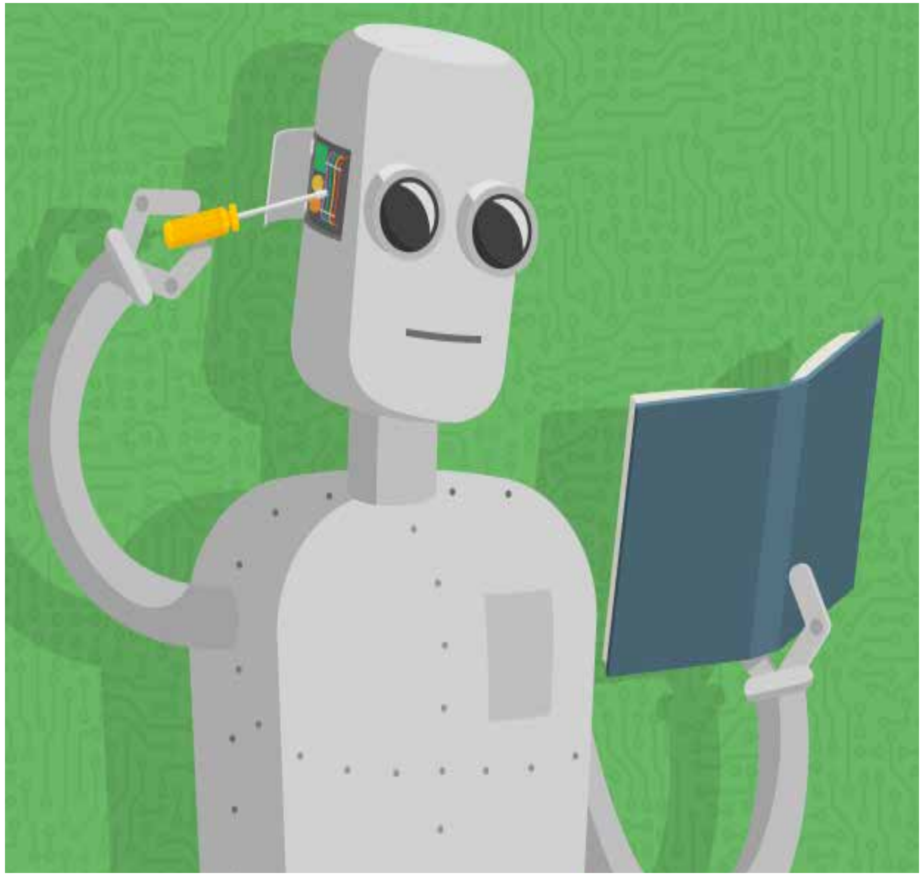
CS 466/566
Introduction to Deep Learning

Lecture 1

Introduction to Machine Learning - Part I

Mostly compiled from the lectures of Andrew Ng

What is Machine Learning?



1959: “[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.”

– **Arthur Samuel** (1901-1990)

- First AI implementation. Checkers game
- 1st implementation of Alpha-beta Pruning (Chess)
- Member of TeX project

1997: “A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.”

– ***Tom Mitchell*** (1951-…)

- Carnegie Mellon University Professor
- Chair of the Machine Learning Department at CMU
- The author of the textbook *Machine Learning.*

Example of ML algorithm’s expected outcome

- Let’s say you want to predict traffic patterns at busy intersections
  (this is task T)
- You can run it through a, so-called, ML algorithm with data about past traffic patterns
  (This is the experience E)
- If it has successfully “learned”, it will then do better at predicting future traffic patterns
  (This is performance measure P)

