Machine Learning: Introduction to Supervised Learning

Announcements

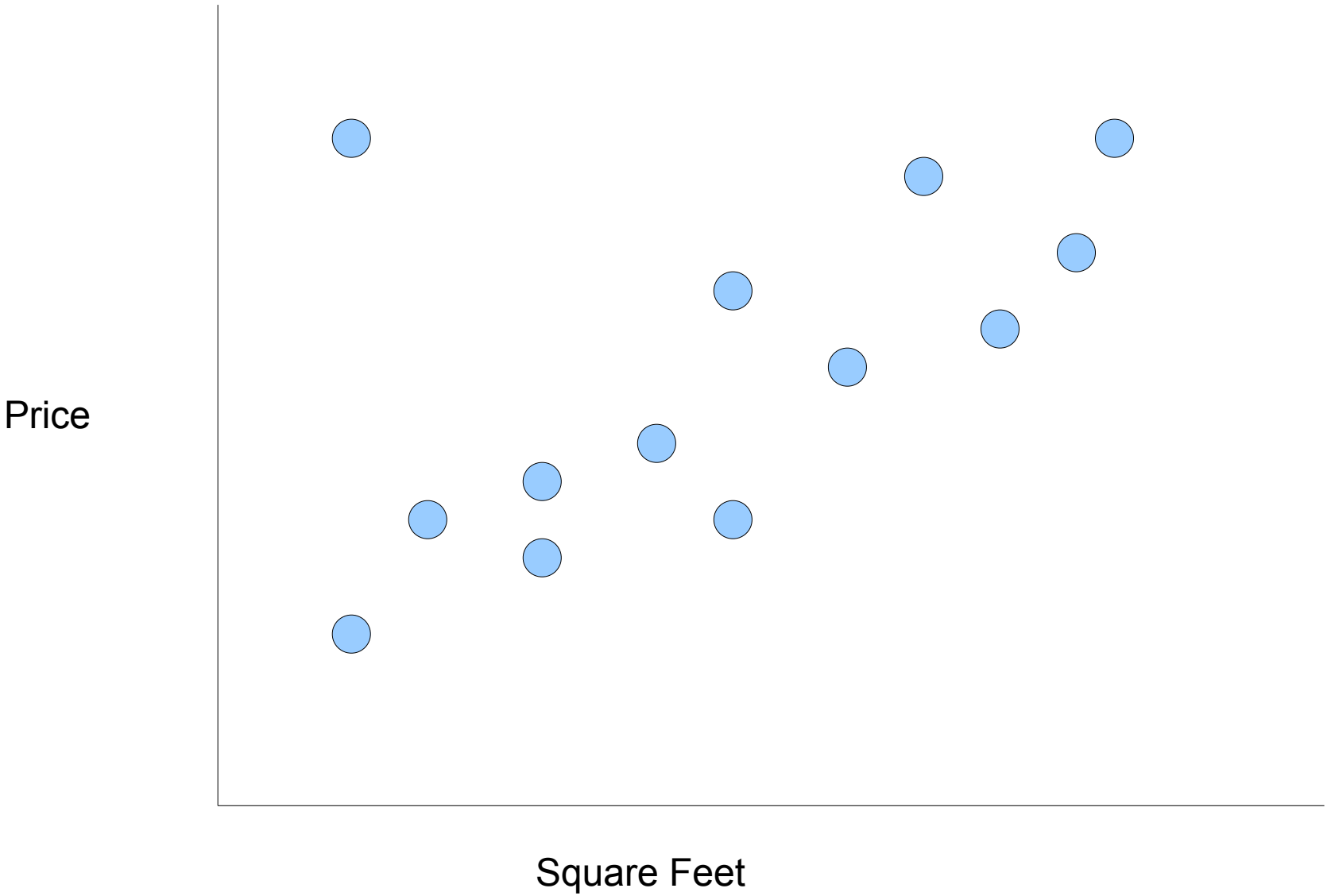
- Please fill out course evaluations!
- Can pick up exams after class
  - Extra exams will be in Gates 160
- Criteria posted online later today
  - All regrade requests **must** be made by this Friday
- Aubrey's Office Hours:
  - Wednesday: 12-5PM
  - Thursday: 12-5PM
  - Friday: 12-5PM (for regrade requests)

Buying Houses

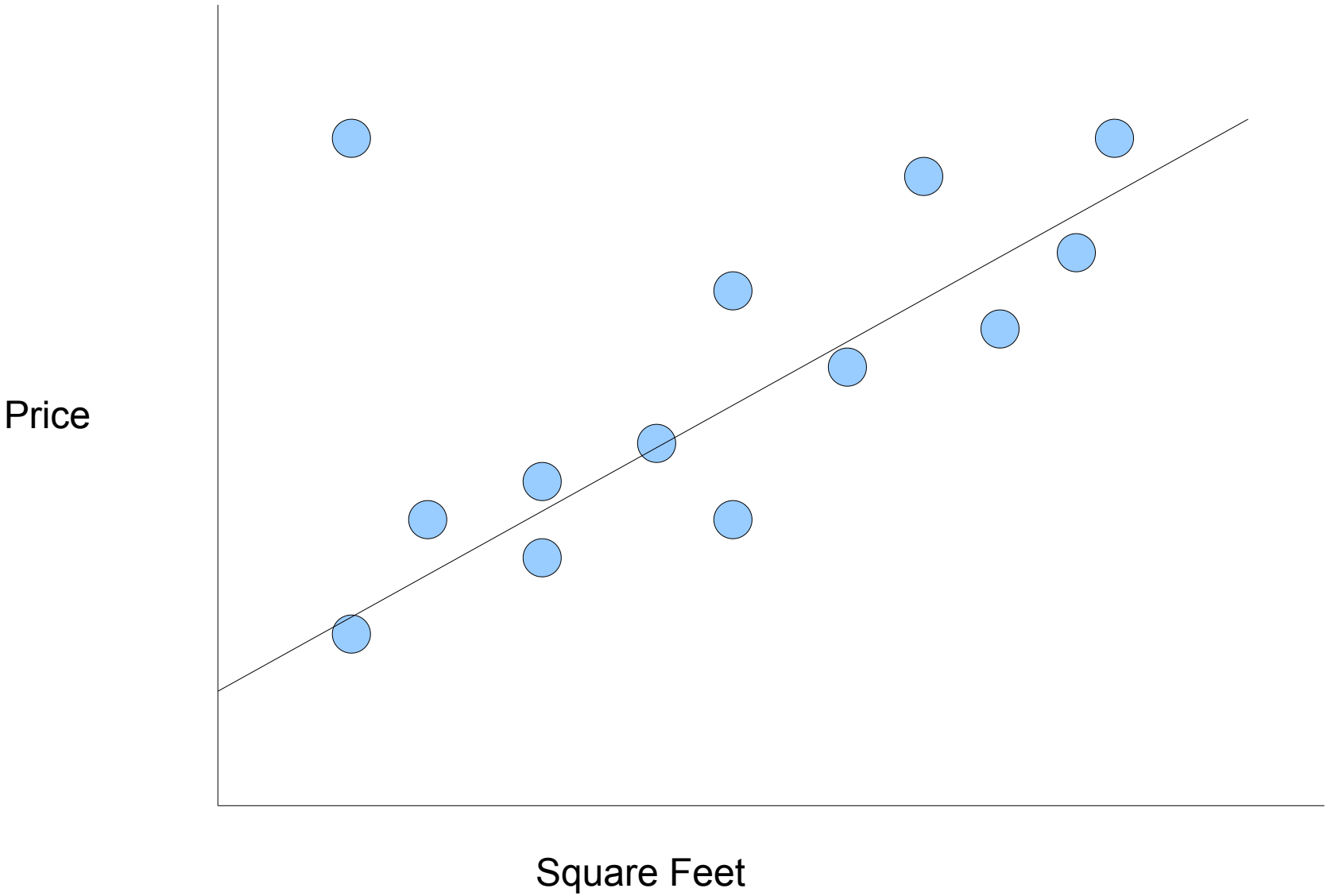
- Given housing price data, how can we make generalizations from that data in order to predict housing prices?

Square Feet	Price
500	$100,000
800	$200,000
400	$120,000
550	$180,000
700	$250,000
900	$300,000
300	$120,000
500	$200,000
475	$120,000

