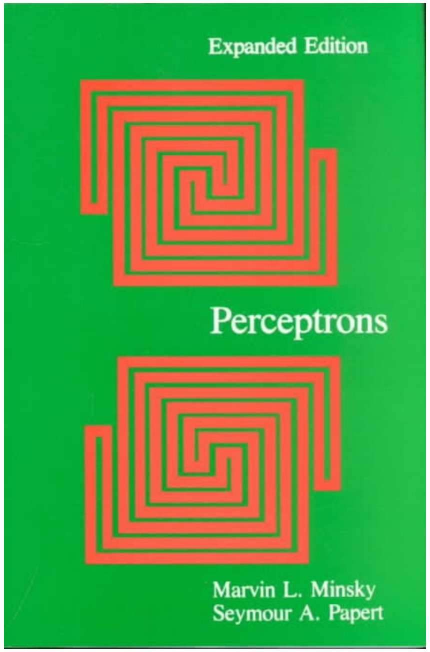




“the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.”

Frank Rosenblatt, 1958

Perceptrons



...until Minky & Papert showed they couldn’t even learn XOR. (1969)

Next class

- Leaning in neural networks