Example ML applications of today

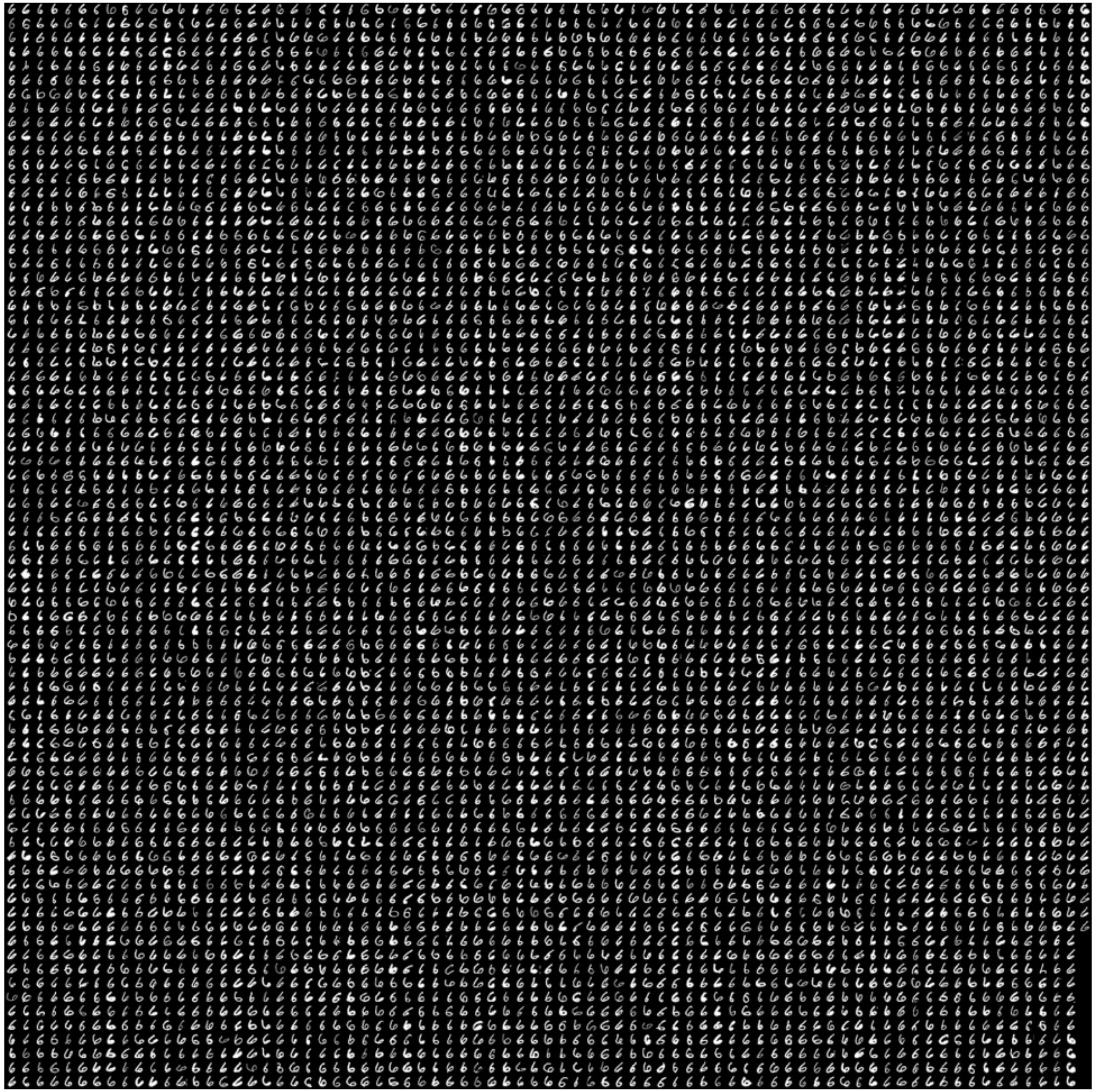
- "Is this cancer?" (2007)
  https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2675494/
- "What is the price of this house" (2008)
  https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1316046
- "Which of these people are good friends with each other?" (2012)
  https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2187186
- "Will this rocket engine explode at take-off?"(NASA-1990)
  https://archive.org/details/nasa_techdoc_19960011791
- "Which digit does this handwritten image represent?" (1998)
  http://yann.lecun.com/exdb/mnist/

Take M-NIST for instance

60000 digit images as training set data.
10000 digit images as test set data.

Each image is 28x28 resolution.
Images are gray scale images (8-bit, single channel)

All of 6 digits in M-NIST Training Set



A crucial distinction in ML types

- **Supervised Machine Learning**:
  - The program is “trained” on a pre-defined set of “training examples”, which then facilitate its ability to reach an accurate conclusion when given new data.

- **Unsupervised machine learning:**
  - The program is given a bunch of data and must find patterns and relationships therein.