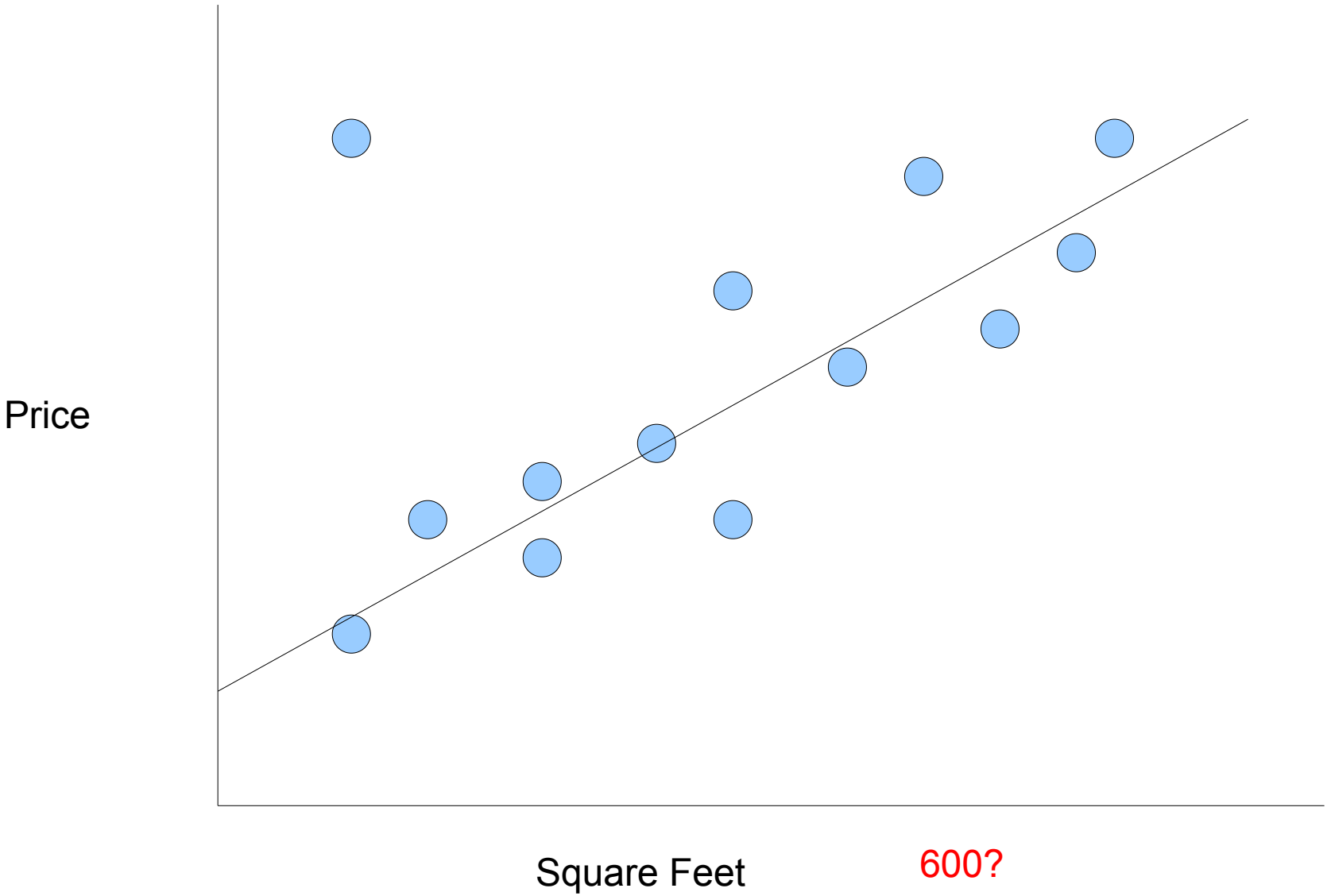
Buying Houses



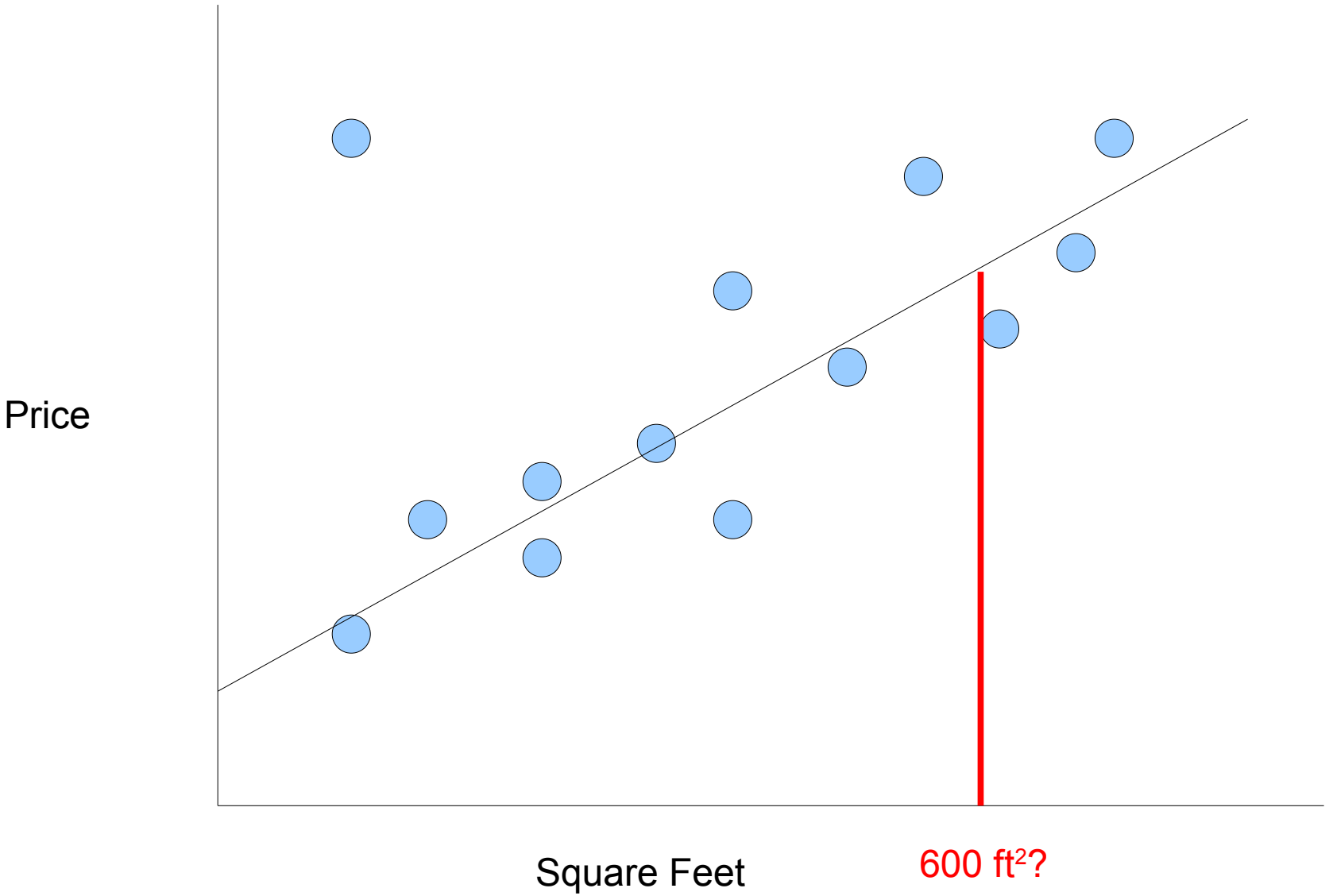
Buying Houses



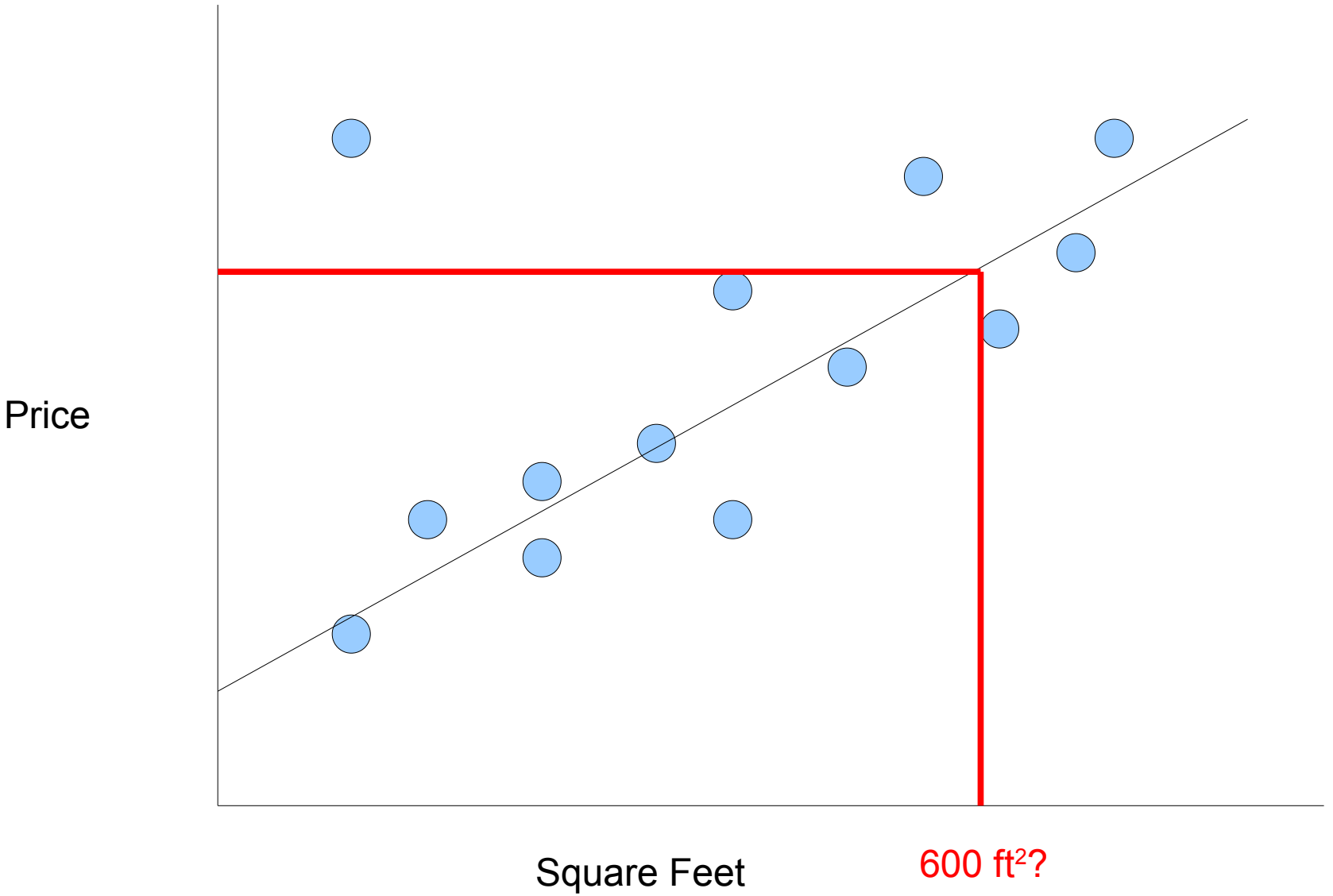
Buying Houses



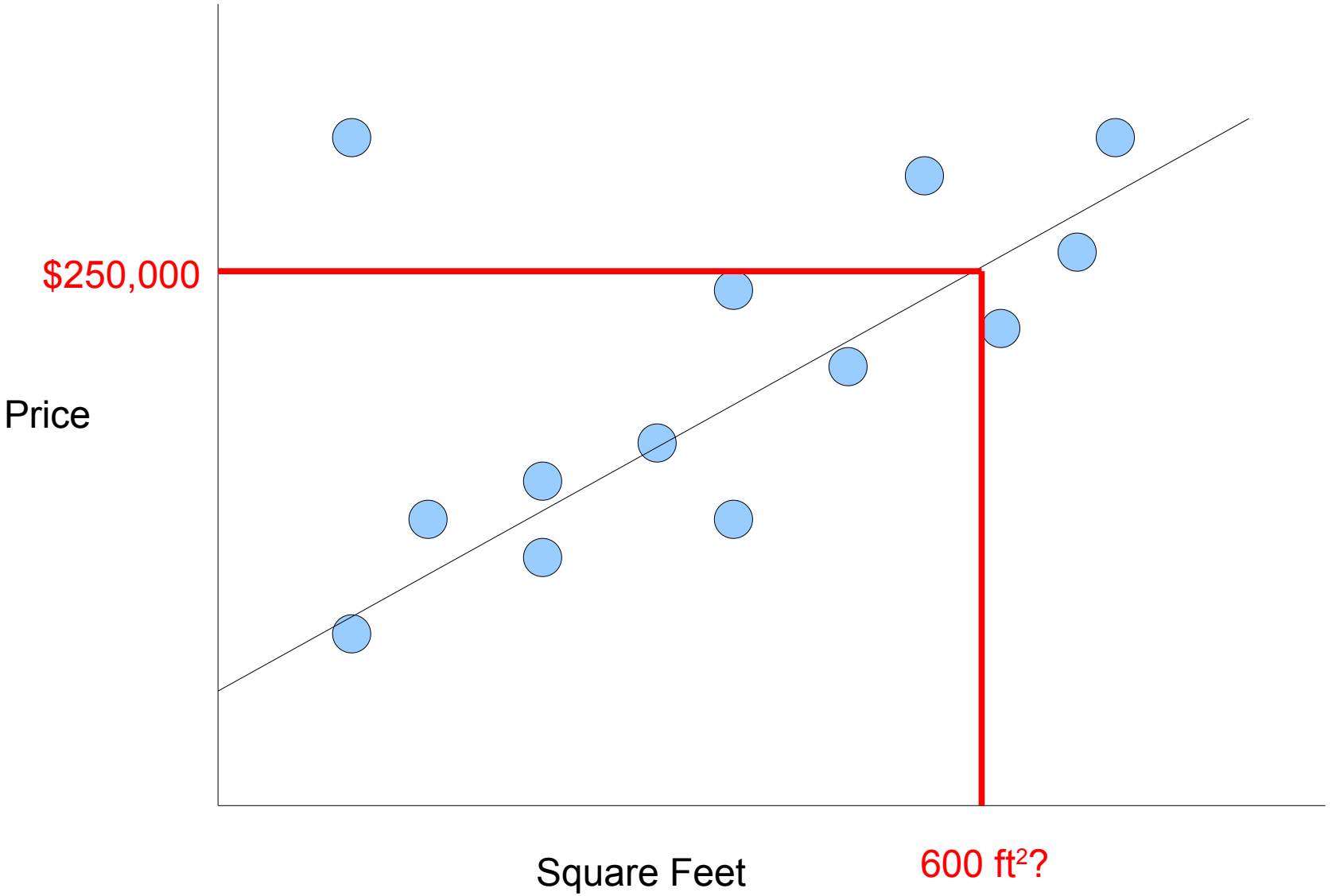
Buying Houses



Buying Houses



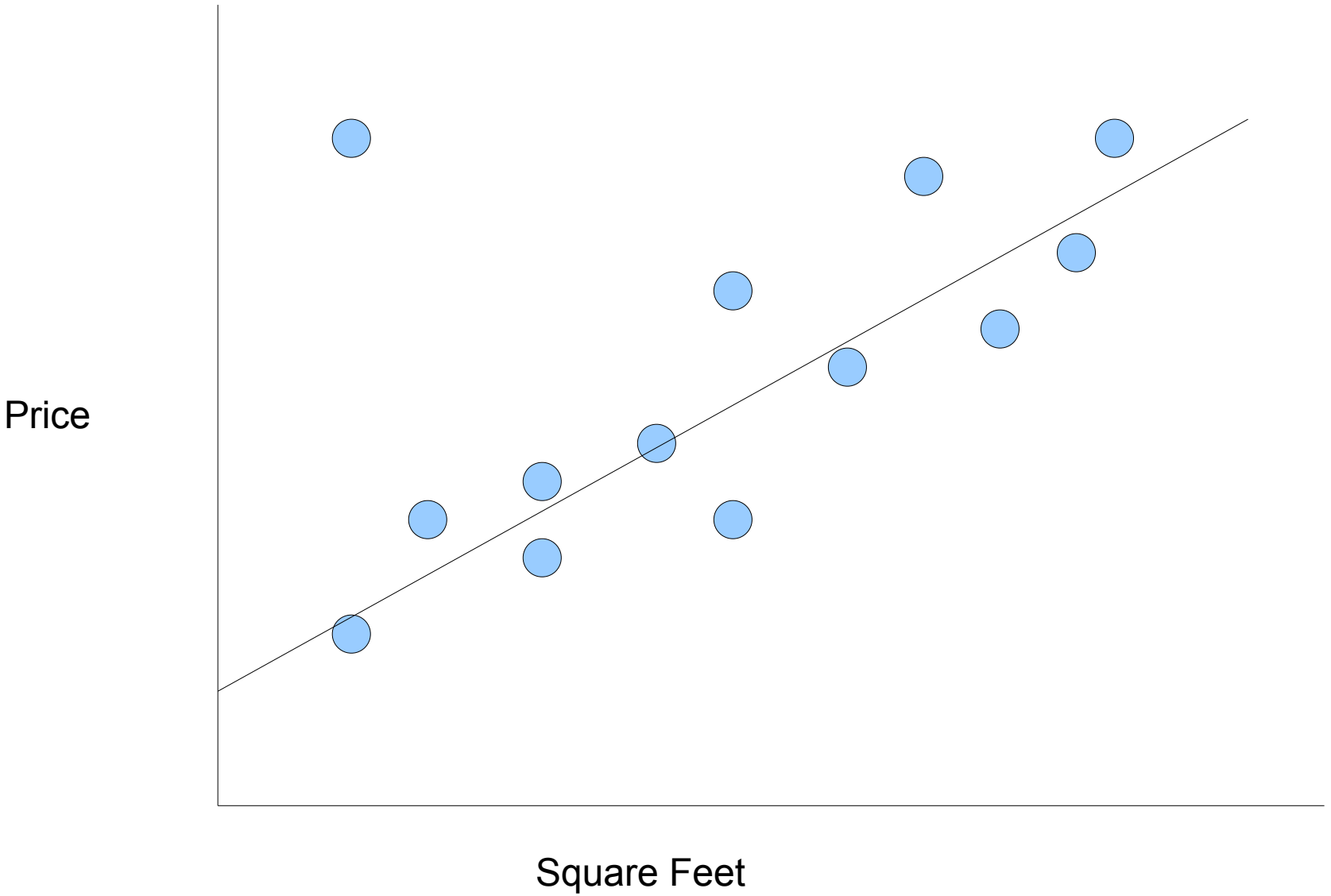
Buying Houses



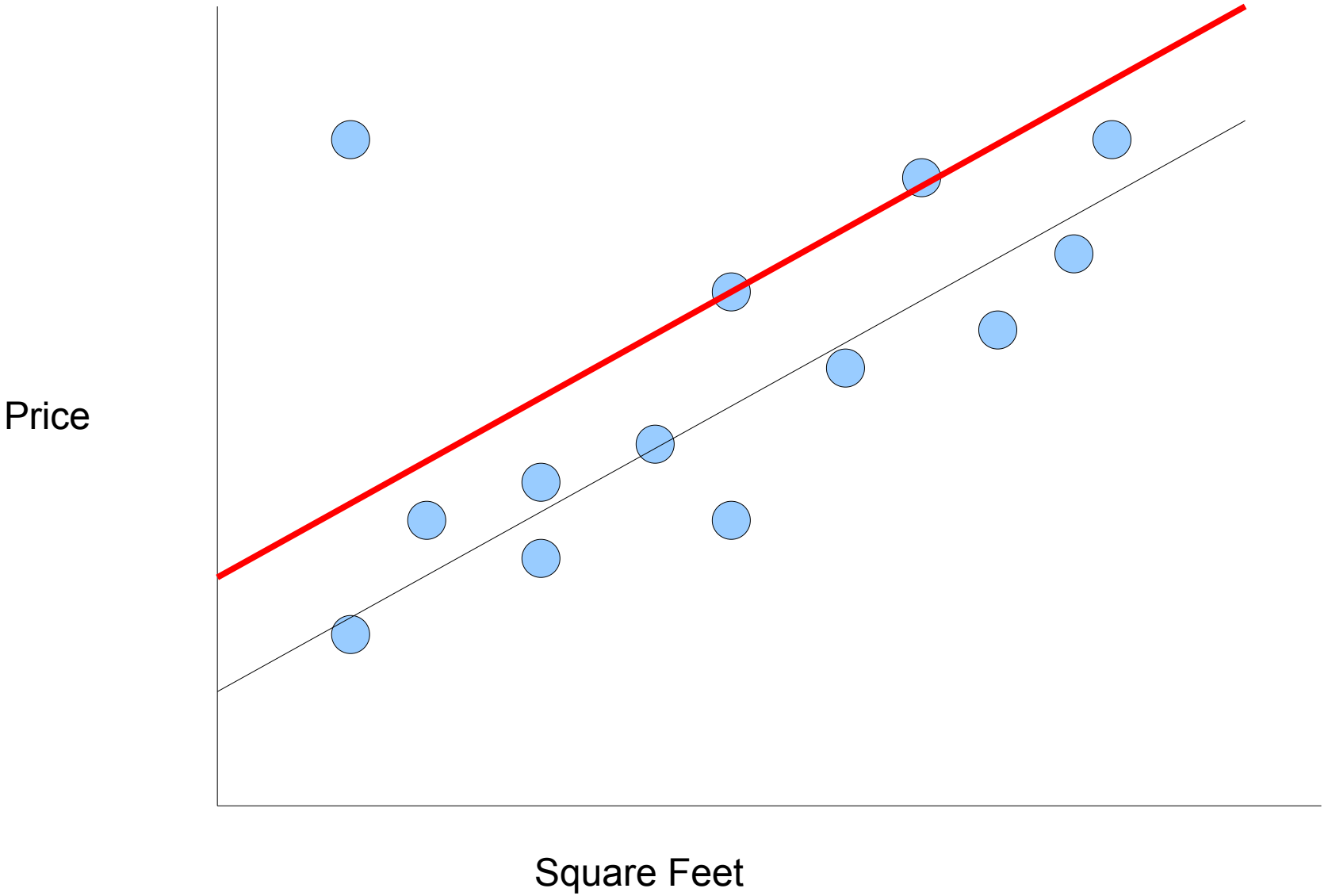
Buying Houses

- The line we picked “looked good”, but how do we decide what line to pick?

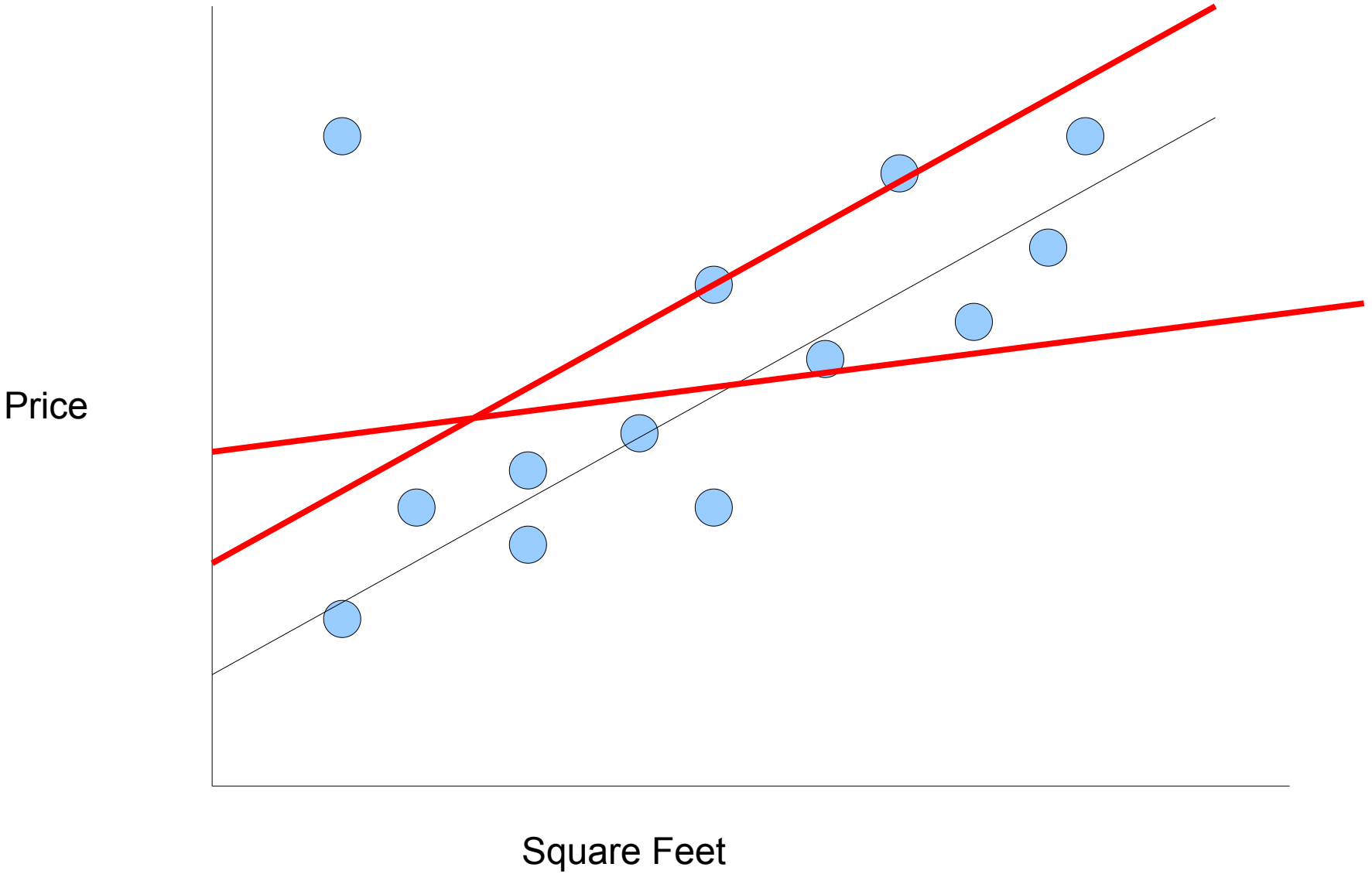
Buying Houses



Buying Houses



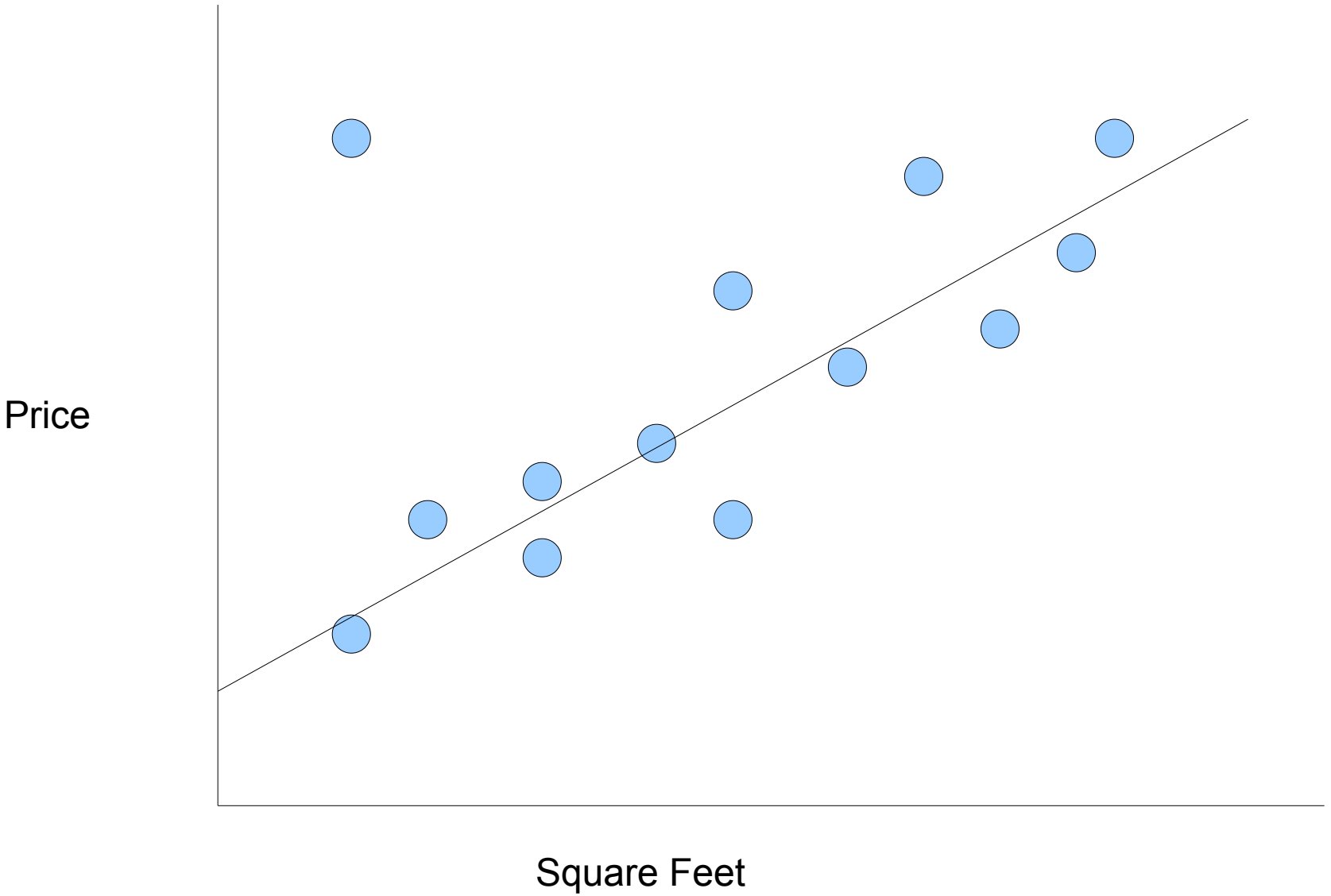
Buying Houses



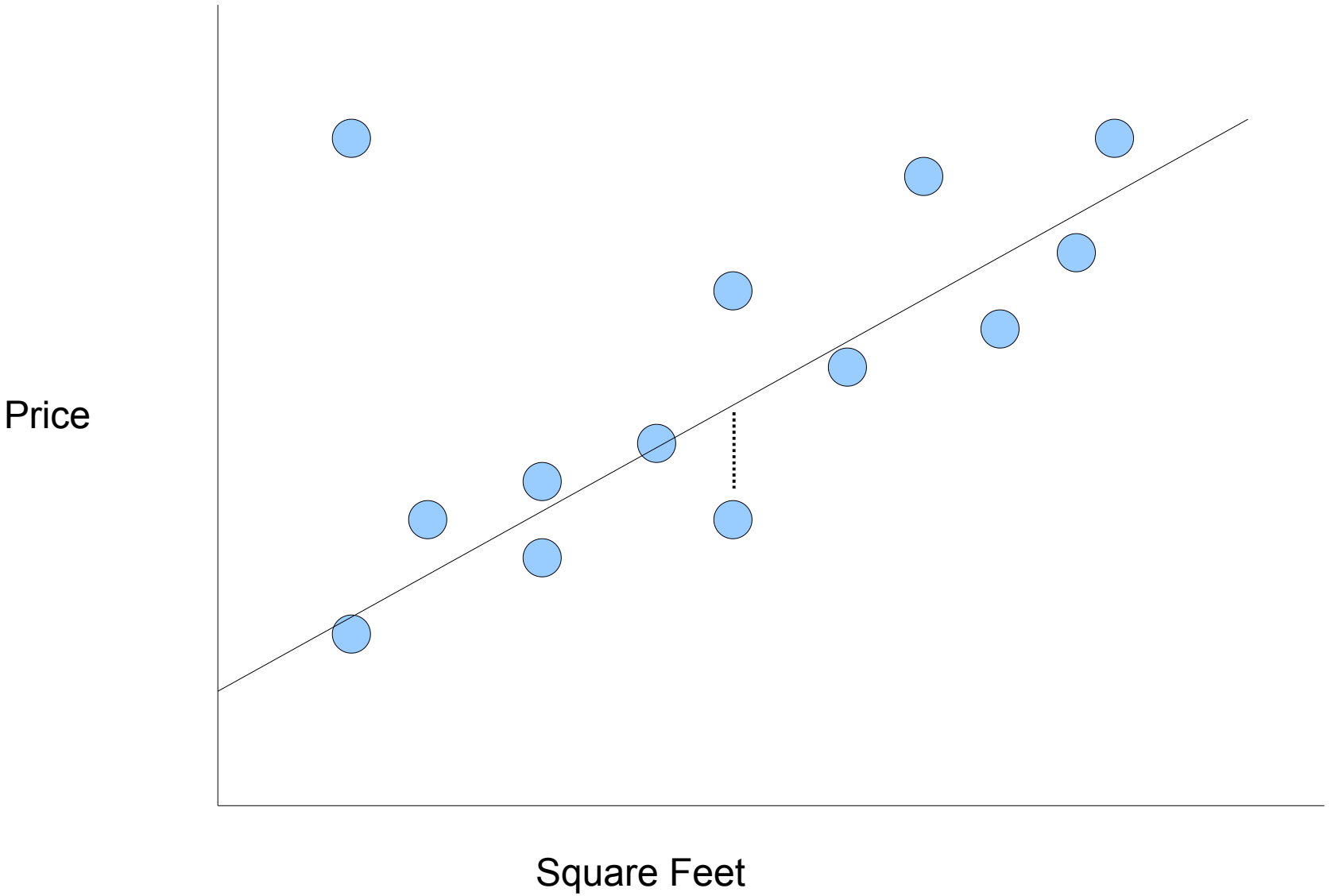
Picking Lines

- One idea is to minimize the **error** between our data points and line we choose.

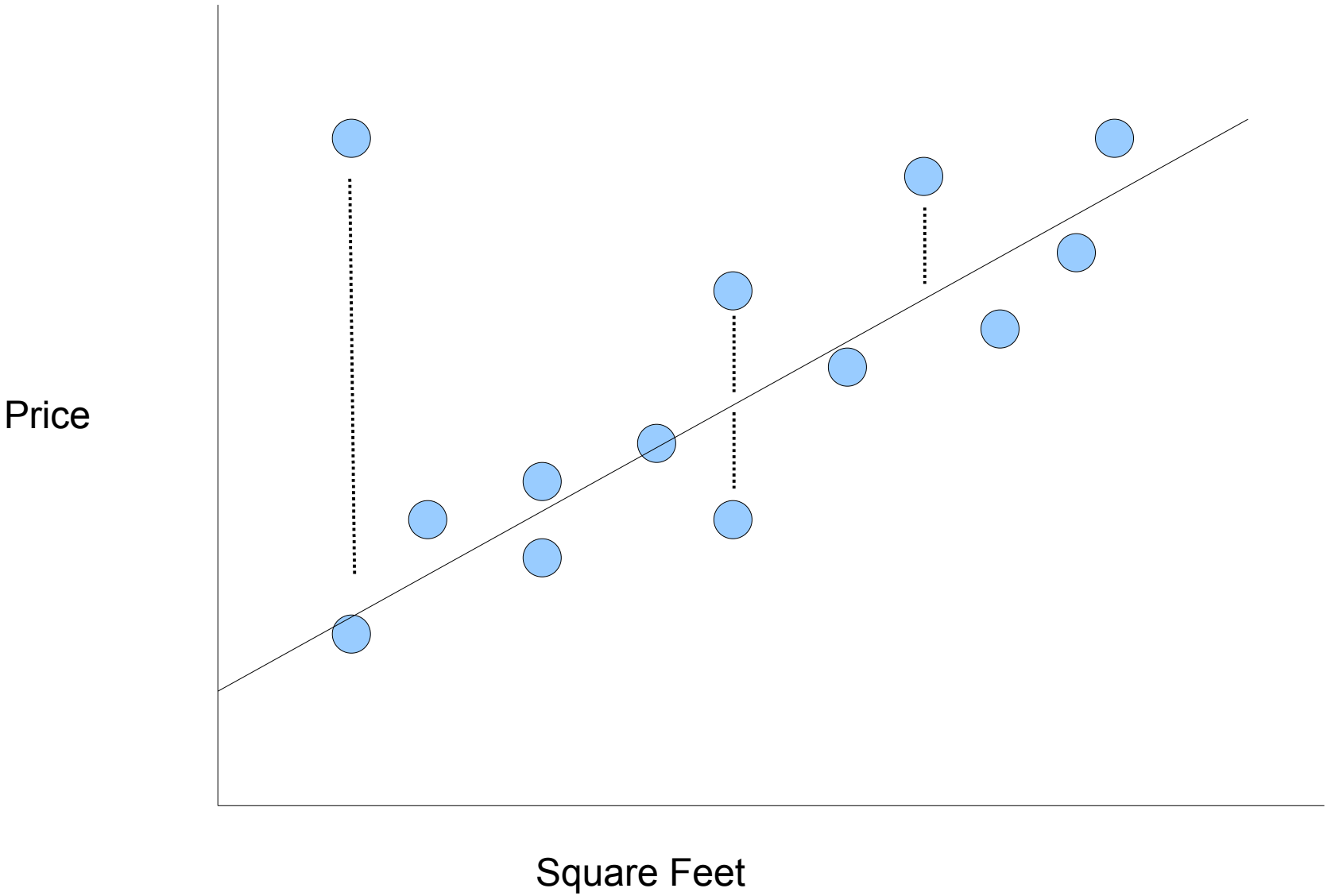
Buying Houses



Buying Houses



Buying Houses



Picking Lines

- One idea is to minimize the **error** between our data points and line we choose.
  - Error = sum of the squared distances between the line and our data points
- How do formulate this mathematically?

Picking Lines

- The function for a line is:

$$f(x) = ax + b$$

- Let's represent our data points symbolically as:

$$square\ footage = x_n$$

$$price = y_n$$

- The error between one of our data points and our prediction for that data point is:

$$error = (y_n - f(x_n))^2$$

$$total\ error = \sum (y_n - f(x_n))^2$$

Picking Lines

- The overall error of our line is:

$$total\ error = \sum \left(y_n - f\left(x_n \right) \right)^2$$

- Now that we have a function for our error, we can evaluate the quality of a line we choose
- This still doesn't tell us *how* to pick a line

Calculus to the Rescue!

- Answer: Some relatively straight forward calculus will give us a closed form solution for both **b** and **a**.
  - We can now pick the “best” line!
    - Are we done? (Hint: NO)

Calculus to the Rescue!

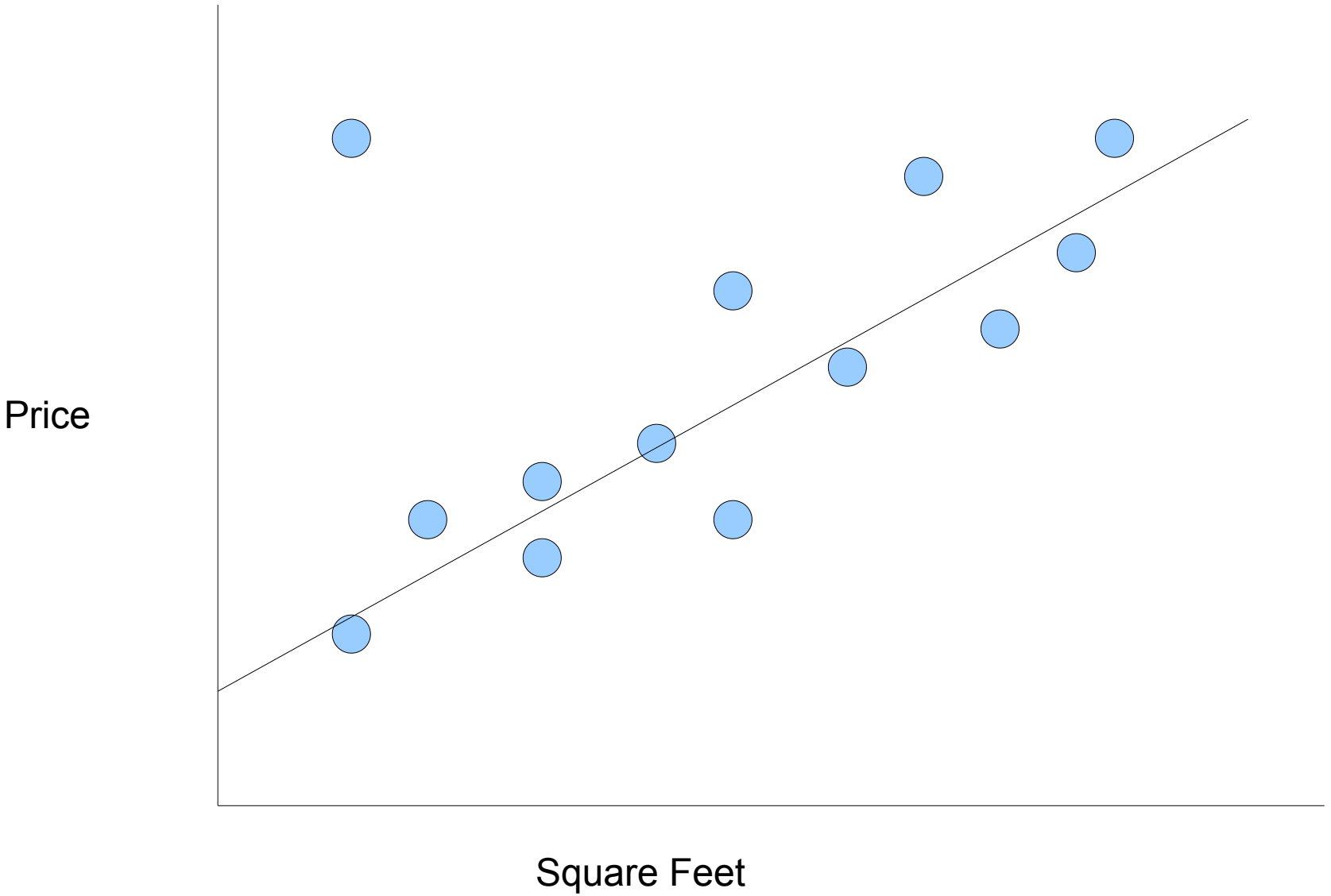
- No! 3 questions remain:

Calculus to the Rescue!

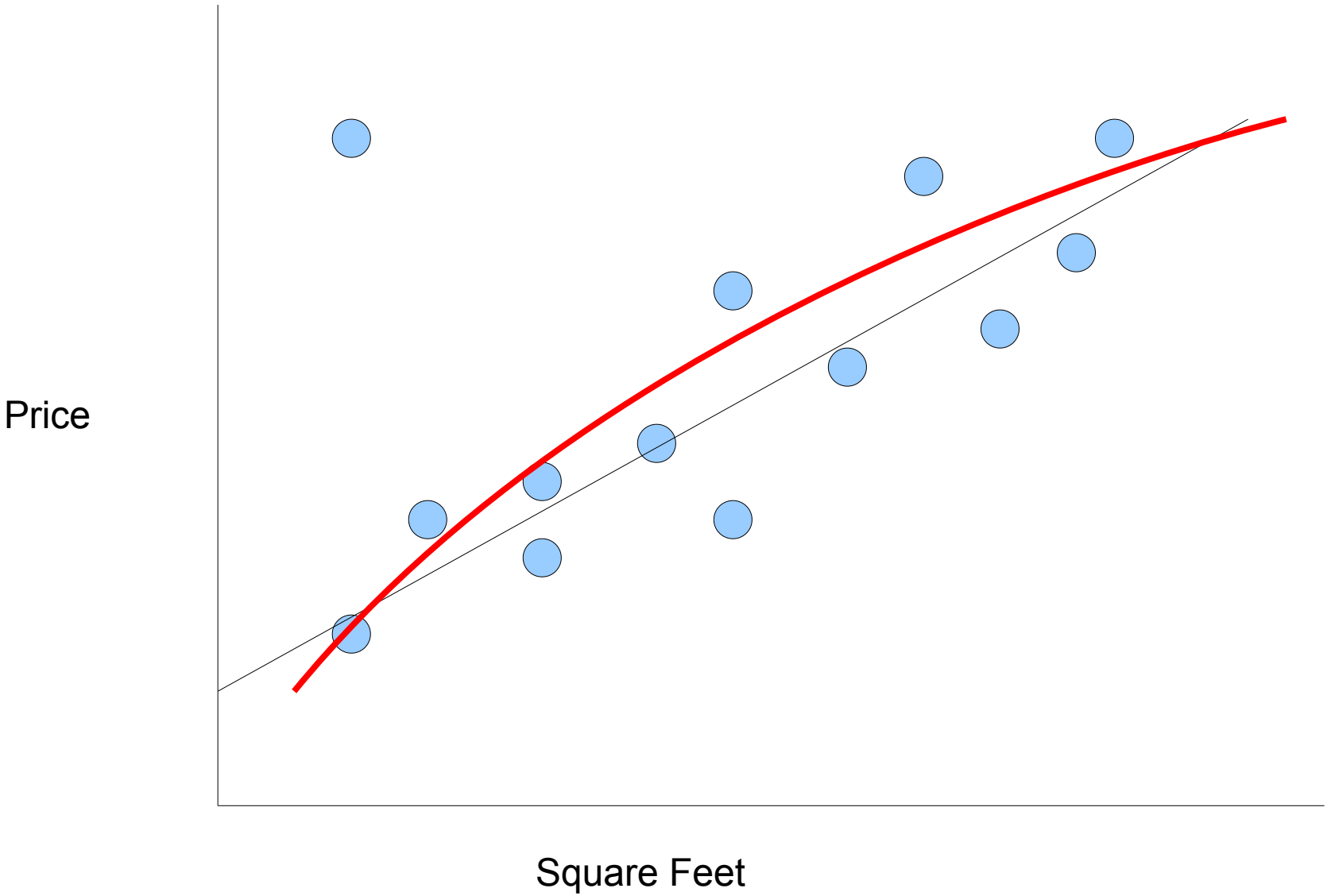
- No! 3 questions remain:

  1) Does our data *really* fit a line?