Supervised Machine Learning

- In the majority of supervised learning applications, the ultimate goal is to develop a finely tuned predictor function **h(x)** (sometimes called the “hypothesis”).
- “**Learning**” consists of using sophisticated mathematical algorithms to optimize this function so that:
  - given input data **x** about a certain domain (say, square meters of a house),

    it will accurately predict some interesting value **h(x)** (say, market price for said house).

x almost always represents multiple data points:

- x_1 = number of bedrooms
- x_2 = number of bathrooms
- x_3 = number of floors
- x_4 = year built
- x_5 = zip-code
- …

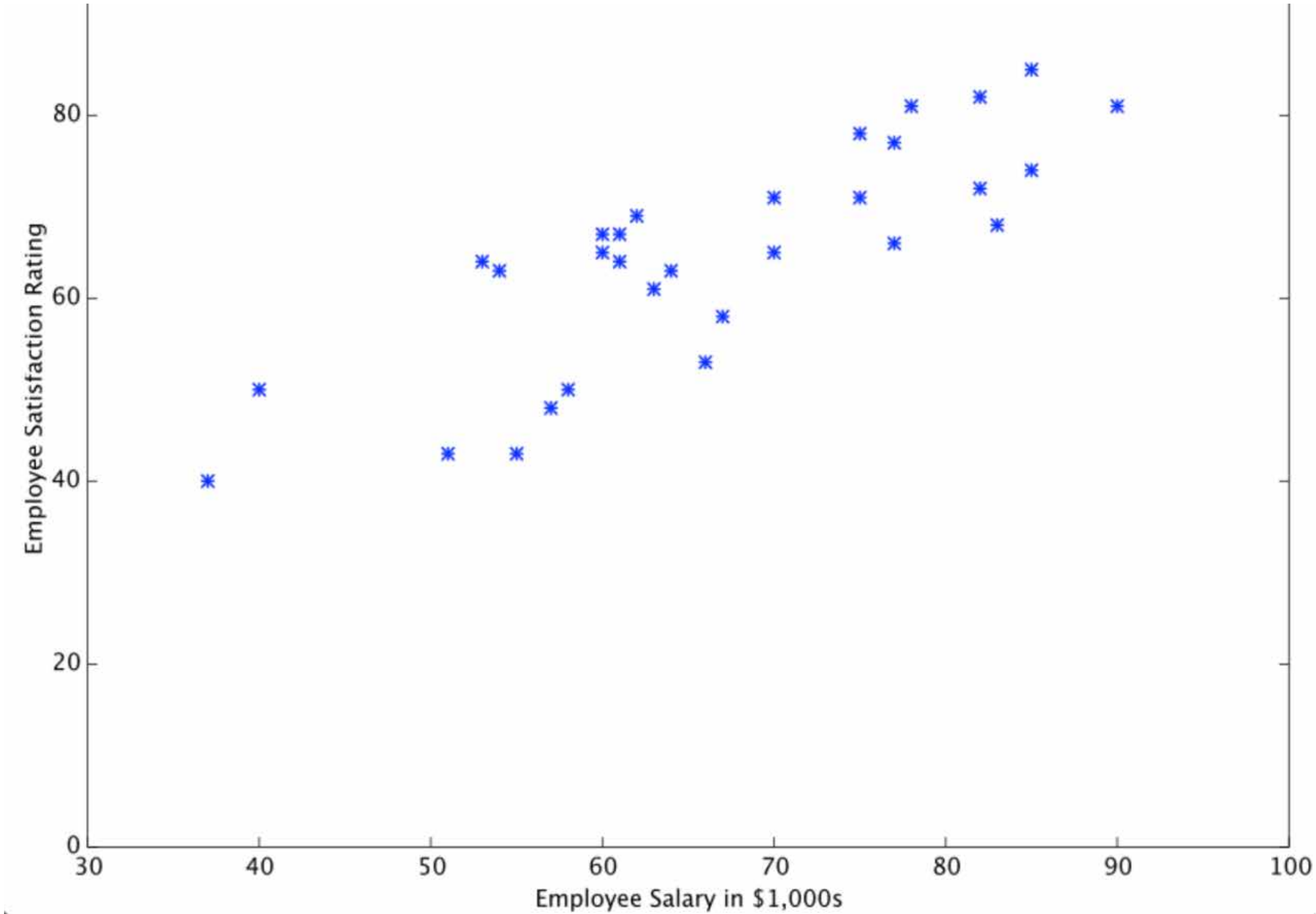
Supervised ML Example

- Let’s say our simple predictor has the form (model) below:

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

- Our goal is to find values of ”a” and “b” to make our predictor as good as it gets.
- Optimizing the predictor y(x) is done by using “**training examples**”.
    - For each training example:
        - We have the correct value of y_{known}, and its prediction y(x_train).
        - Find difference between y_{known} and y(x_train).
        - Measure the wrongness (error) amount (by using enough training examples)
        - Tweak y(x) predictor by changing ”a” and “b” parameters of our model.
        - Hope that it will be a better estimator!

Company Employees (Satisfaction/Salary)



Notice that data is a bit noisy.

We can see a pattern.

But it does not satisfy all the points.

Points being employees here.

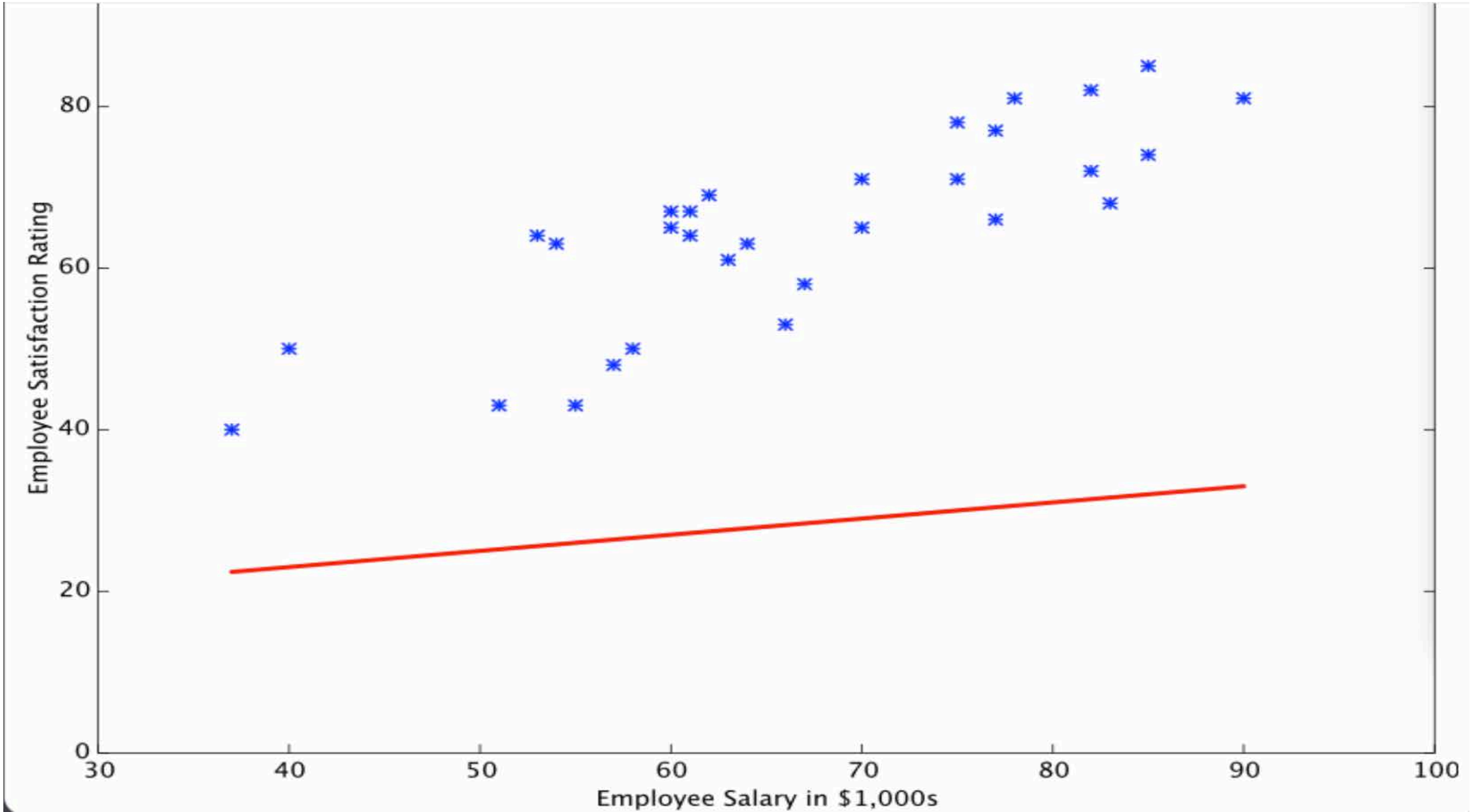
Data will always be noisy.