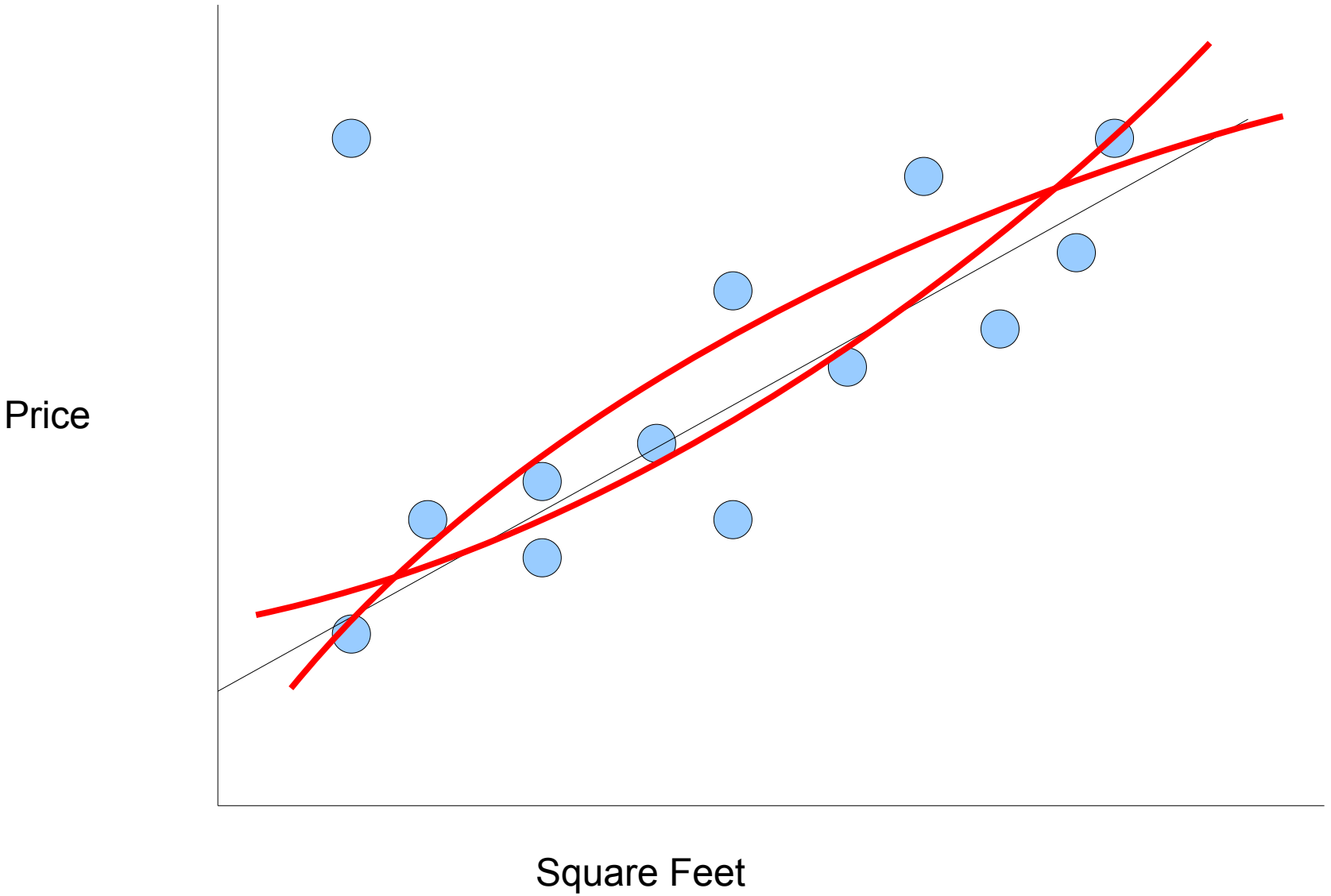
Buying Houses



Buying Houses



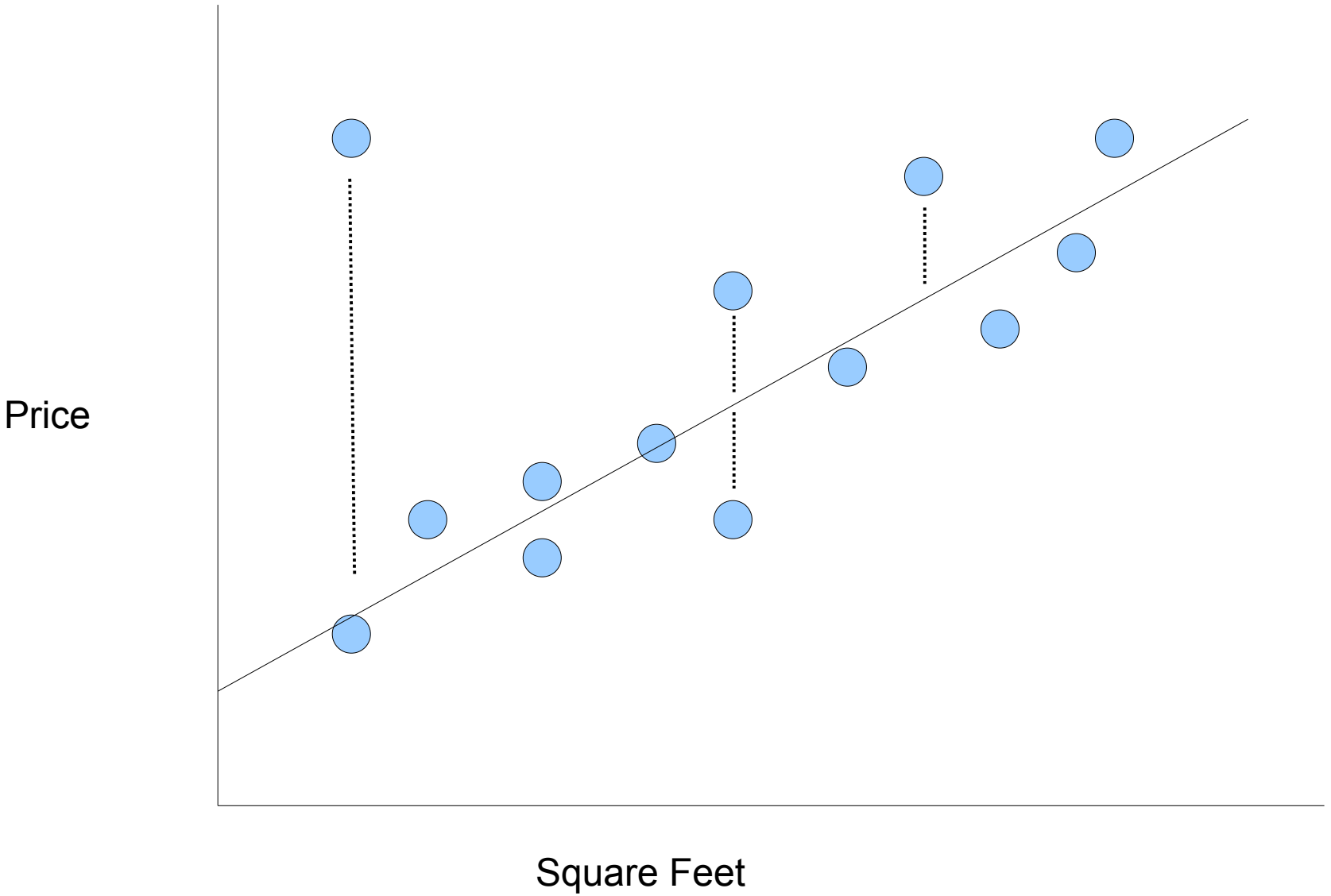
Buying Houses



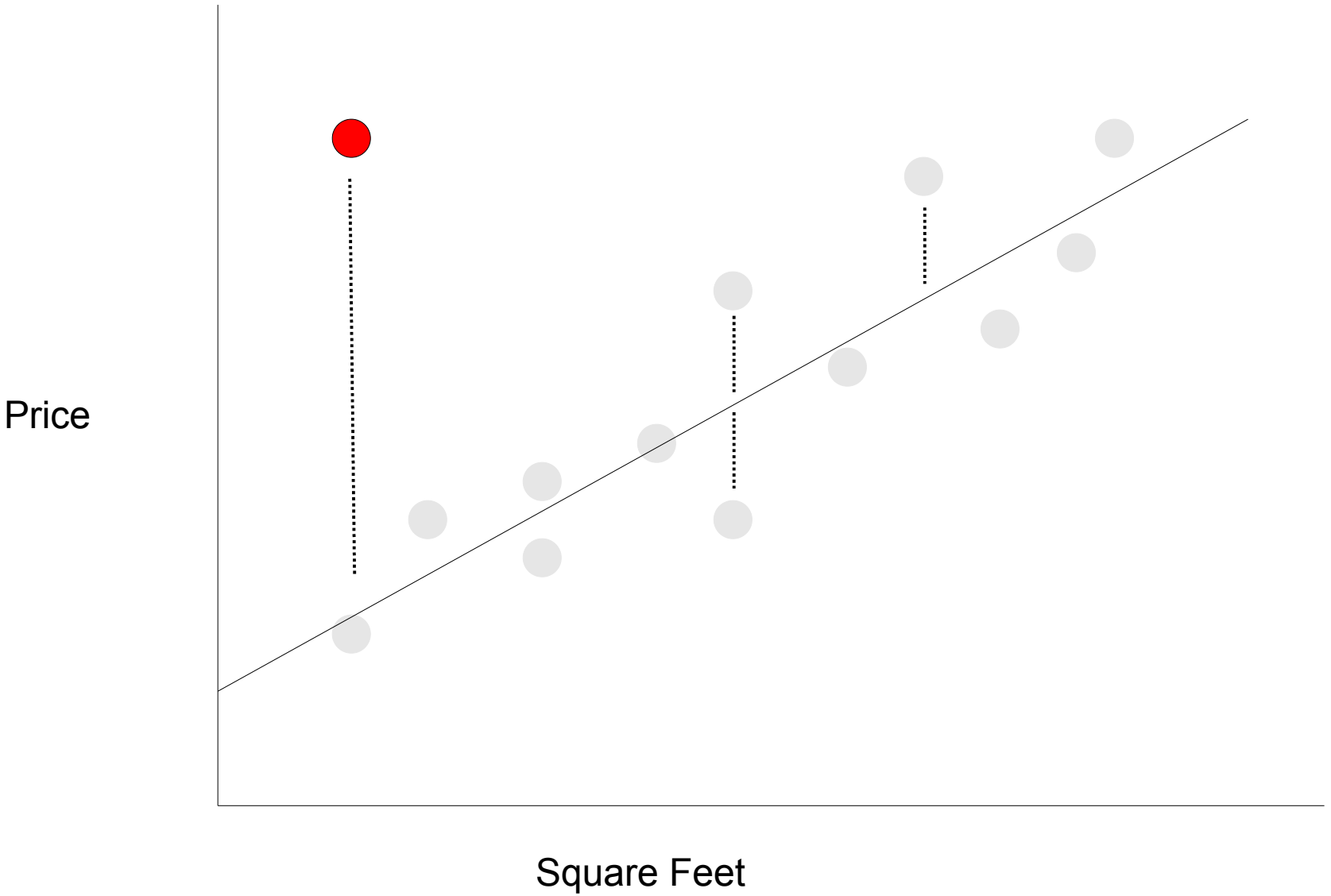
Calculus to the Rescue!

- No! 3 questions remain:
  1) Does our data *really* fit a line?
  2) Did we pick the right error function?

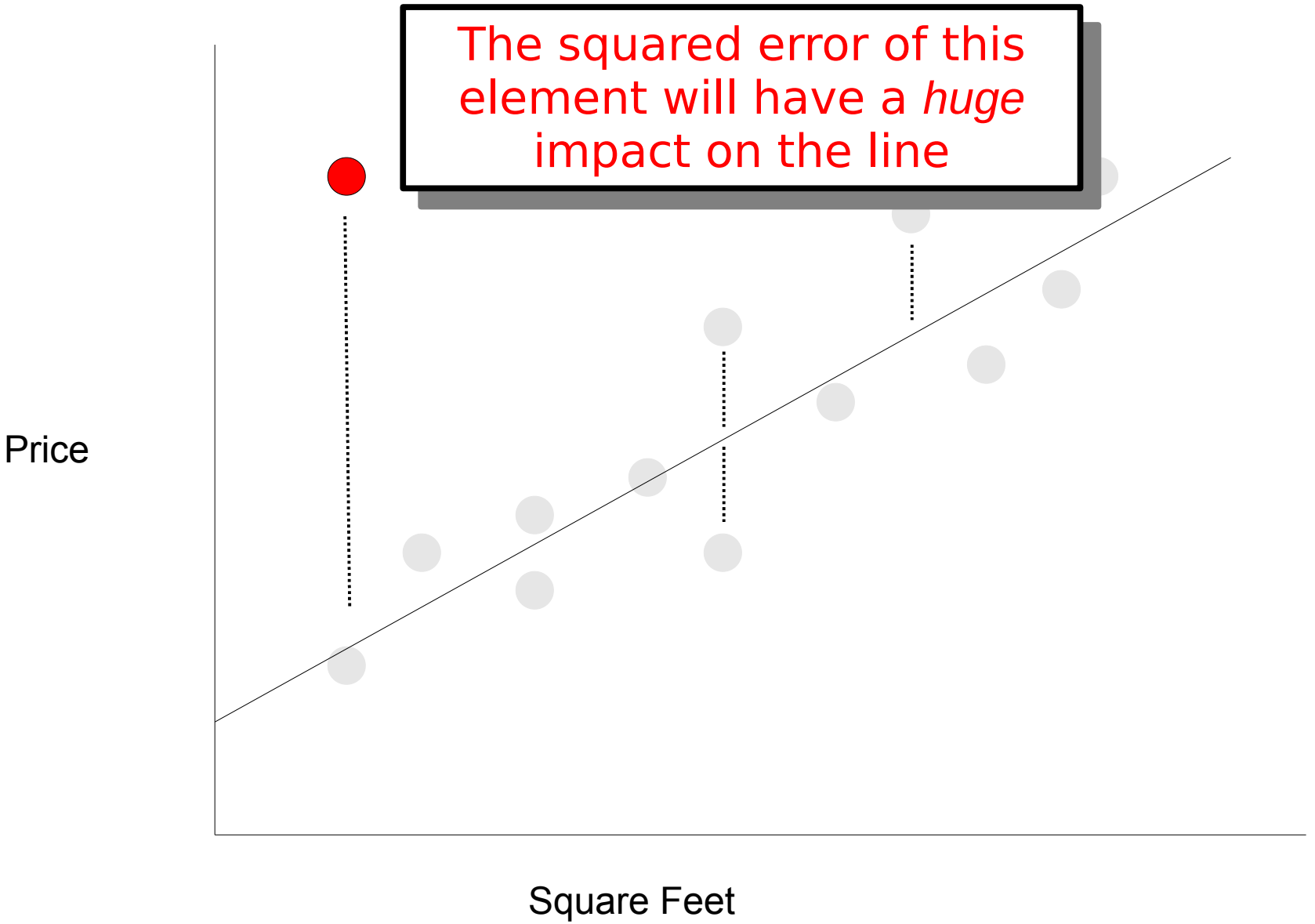
Buying Houses



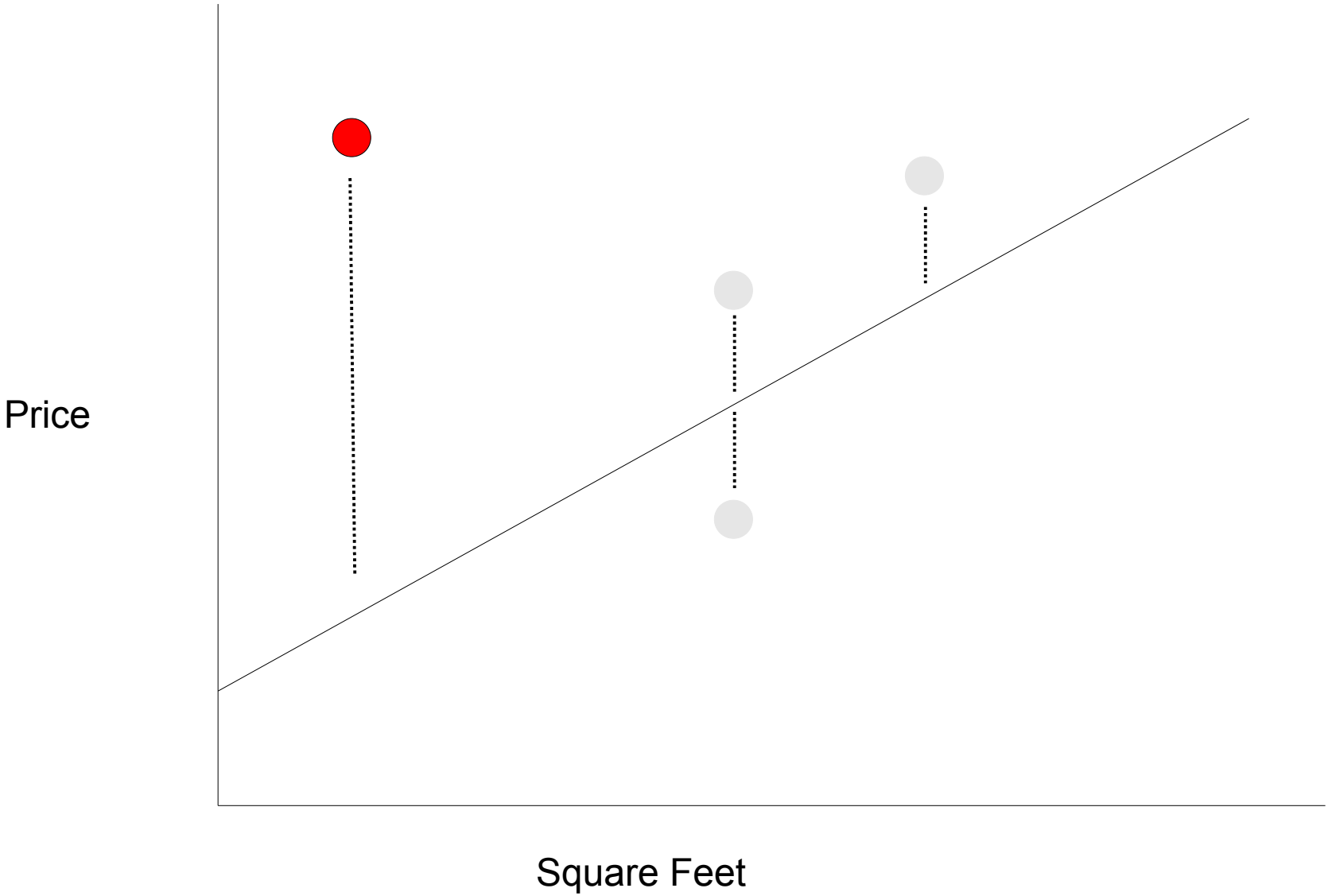
Buying Houses



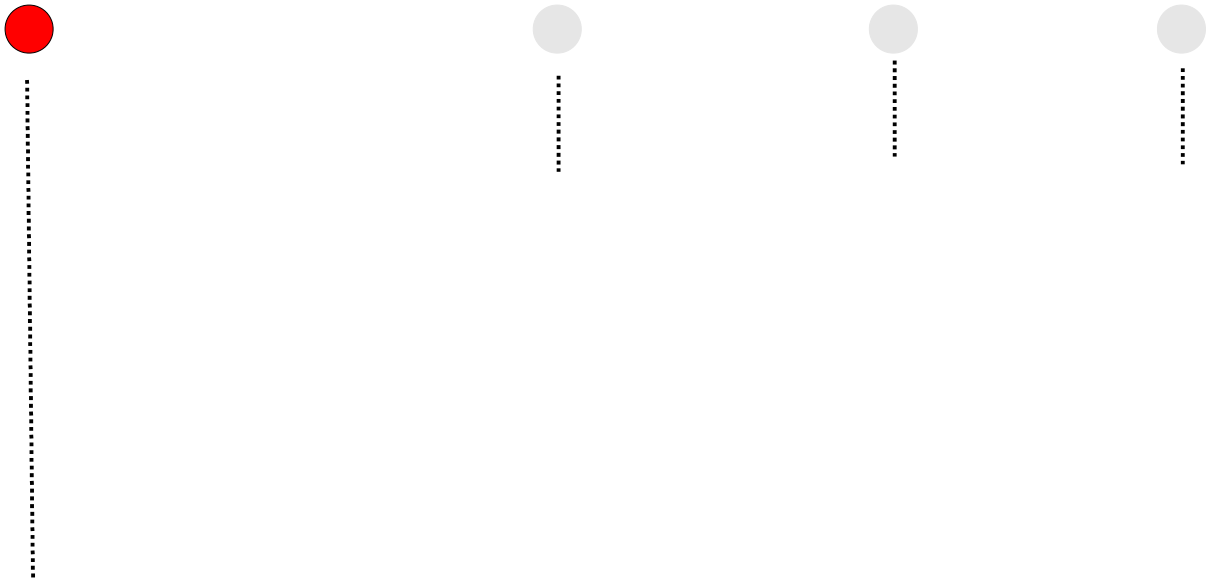
Buying Houses



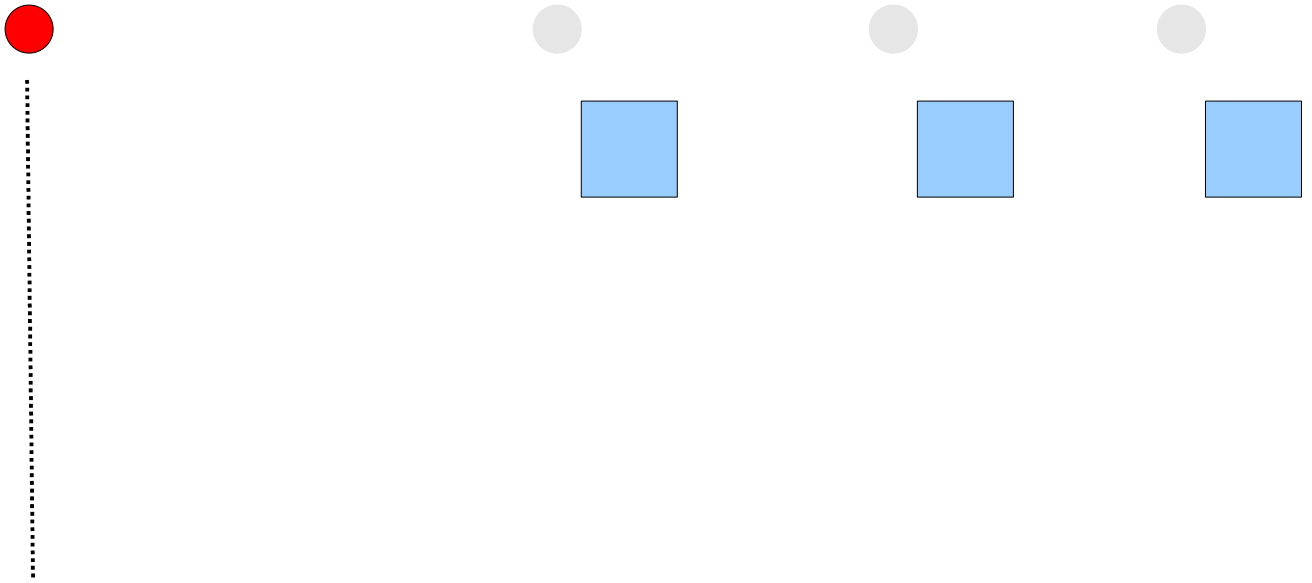
Visualizing Squared Error



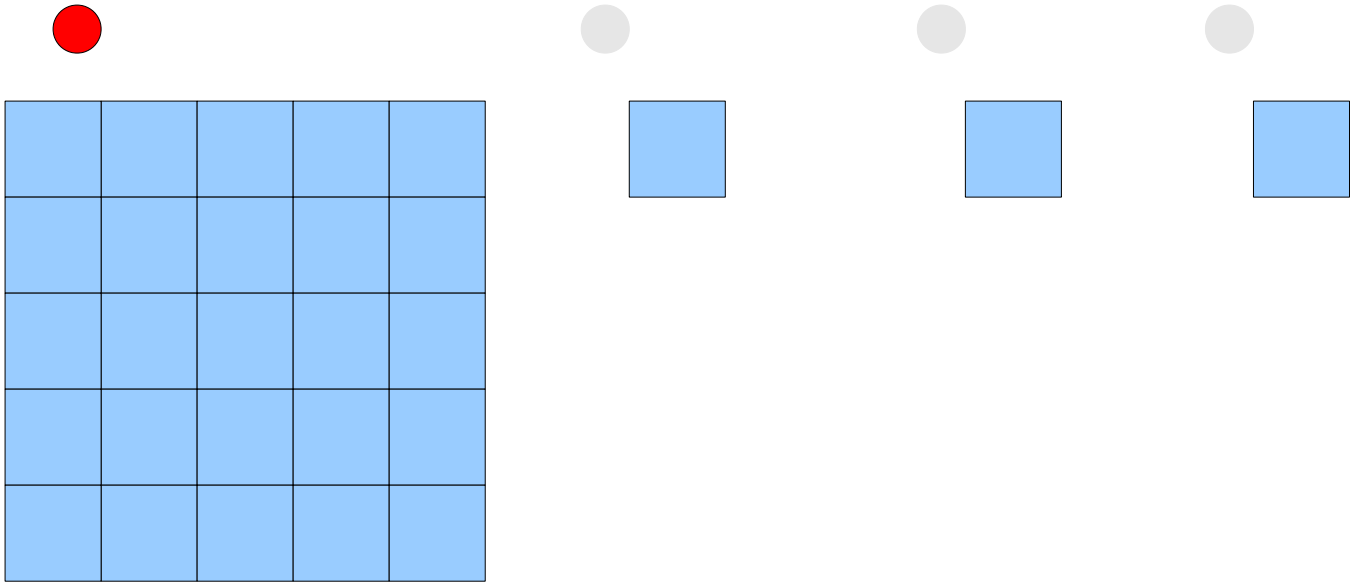
Visualizing Squared Error



Visualizing Squared Error



Visualizing Squared Error



Squared error puts *a lot* of emphasis on outliers.

Calculus to the Rescue!

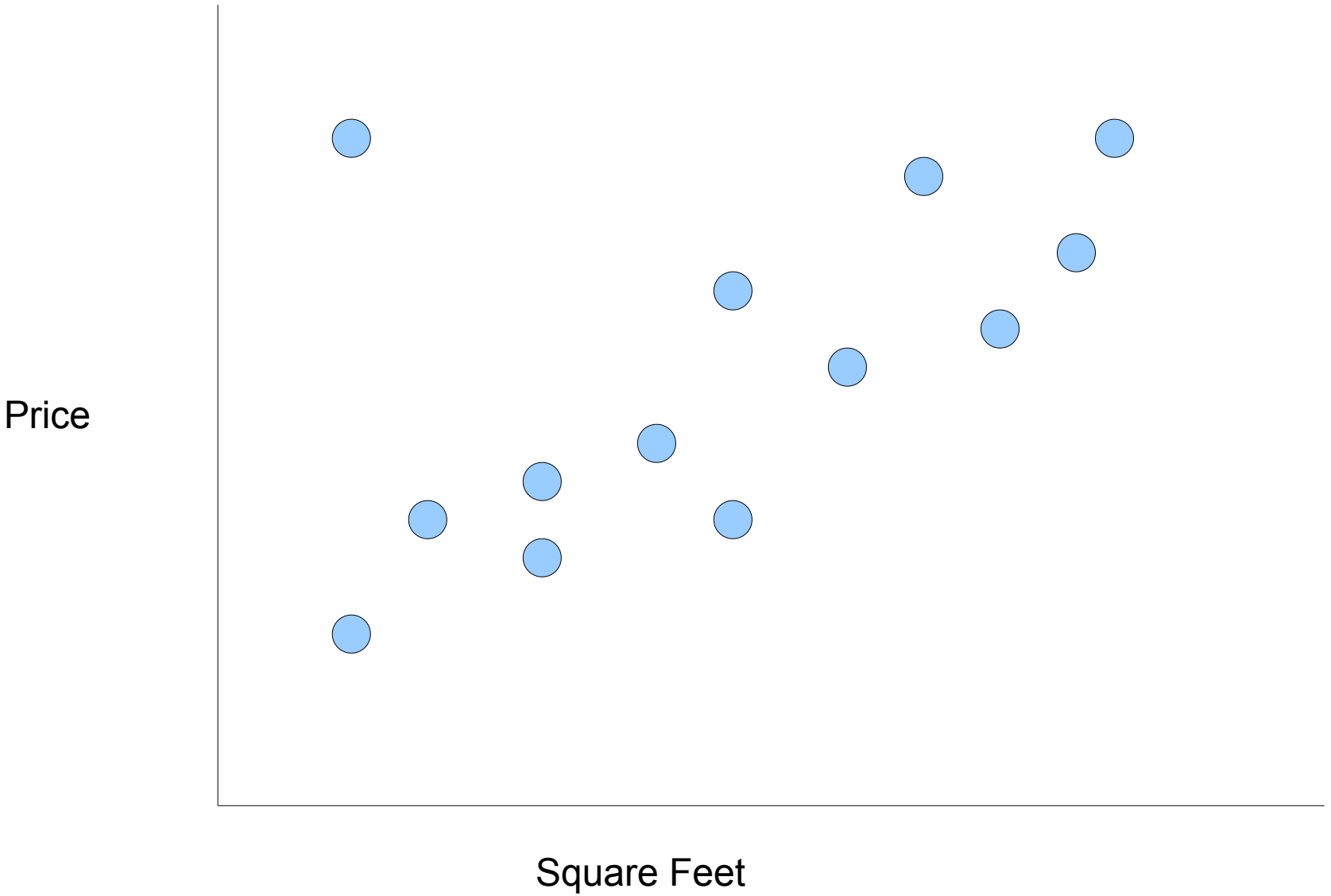
- No! 3 questions remain:

  1) Does our data *really* fit a line?