ML algorithm must be robust to it.

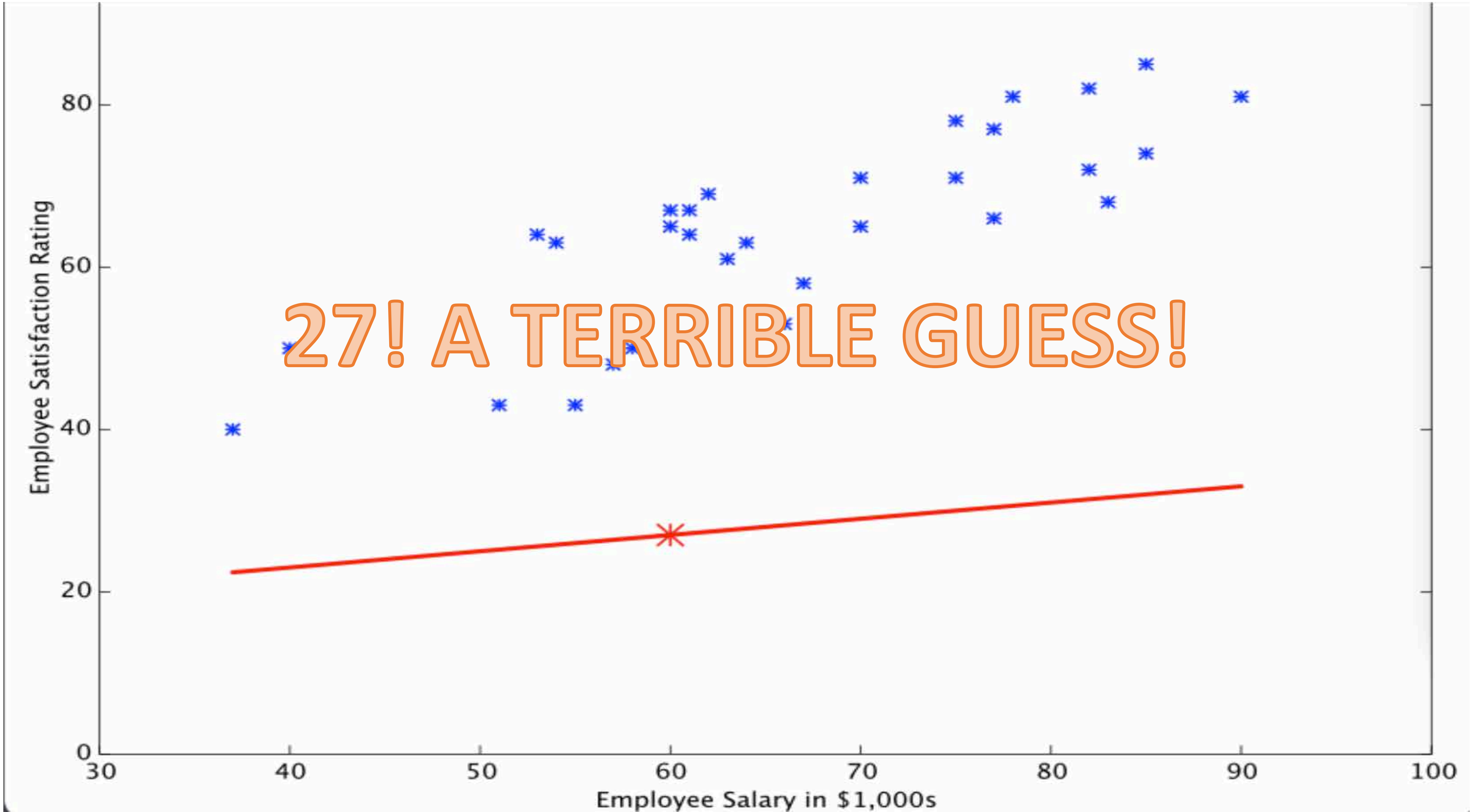
How can we **perfectly** predict the satisfaction of a new employee?
We can't!