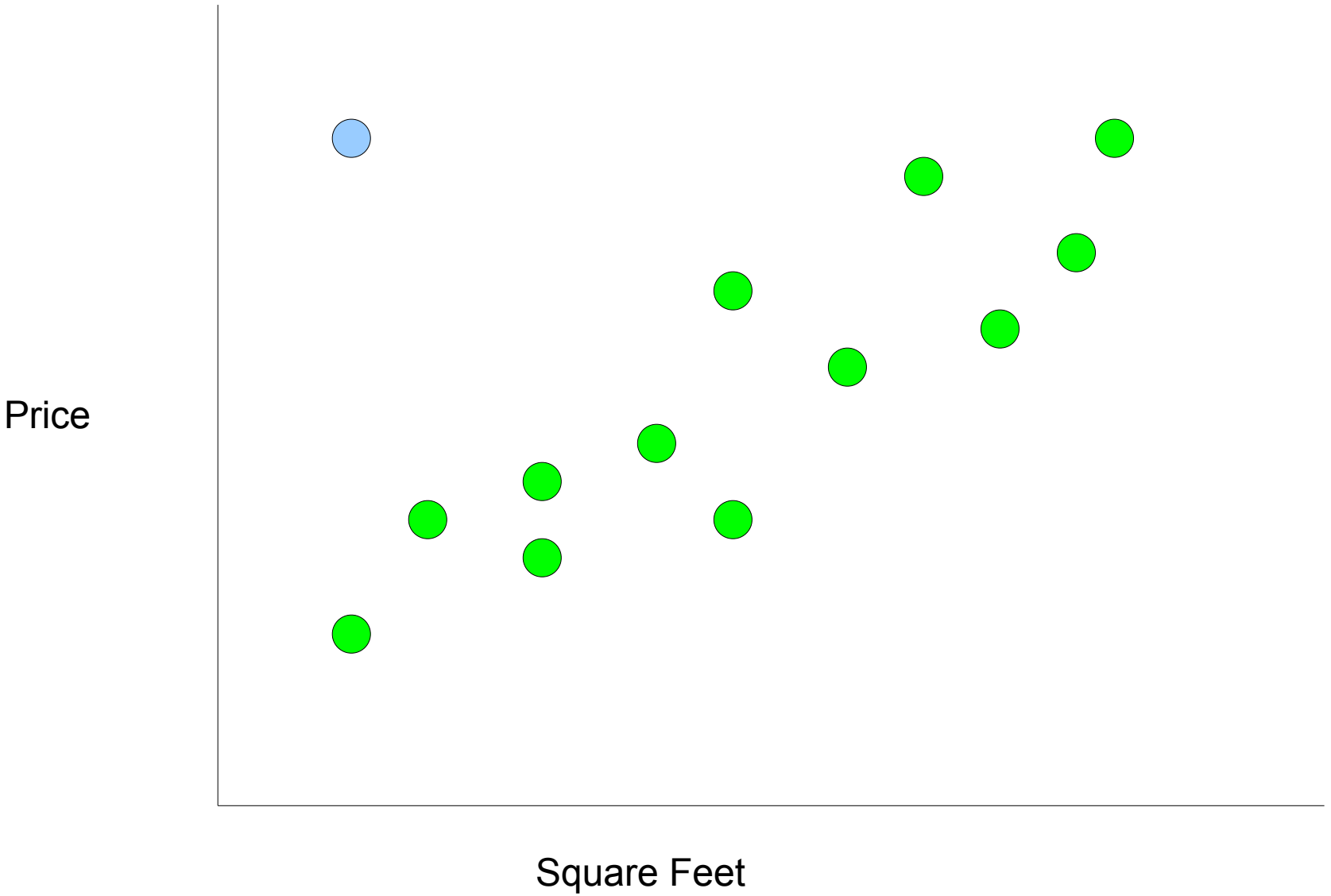
  2) Did we pick the right error function?

  3) Is square footage the best way to predict housing prices?

Buying Houses



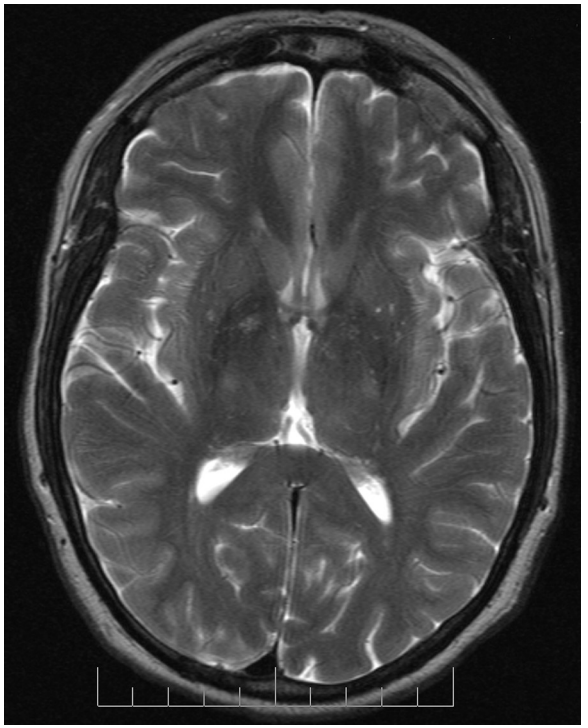
Buying Houses



Buying Houses

Supervised Learning

- **Supervised Learning** is a form of Machine Learning in which you have labeled training data and you train a model to predict these labels

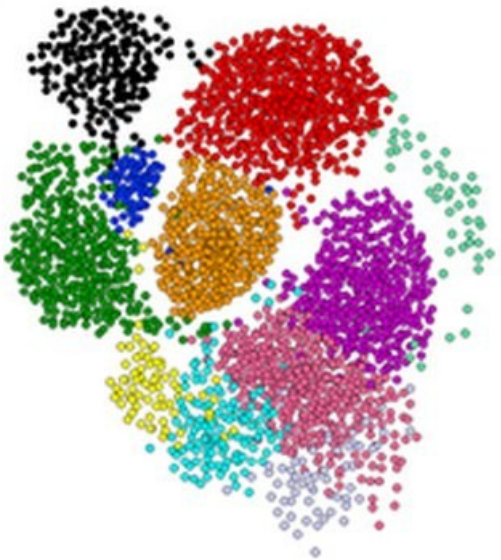


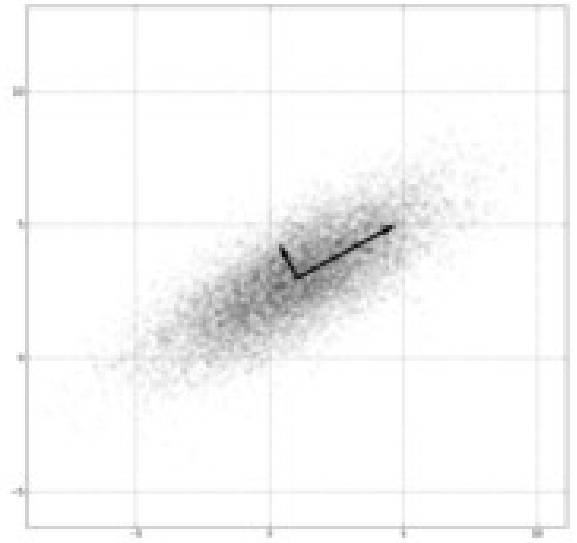
Alzheimer's?

YES/NO

Other Forms of Learning

- **Unsupervised Learning**: Given training data *without labels* tell me something interesting about the data



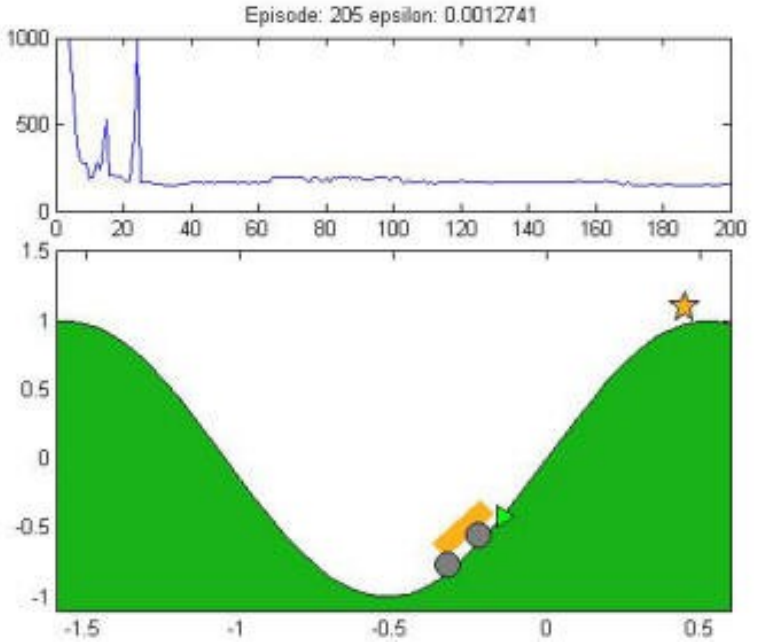


http://tagc.univ-mrs.fr/tagc/images/dputhier/tb2.jpg

http://upload.wikimedia.org/wikipedia/commons/thumb/1/15/GaussianScatterPCA.png/220px-GaussianScatterPCA.png

Other Forms of Learning

- **Reinforcement Learning**: Given an environment and an objective function, learn a model to make choices in the environment.



http://www.dacya.ucm.es/jam/images/MountainCar.jpg

Machine Learning

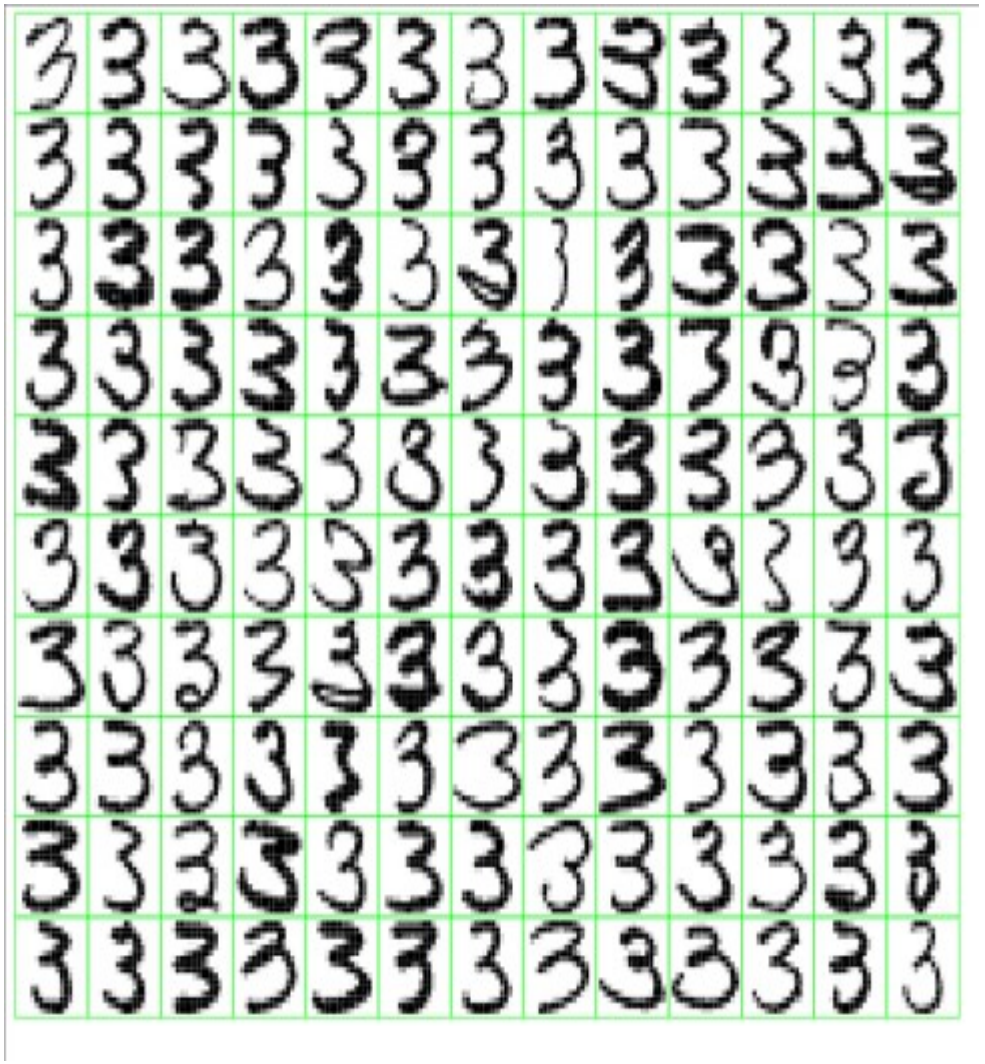
- When posed with a problem in Machine Learning you have to answer 3 questions:
  1) How do I represent my data?
     - Square footage? Number of bathrooms? Quality of nearby schools?
  2) What type of function do I want to learn?
     - Line? Parabola?
     - What is our Error function?
  3) How do I optimize my model?
     - Gradient Ascent? Solve directly?
     - Matrix factorization?

Other Questions

- What I don't have very much training data?
  - What if you have extra “unlabeled” data?
    - e.g. MRI scans of undiagnosed patients
  - What if you have other types of supervision?
    - e.g. I **know** these two instances belong to the same cluster, I just don't know which cluster they belong to.
  - What if you know something about the function you're going to learn?
    - e.g. Most of the features are not useful.

Introduction to Perceptrons

Handwriting Analysis

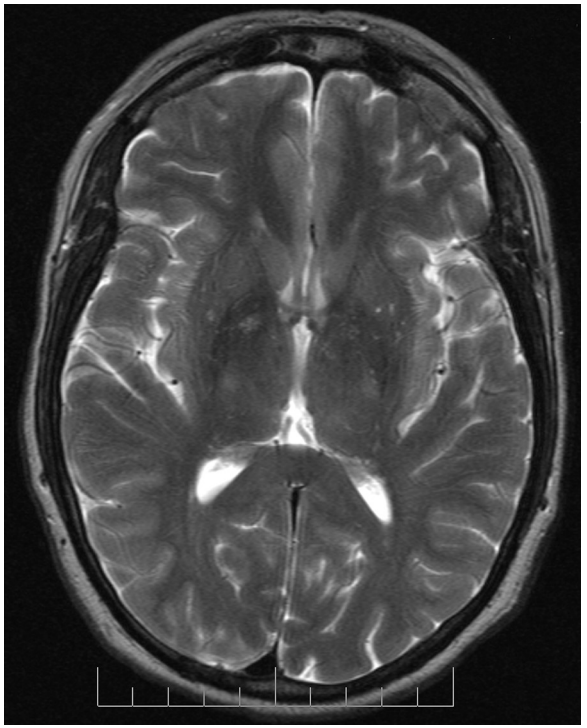


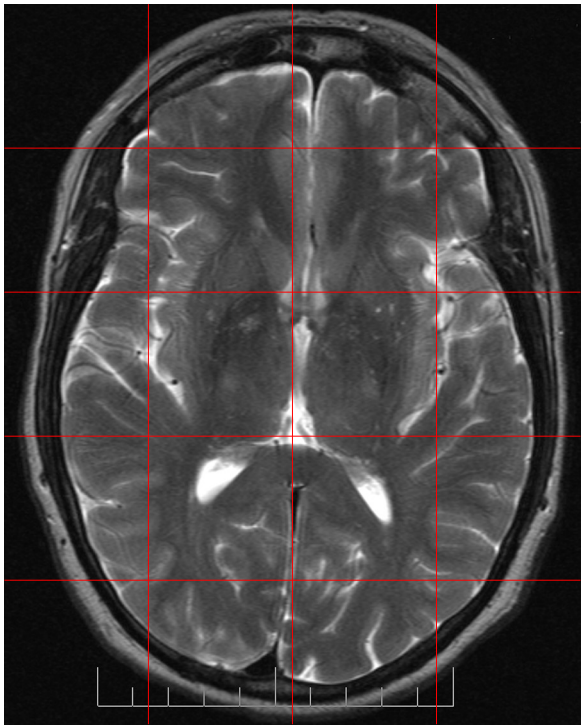
- Train a computer to recognize images of handwritten numbers 0 – 9.
- Large training and test set available (MNIST Handwritten Digit Database)

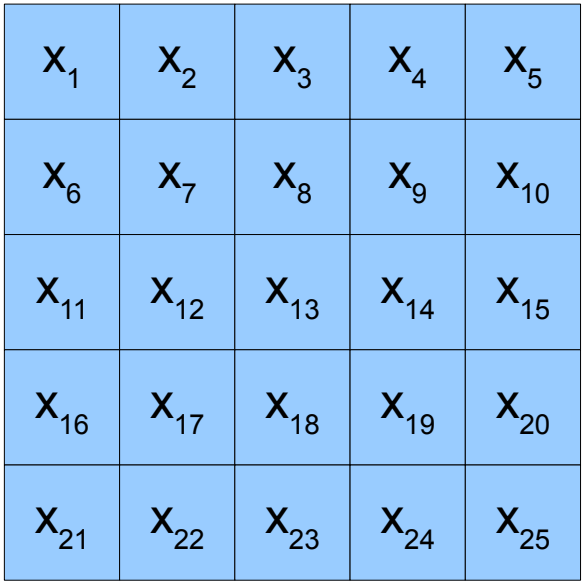
Classification Problem

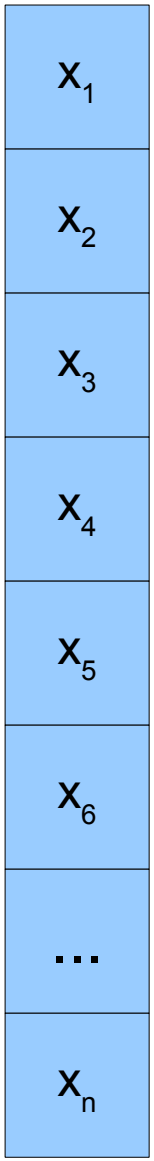
- The problem we are considering is a **classification problem**: given an image, we need to figure what **class** it belongs to (0, 1, 2, 3, etc.).
- Just to be concrete, we're going to write a program with:
  - Input: Image of a handwritten digit
  - Output: Prediction for what digit the image is

Question 1: How do we represent an image?