"all models are wrong, but some are useful"
- **George Edward Pelham Box** (1919-…)

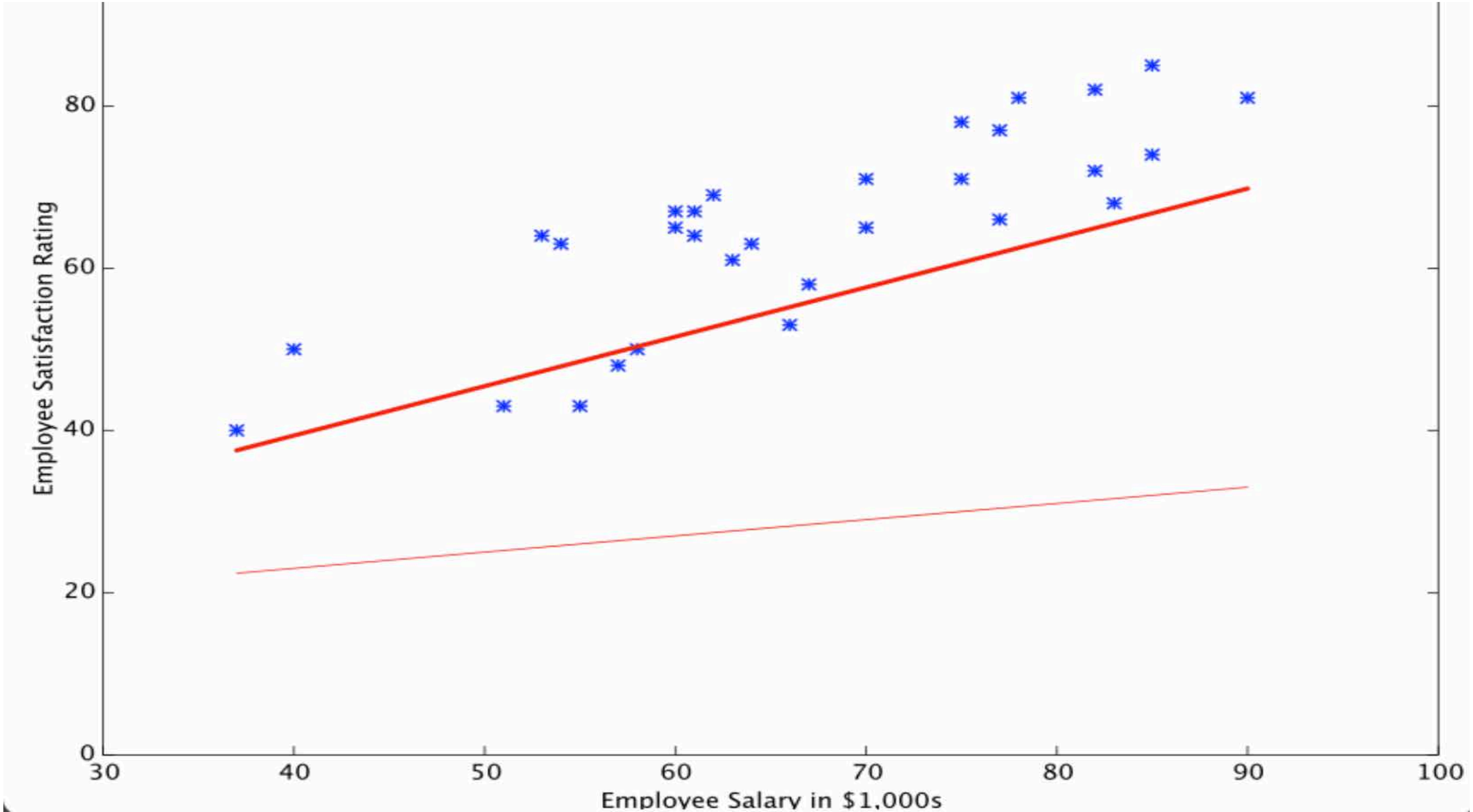
Initialize our predictor $y(x) = 0.20x + 12.00$



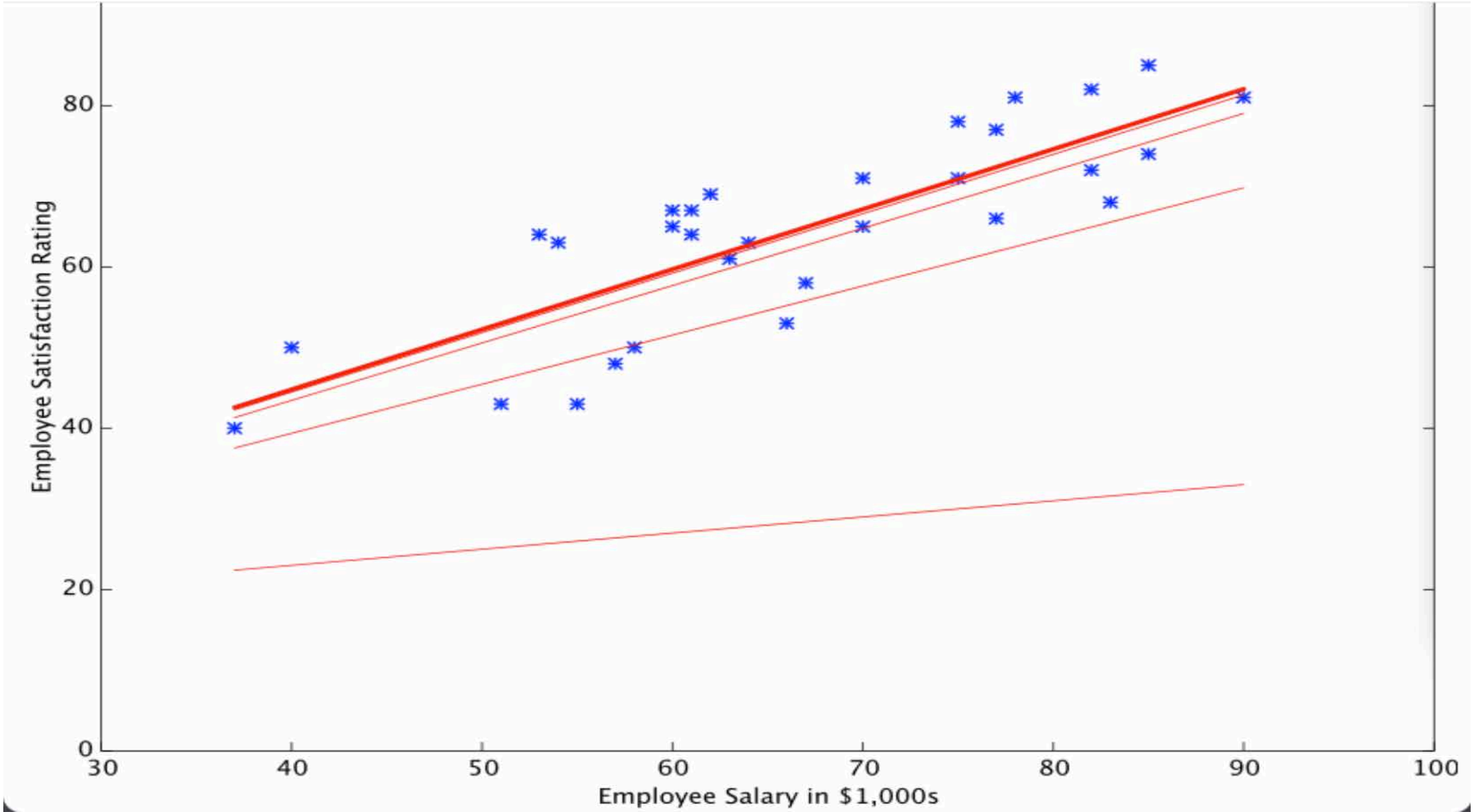
Satisfaction of an employee making \$60k → 27