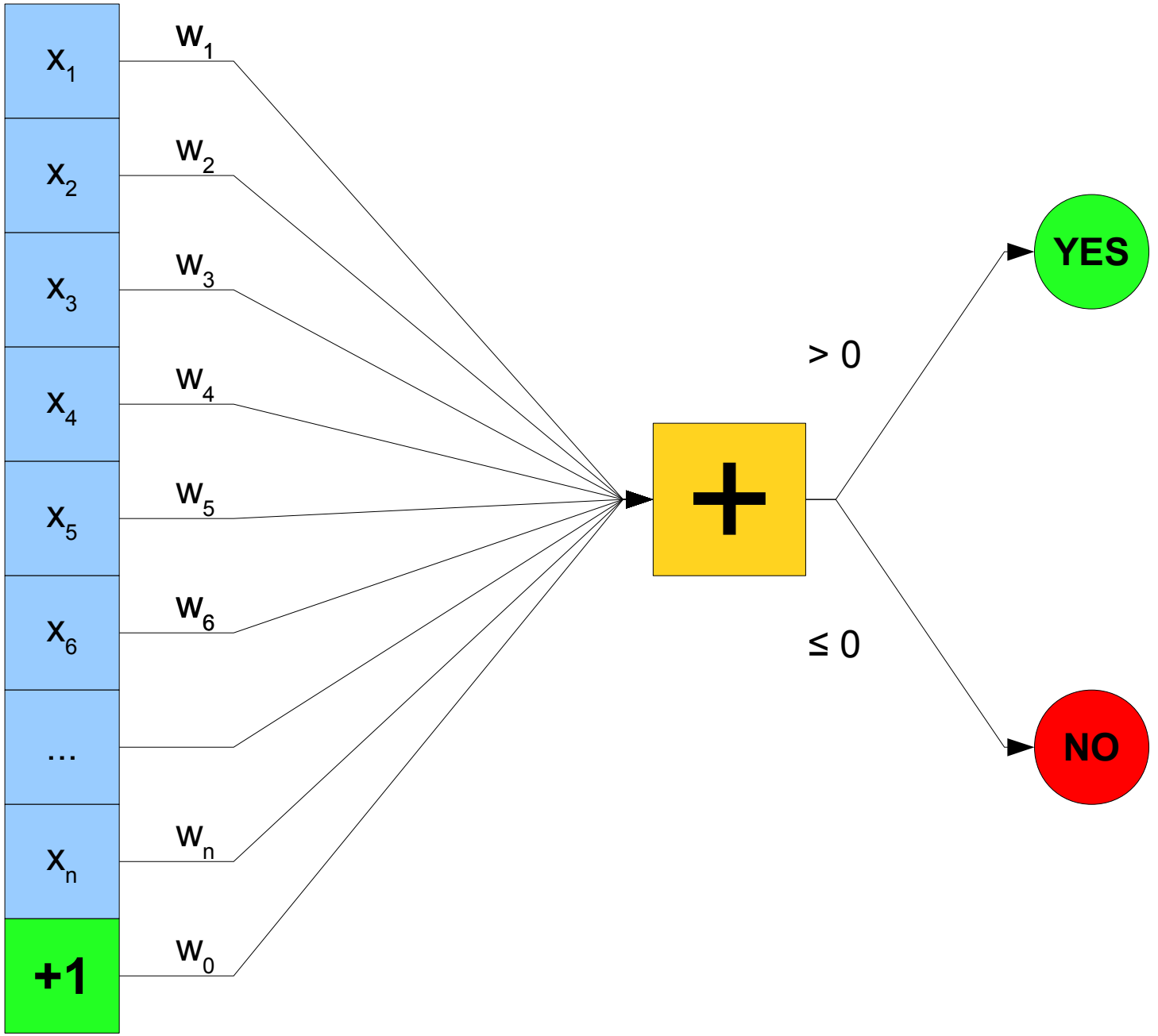


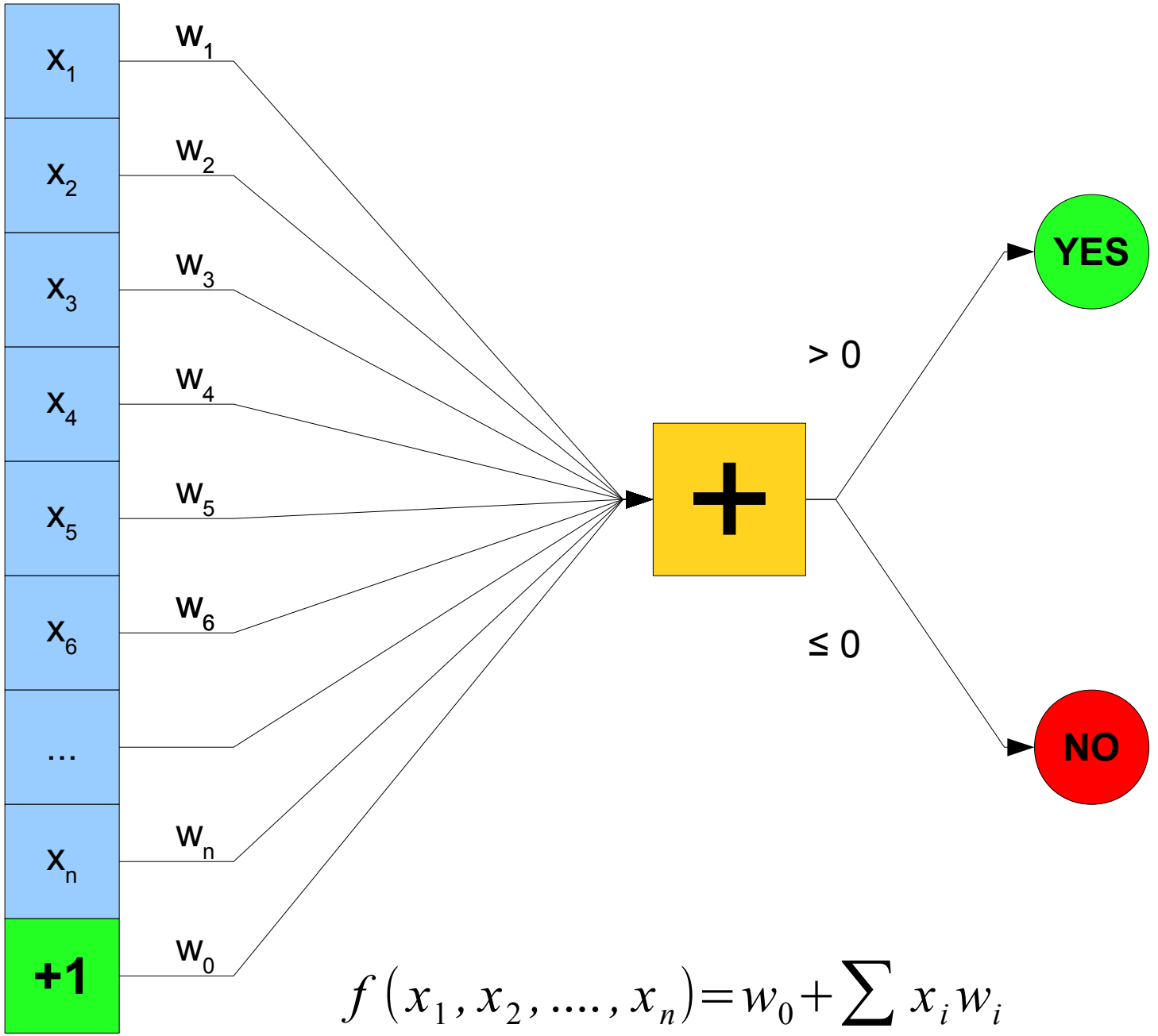



x_1 x_2 x_3 x_4 x_5
x_6 x_7 x_8 x_9 x_{10}
x_{11} x_{12} x_{13} x_{14} x_{15}
x_{16} x_{17} x_{18} x_{19} x_{20}
x_{21} x_{22} x_{23} x_{24} x_{25}

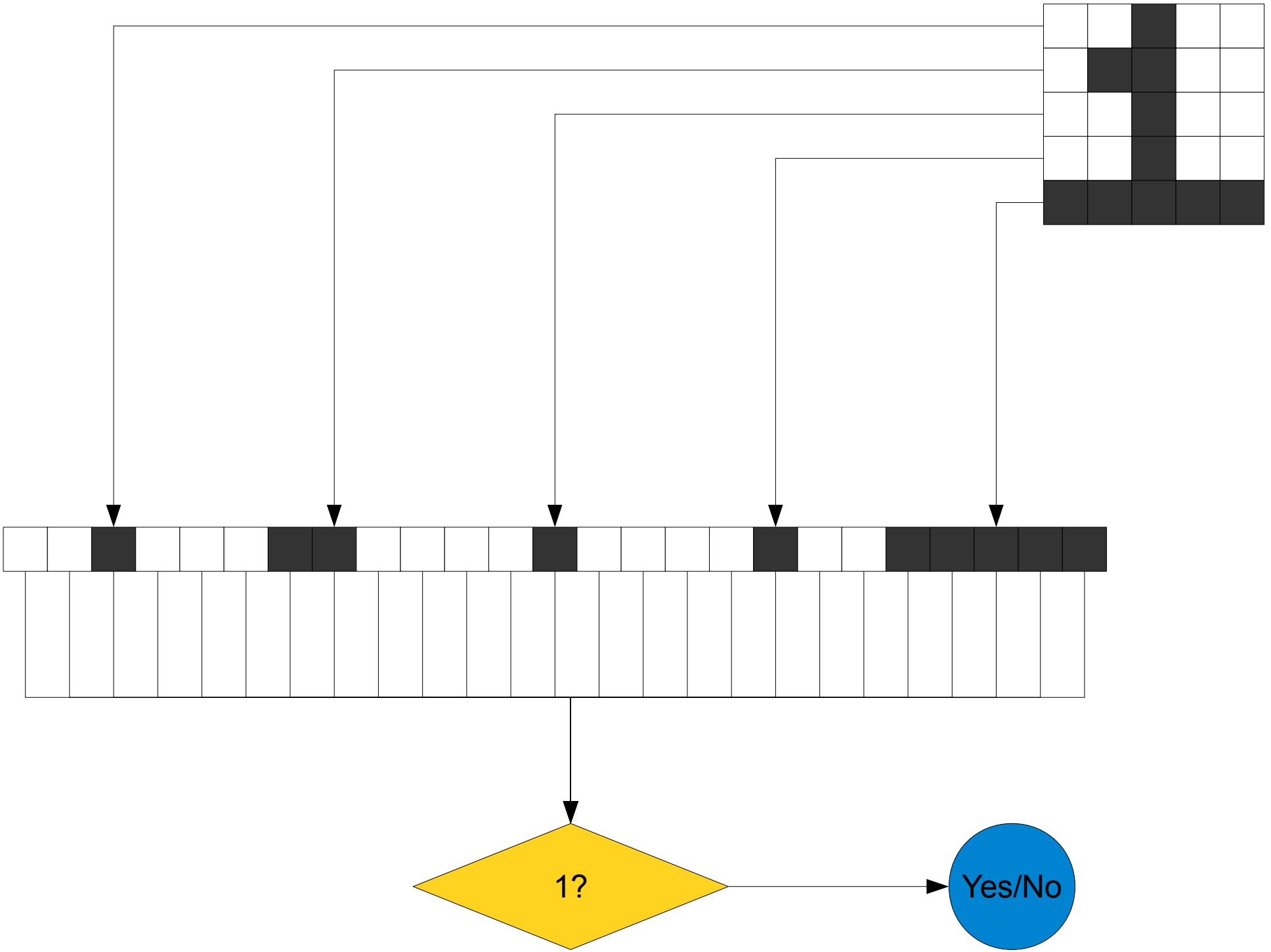

x_1
x_2
x_3
x_4
x_5
x_6
...
x_n

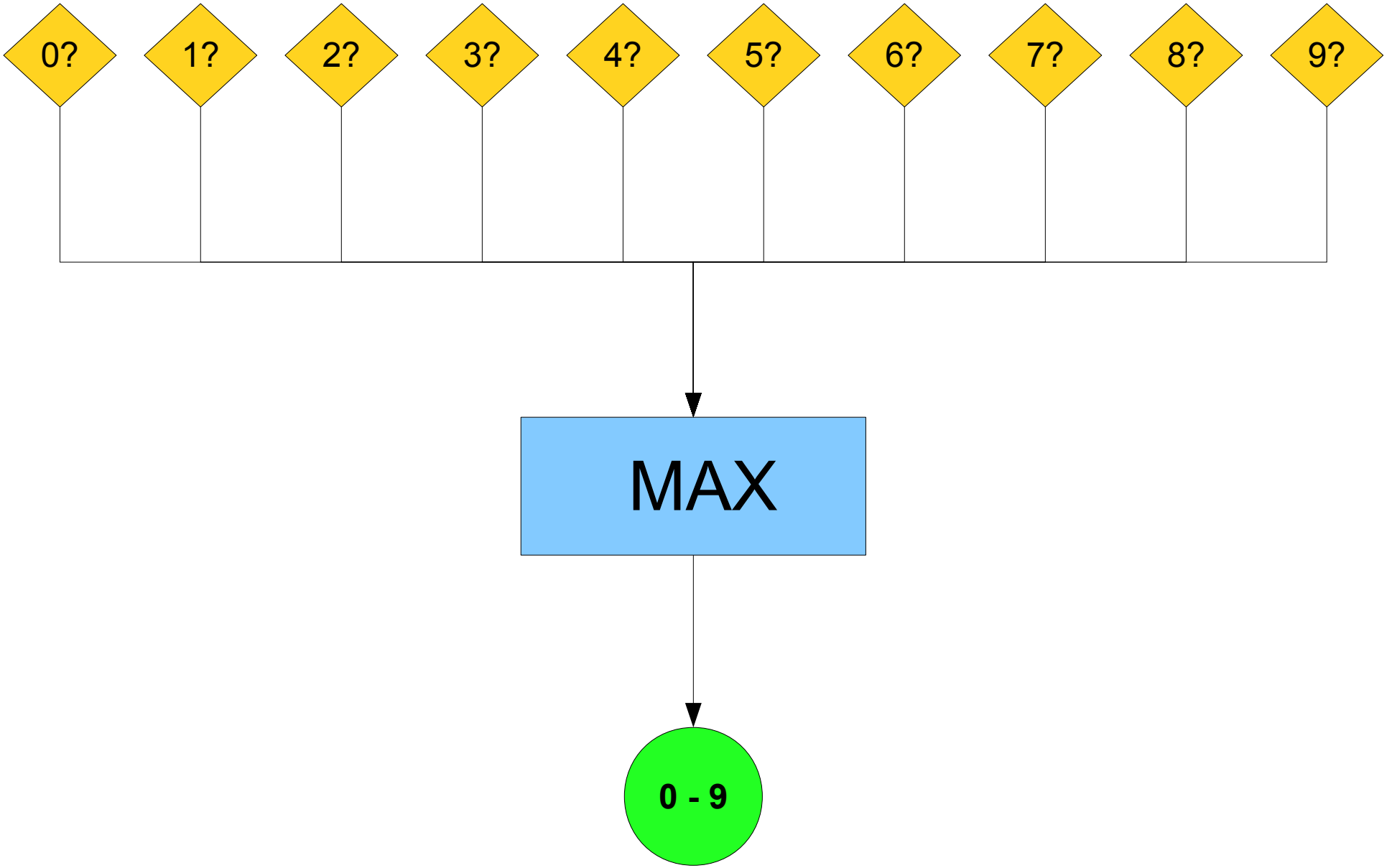
Question 2: What function are we trying to optimize?


x1
x2
x3
x4
x5
x6
...
xn
+1
w1
w2
w3
w4
w5
w6
wn
w0
+
> 0
≤ 0
YES
NO



$$f(x_1, x_2, \ldots, x_n) = w_0 + \sum x_i w_i$$


1?
Yes/No


0?
1?
2?
3?
4?
5?
6?
7?
8?
9?
MAX
0 - 9

Question 3: How do we optimize our perceptrons?

How do we choose good values for $w_0 \ldots w_n$?

One Approach

- **Train** the perceptron on valid data.

One Approach

- **Train** the perceptron on valid data.
- For each data point (image of a digit):

One Approach

- **Train** the perceptron on valid data.
- For each data point (image of a digit):
  - Ask the perceptron what it thinks.

One Approach

- **Train** the perceptron on valid data.
- For each data point (image of a digit):
  - Ask the perceptron what it thinks.
  - If correct, do nothing.

One Approach

- **Train** the perceptron on valid data.
- For each data point (image of a digit):
  - Ask the perceptron what it thinks.
  - If correct, do nothing.
  - Otherwise, nudge w_0 … w_n in the right direction.

One Approach

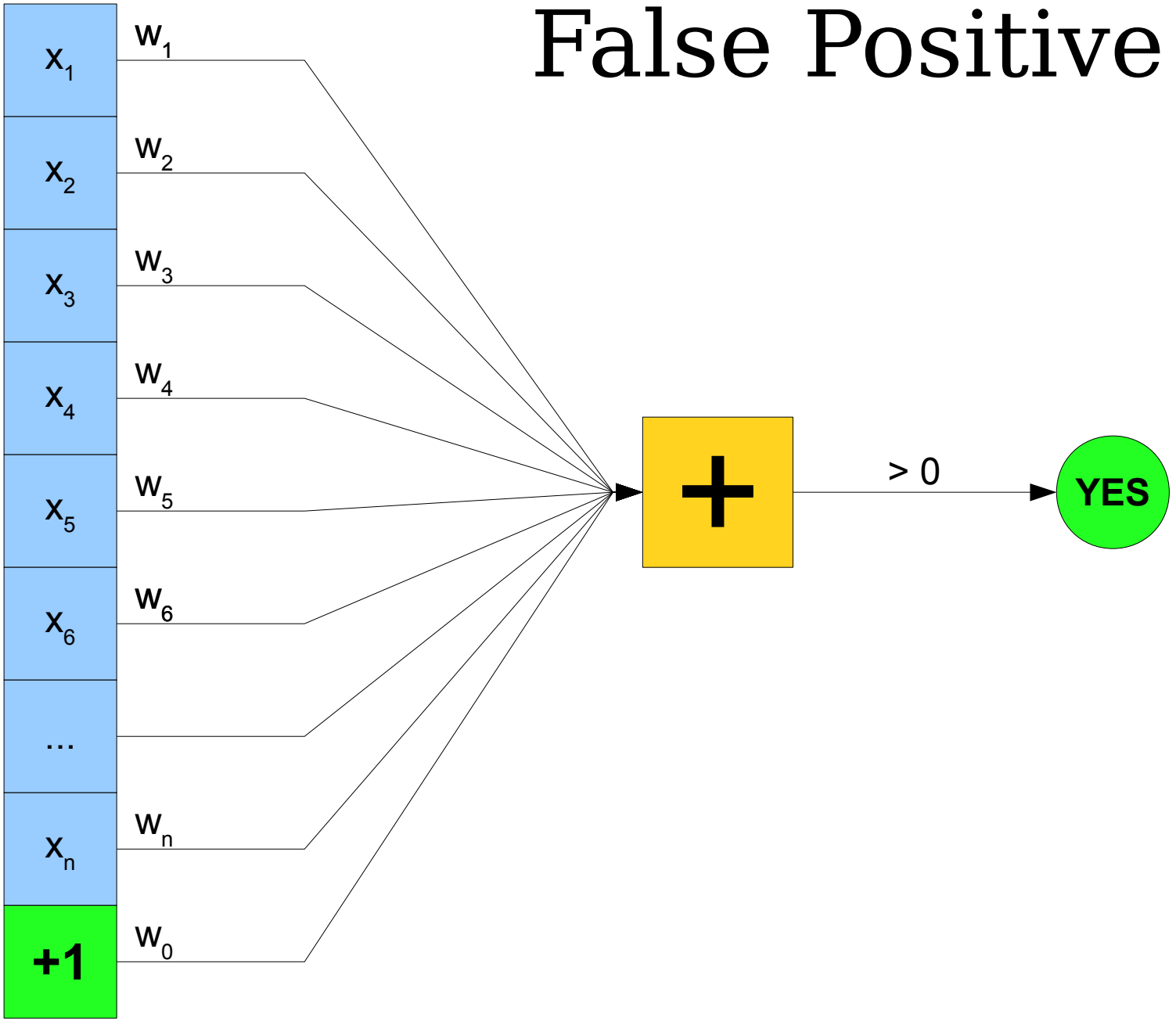
- **Train** the perceptron on valid data.
- For each data point (image of a digit):
  - Ask the perceptron what it thinks.
  - If correct, do nothing.
  - Otherwise, nudge w_0 … w_n in the right direction.
- Repeat until number of errors is “small enough.”

One Approach

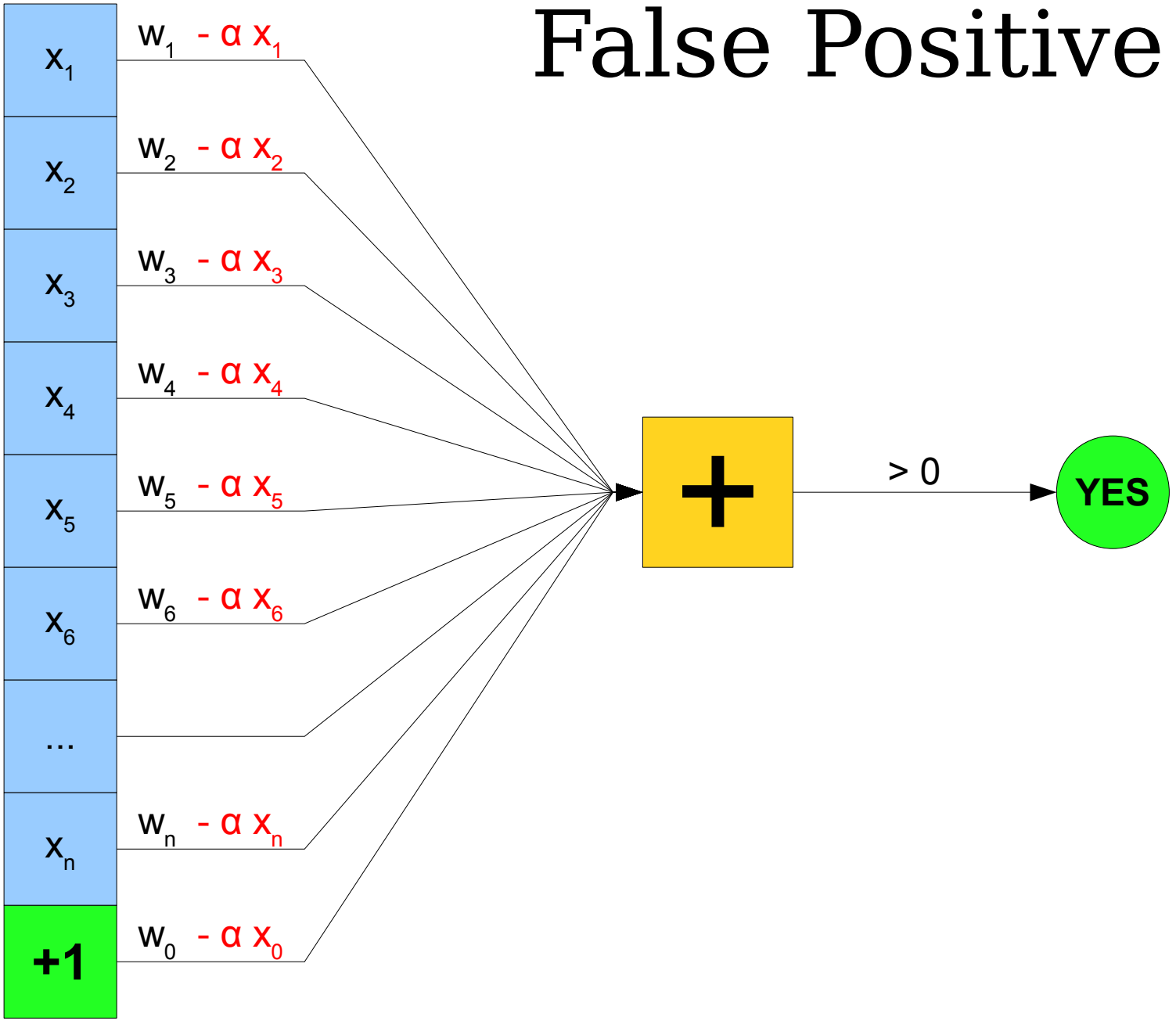
- **Train** the perceptron on valid data.
- For each data point (image of a digit):
  - Ask the perceptron what it thinks.
  - If correct, do nothing.
  - Otherwise, nudge w_0 … w_n in the right direction.
- Repeat until number of errors is “small enough.”
- Question: What kind of mistakes can we make?

False Positive

False Positive

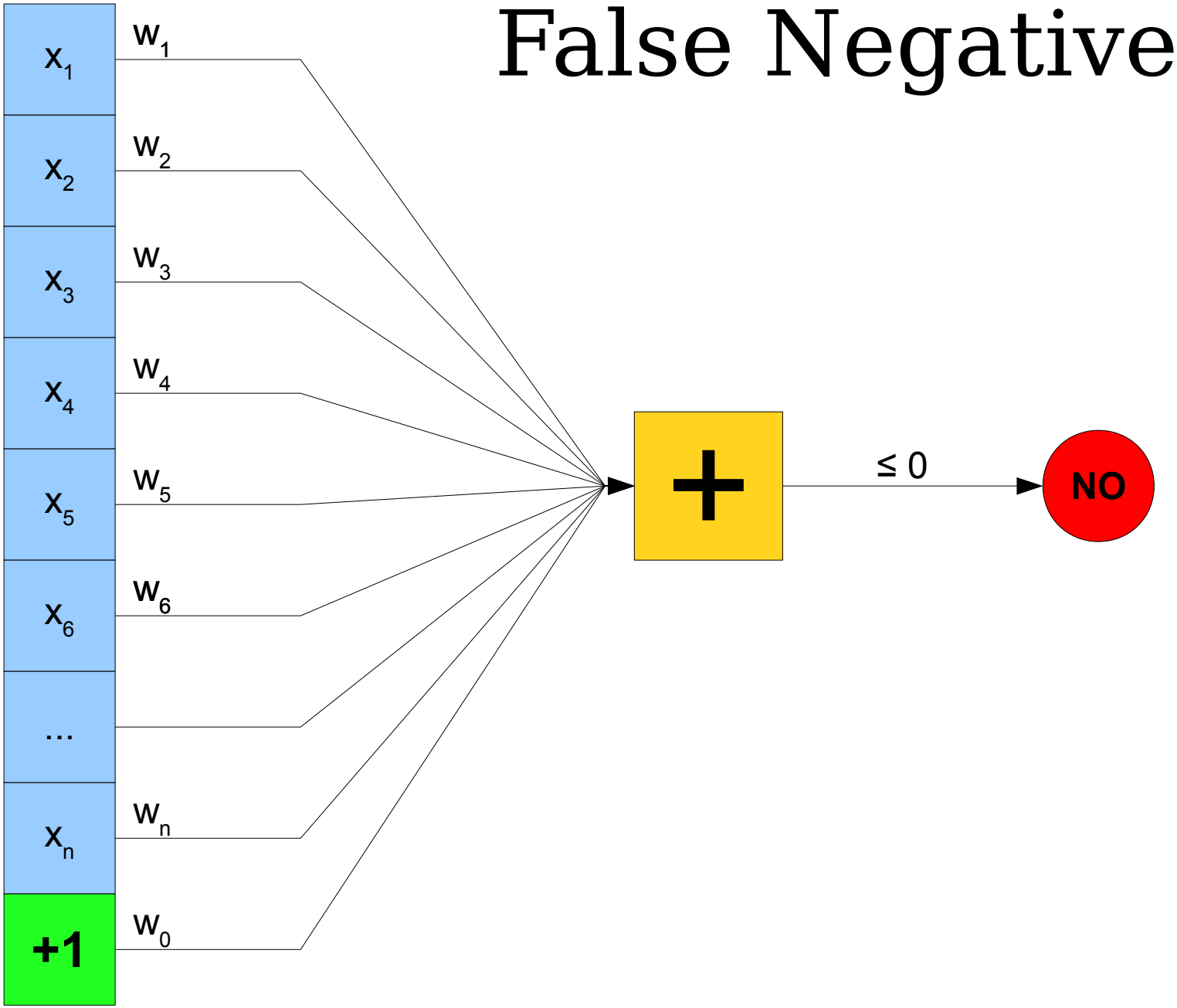


False Positive

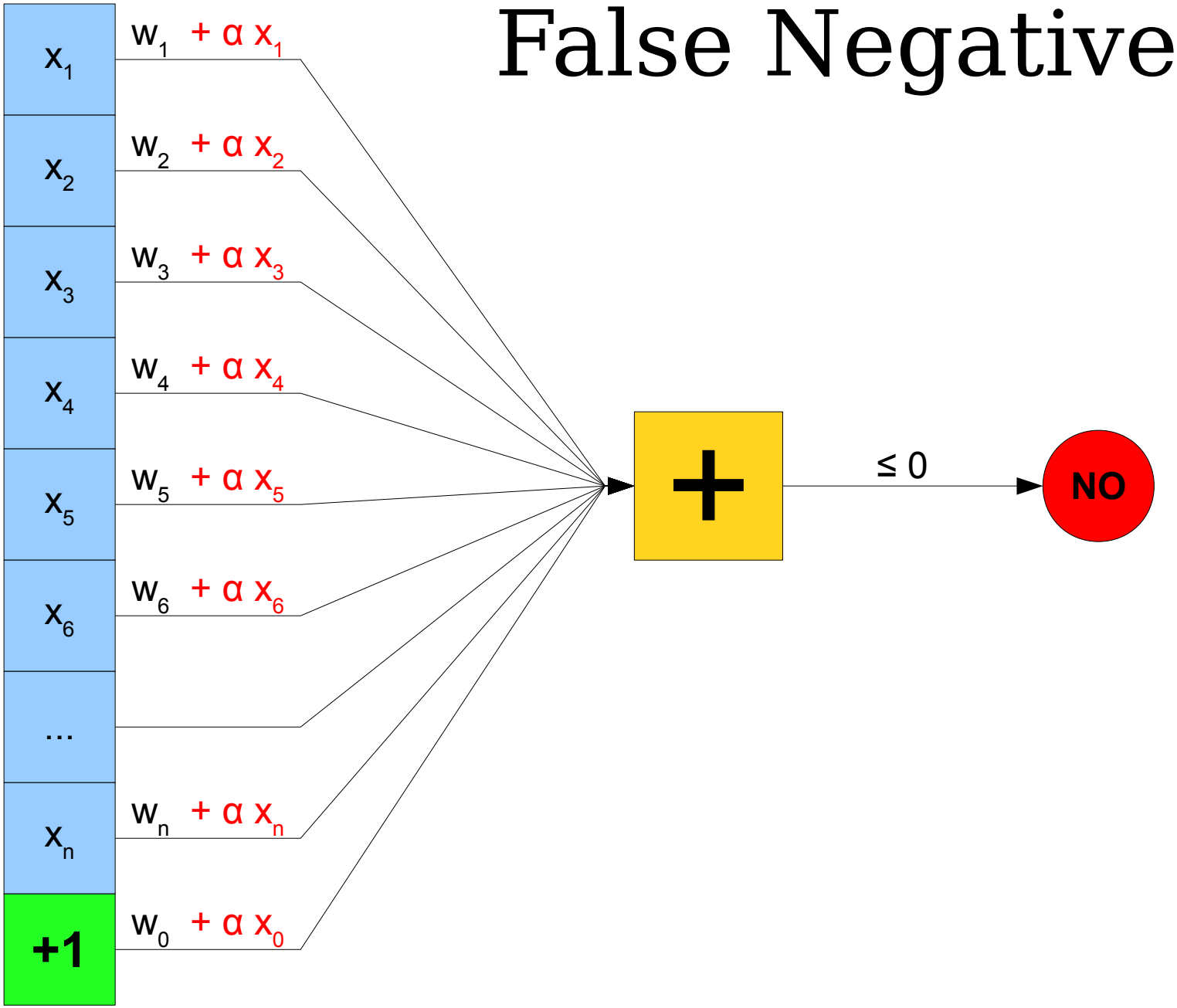


False Negative

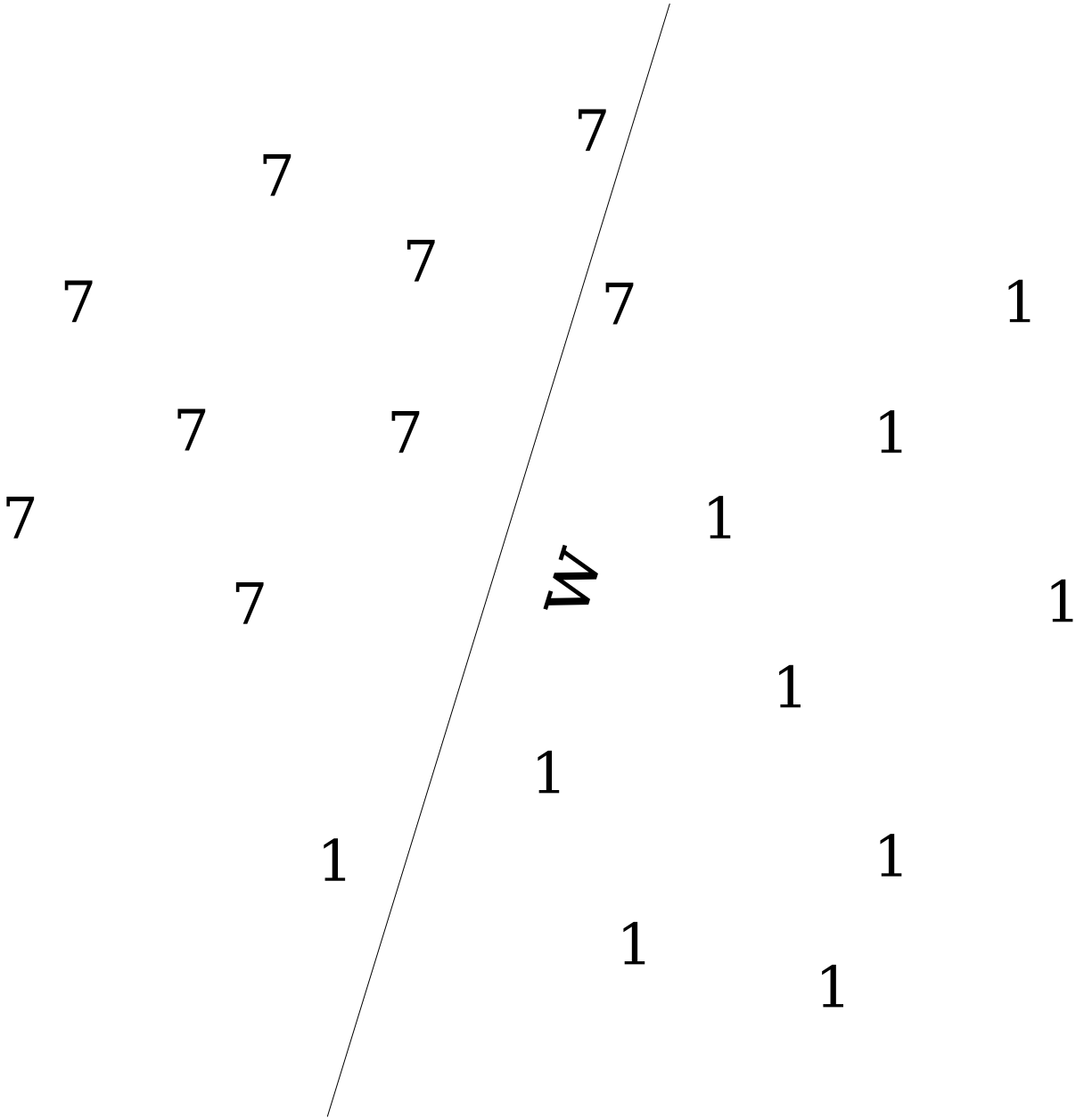
False Negative



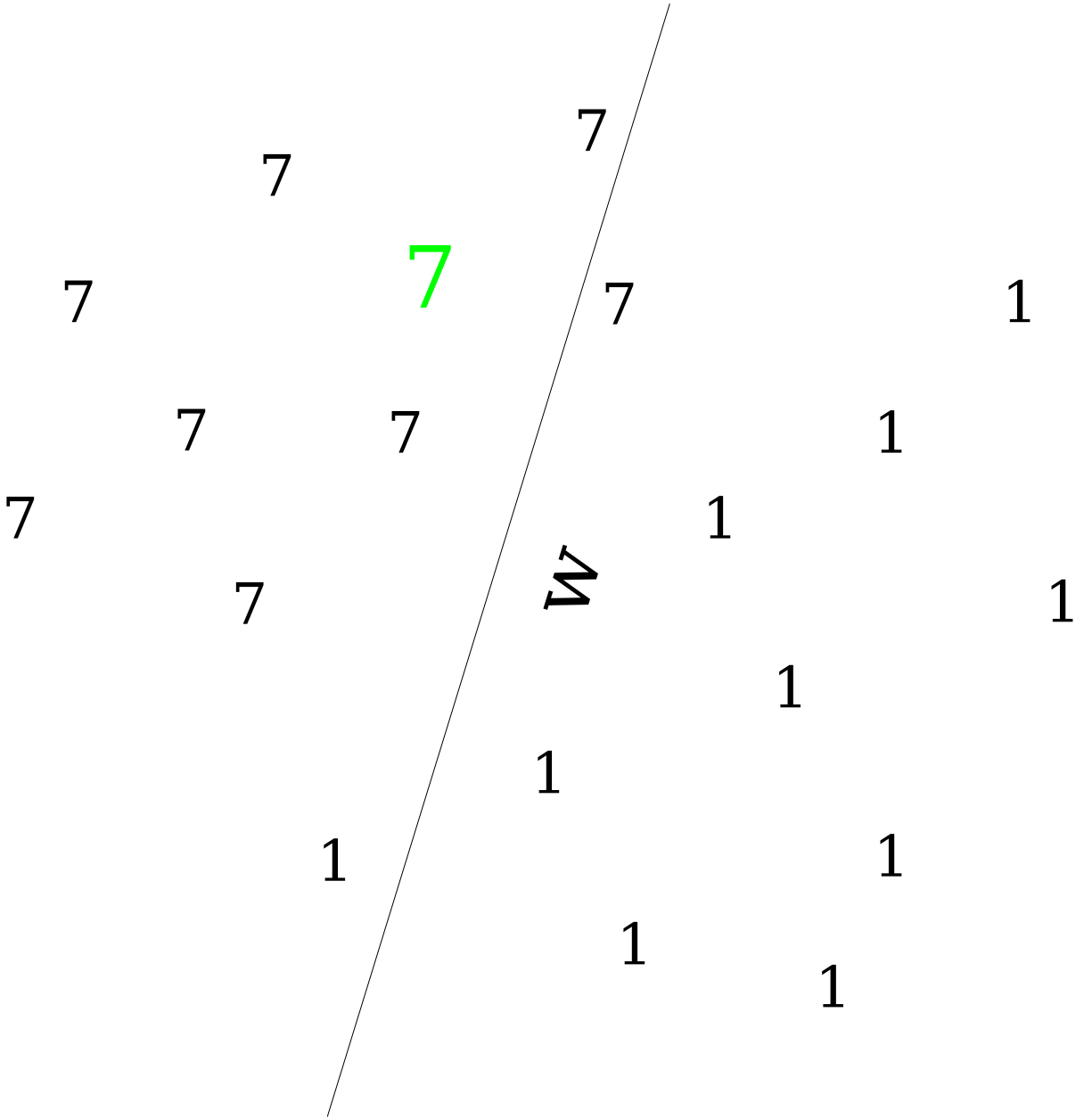
False Negative



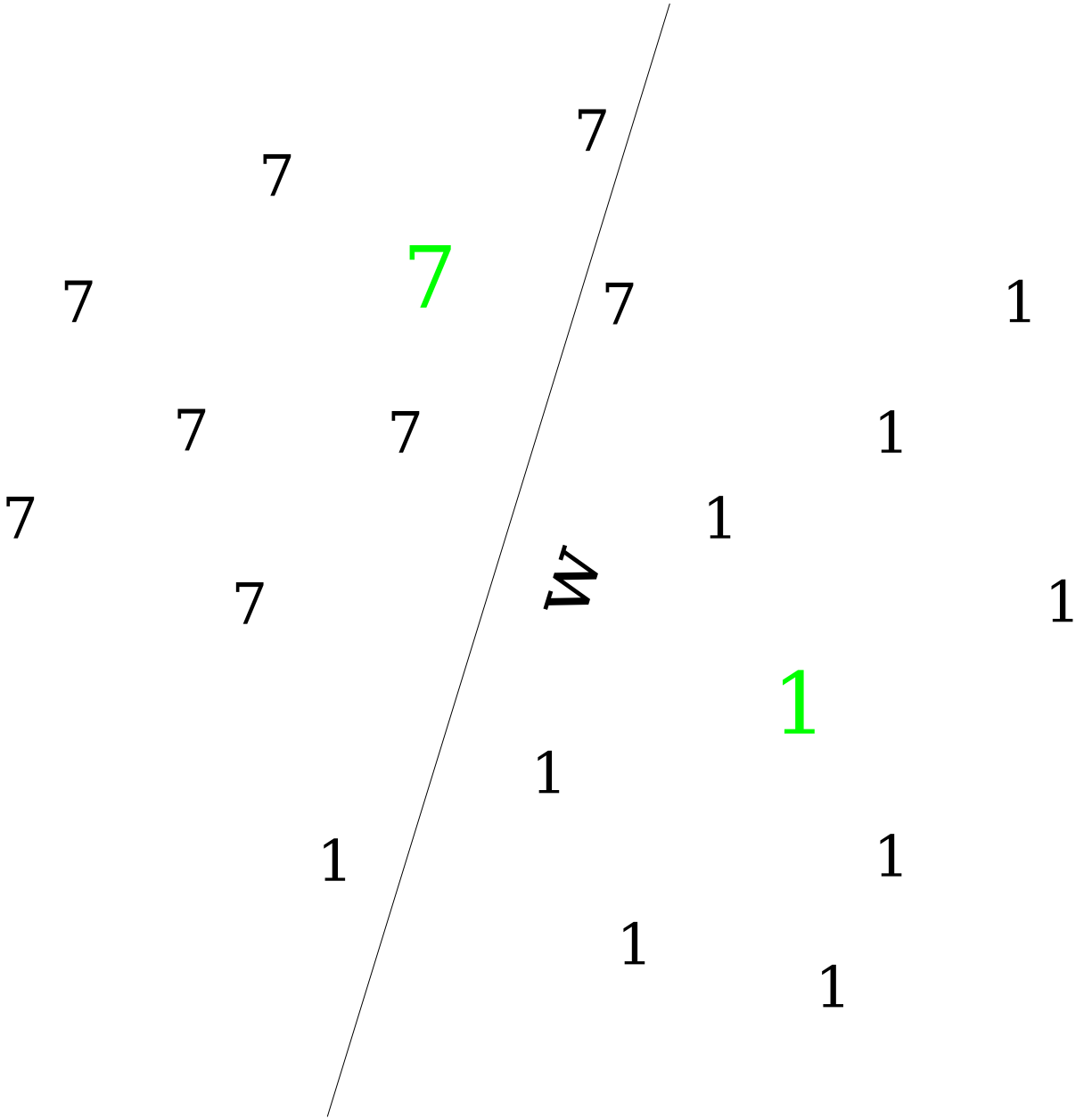
Visualizing w



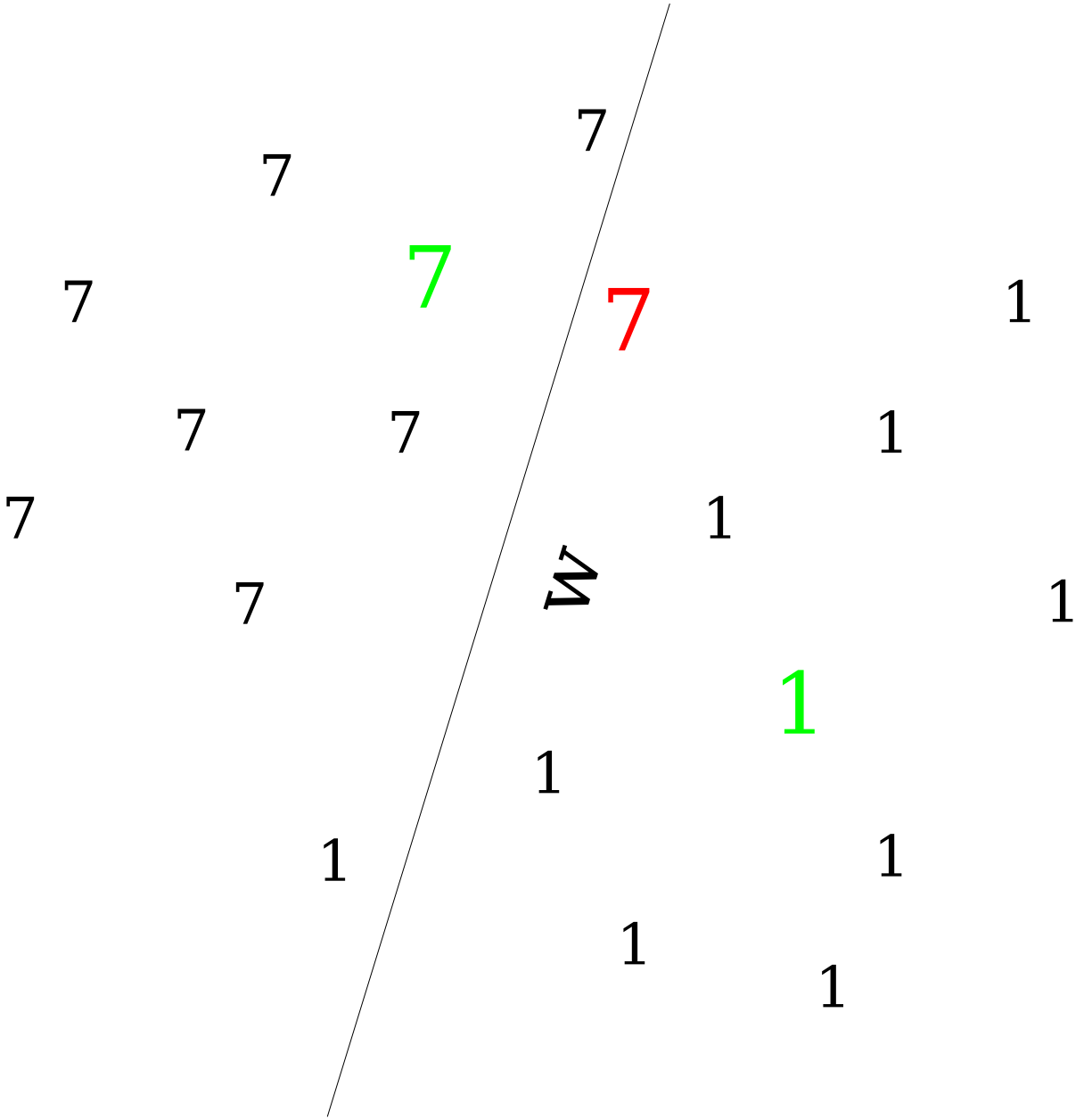
Visualizing w



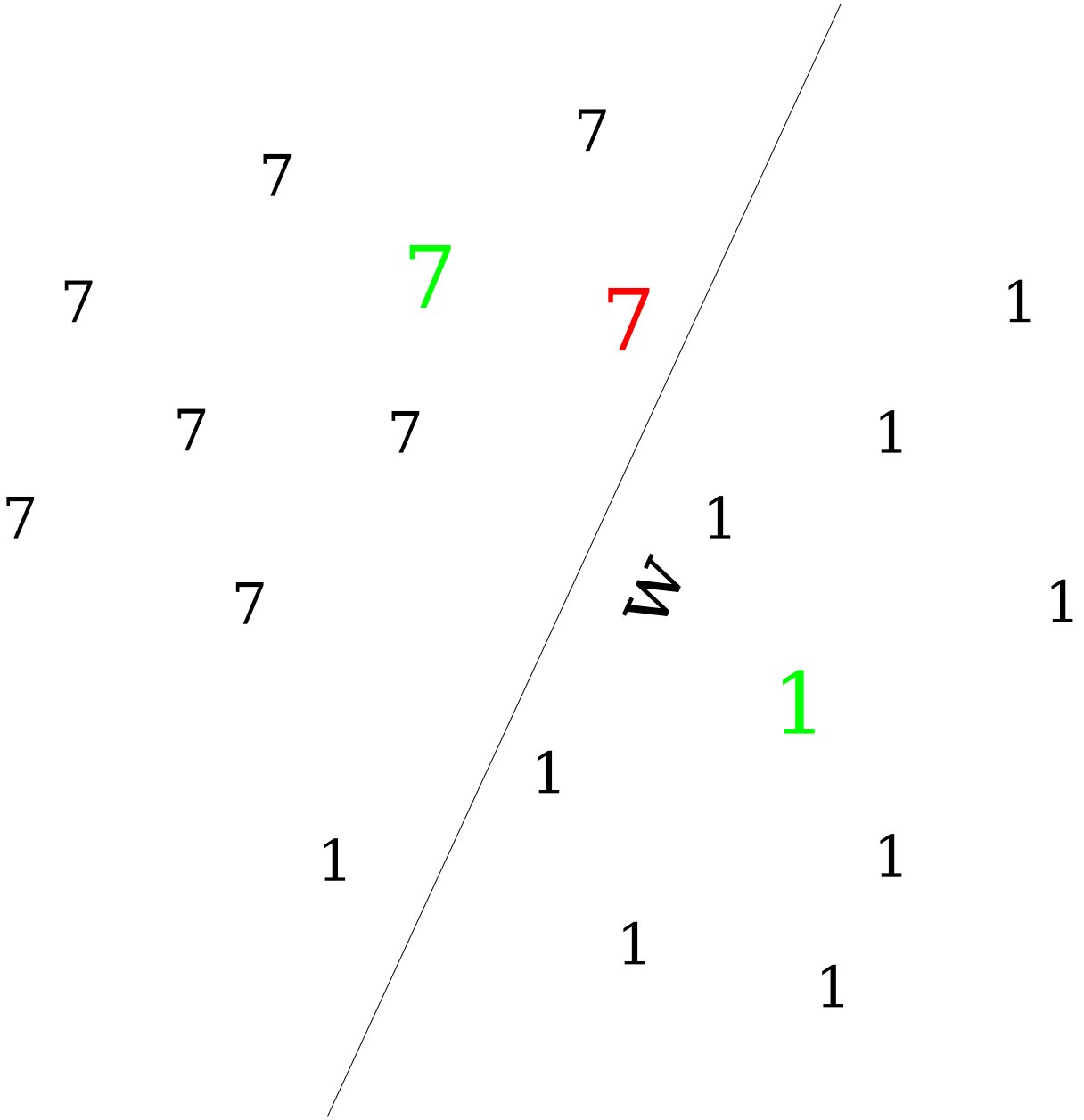
Visualizing w



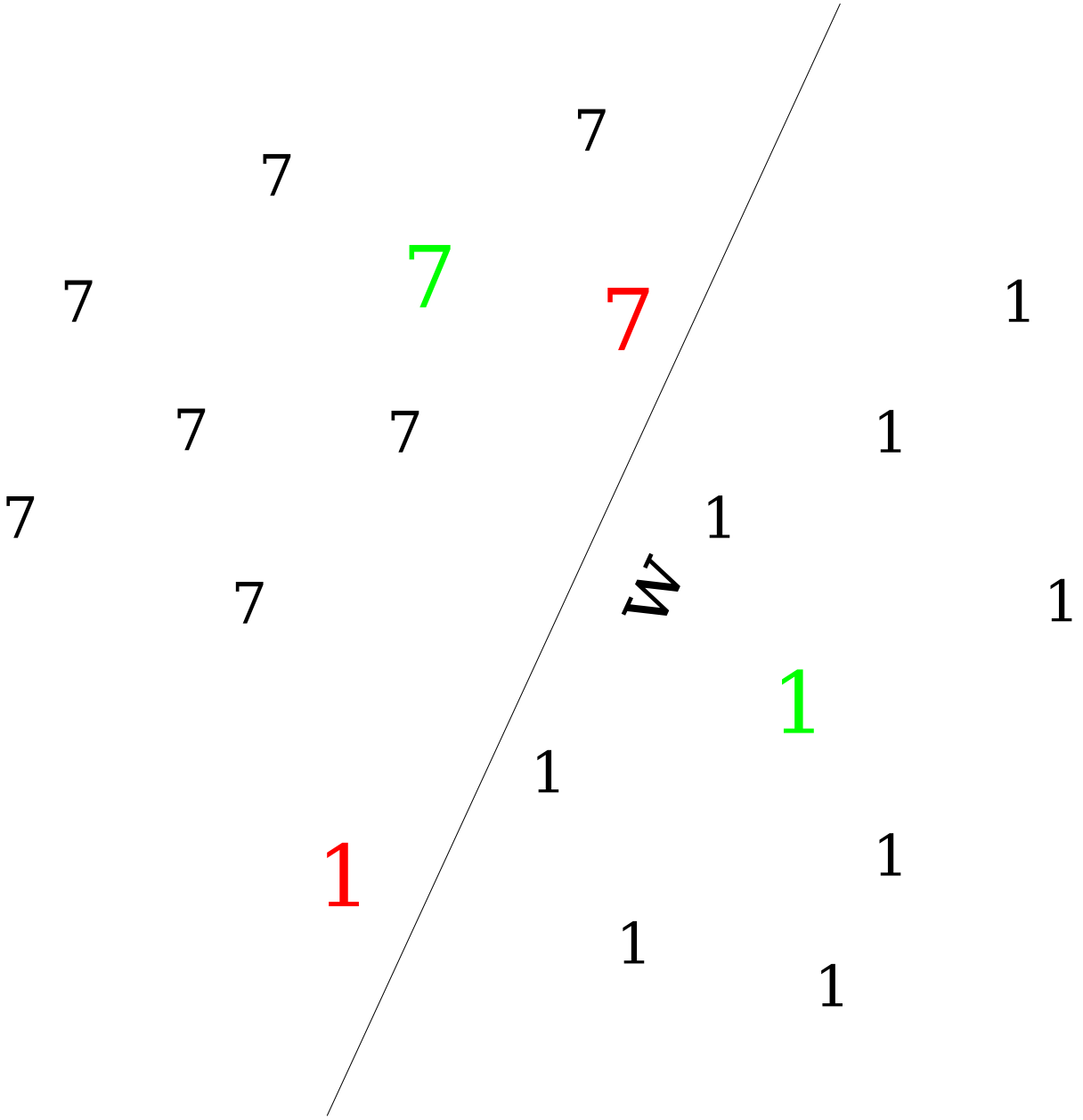
Visualizing w



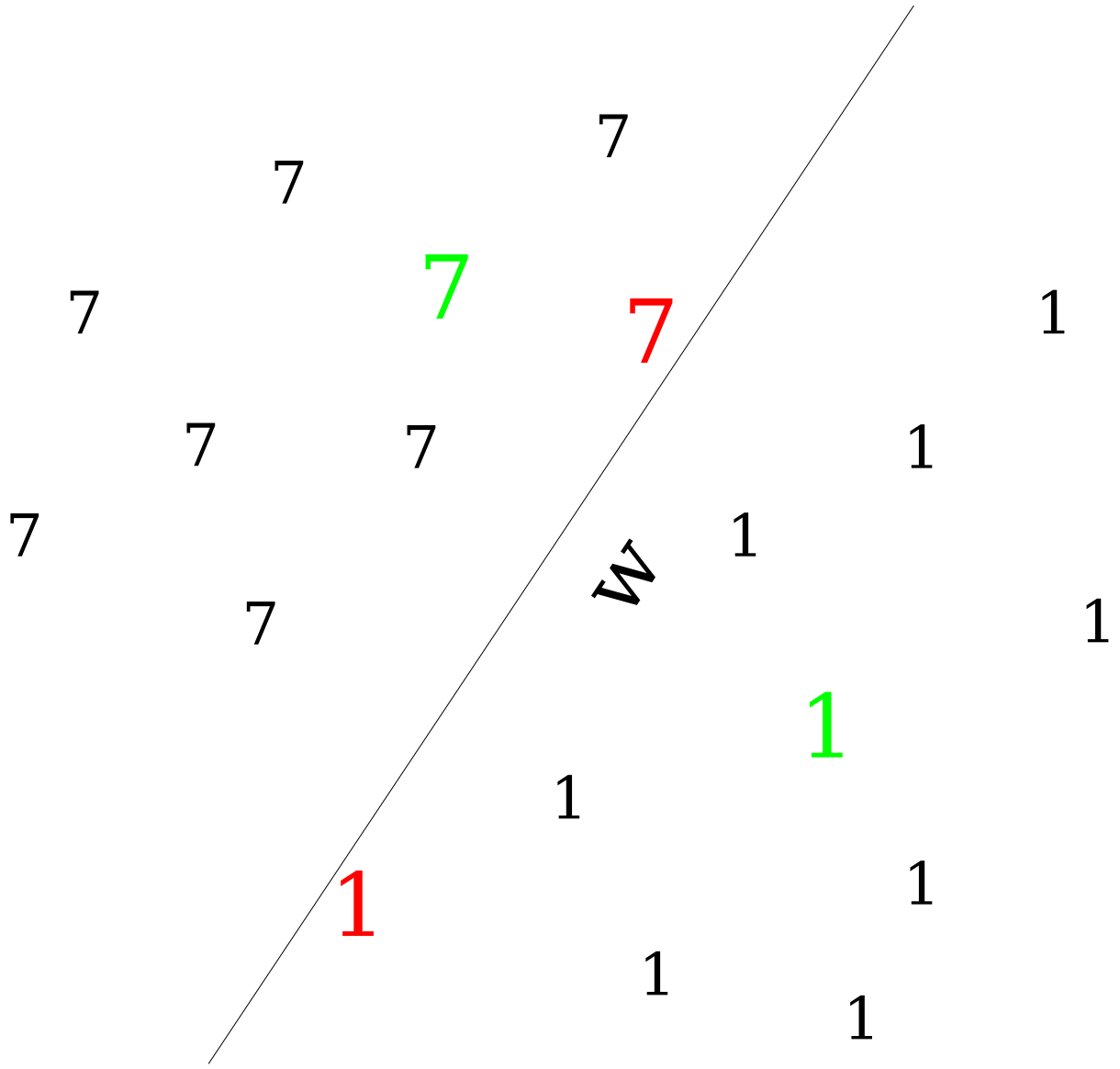
Visualizing w



Visualizing w



Visualizing w



A Nice Math Trick

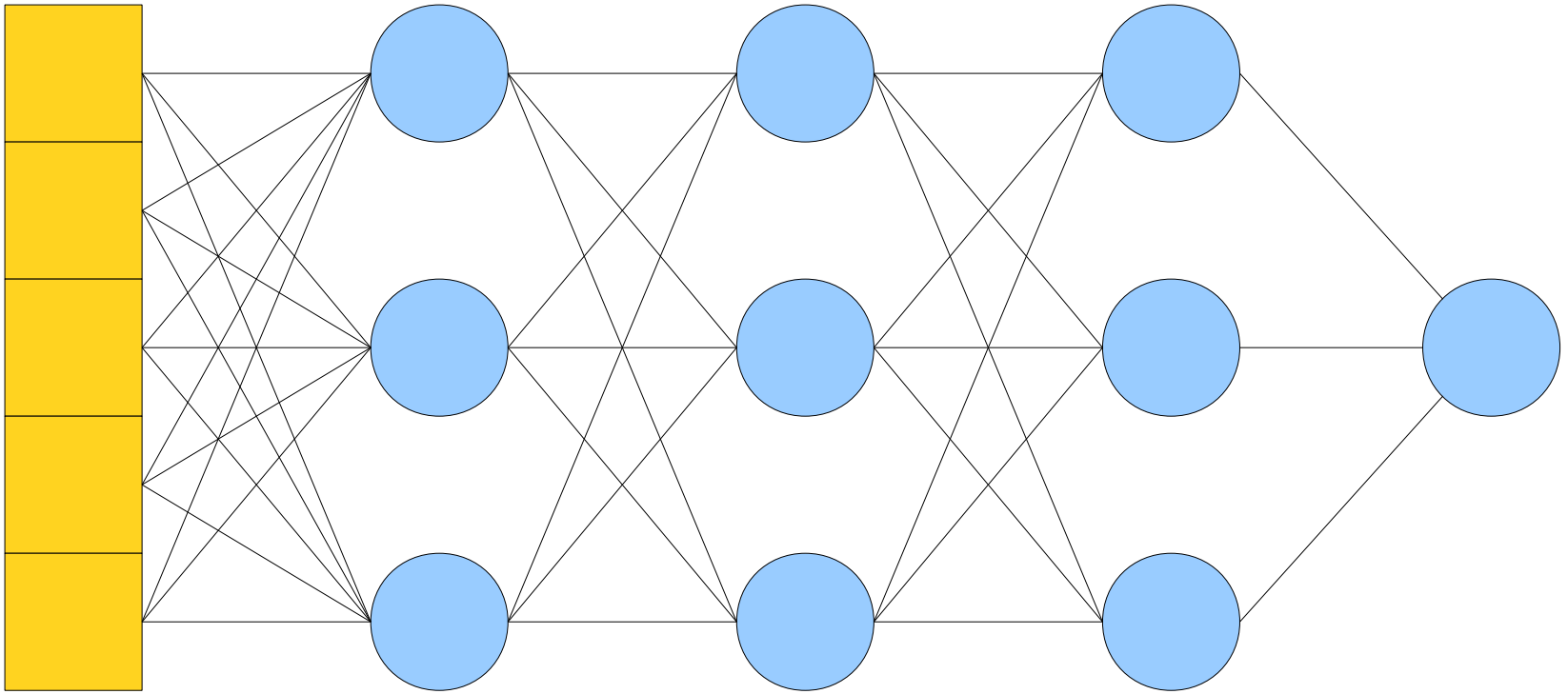
- For false positives, set $w_i = w_i - \alpha x_i$.
- For false negatives, set $w_i = w_i + \alpha x_i$.
- For correct answers, set $w_i = w_i$.
- Let “YES” be 1 and “NO” be 0.
- Consider the difference between **actual answer** and **perceptron guess**:
  - False positive: Actually NO, we say YES. Difference is -1.
  - False negative: Actually YES, we say NO. Difference is +1.
  - Correct answer: Both YES or both NO. Difference is 0.
- General update rule: $\mathbf{w_i = w_i + \alpha (real - guess) x_i}$.

Perceptron Learning Algorithm

- Start with a random guess of each w_i.
- Repeat until perceptron is sufficiently accurate:
  - Choose a training example (x_0, x_1, ..., x_n).
  - Let **real** be the real answer, **guess** be the perceptron's guess.
  - For each i, set $\mathbf{w_i := w_i + \alpha (real - guess) x_i}$

Visualization of `w`!

Combining Perceptrons



This is called a neural network.

Perceptrons

- How “good” are perceptrons?
  - Not very…they have lots of issues.
  - Better algorithms exist for classification
- Chose to cover them because they don't require much math.

Machine Learning

- Interesting in machine learning? Take CS109 and CS229.

Exam Statistics

- Median: 35
- First Quartile: 24
- Third Quartile: 42