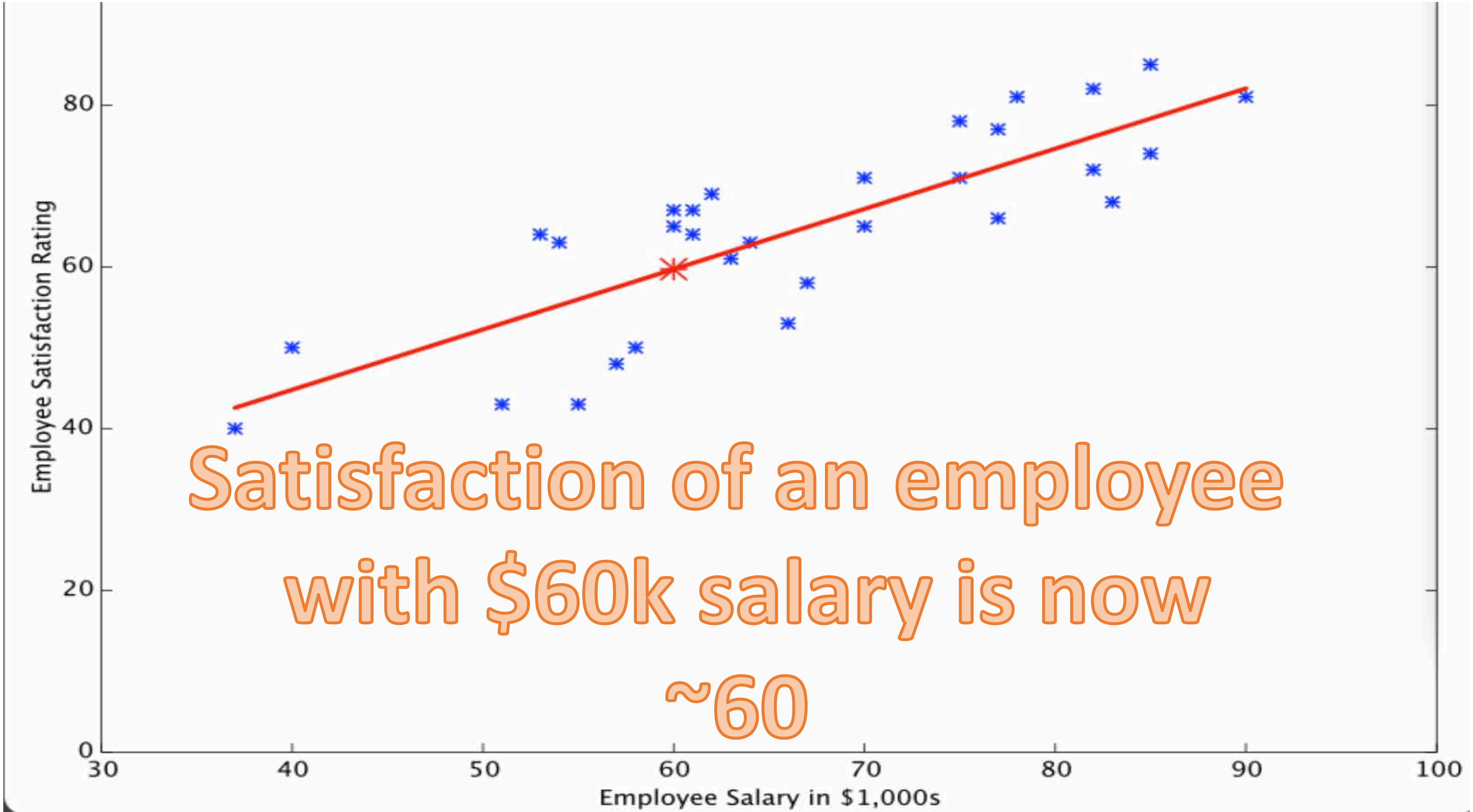
Imagine, we have a direction for goodness!



Apply our magic algorithm.



$$y(x) = 0.75x + 15.54$$



Satisfaction of an employee with $60k salary is now ~60

Univariate linear regression

- Can you solve for Univariate Linear Regression and find the direct answer?
  - Yes, of course.
- Then, why do we use a magic algorithm for predicting something that we can directly calculate using “Least Squares”?
  - $y(x_1, x_2, x_3, x_4) = a + bx_1^2 + cx_2^2 + \mathrm{d}x_3^3 x_4^2 + \ \ldots$
- Deriving a normal equation for this function is a significant challenge. Many modern machine learning problems take thousands or even millions of dimensions of data to build predictions using hundreds of coefficients.

Dimensionality of today’s problems

- Many modern ML problems take thousands or even millions of dimensions of data to build predictions using hundreds of coefficients.
- Fortunately, the iterative approach taken by ML systems is much more resilient in the face of such complexity.
- Instead of using brute force, a machine learning system “feels its way” to the answer.
- For big problems, this works much better.

Gradient Descent (Minimize "Wrongness")

- How do we measure that our univariate linear model gets better?
- Because we have a measurement of wrongness:
  - It's the so called **cost function** (or loss function, or error function), $J(\theta)$
  - θ represents all of our parameters (i.e. *a* and *b* from our 1st linear regression)
- The choice of the cost function is another important piece of an ML program. Contextually, being "wrong" can mean very different things.
- Well-established cost function for our case is **linear least squares**.

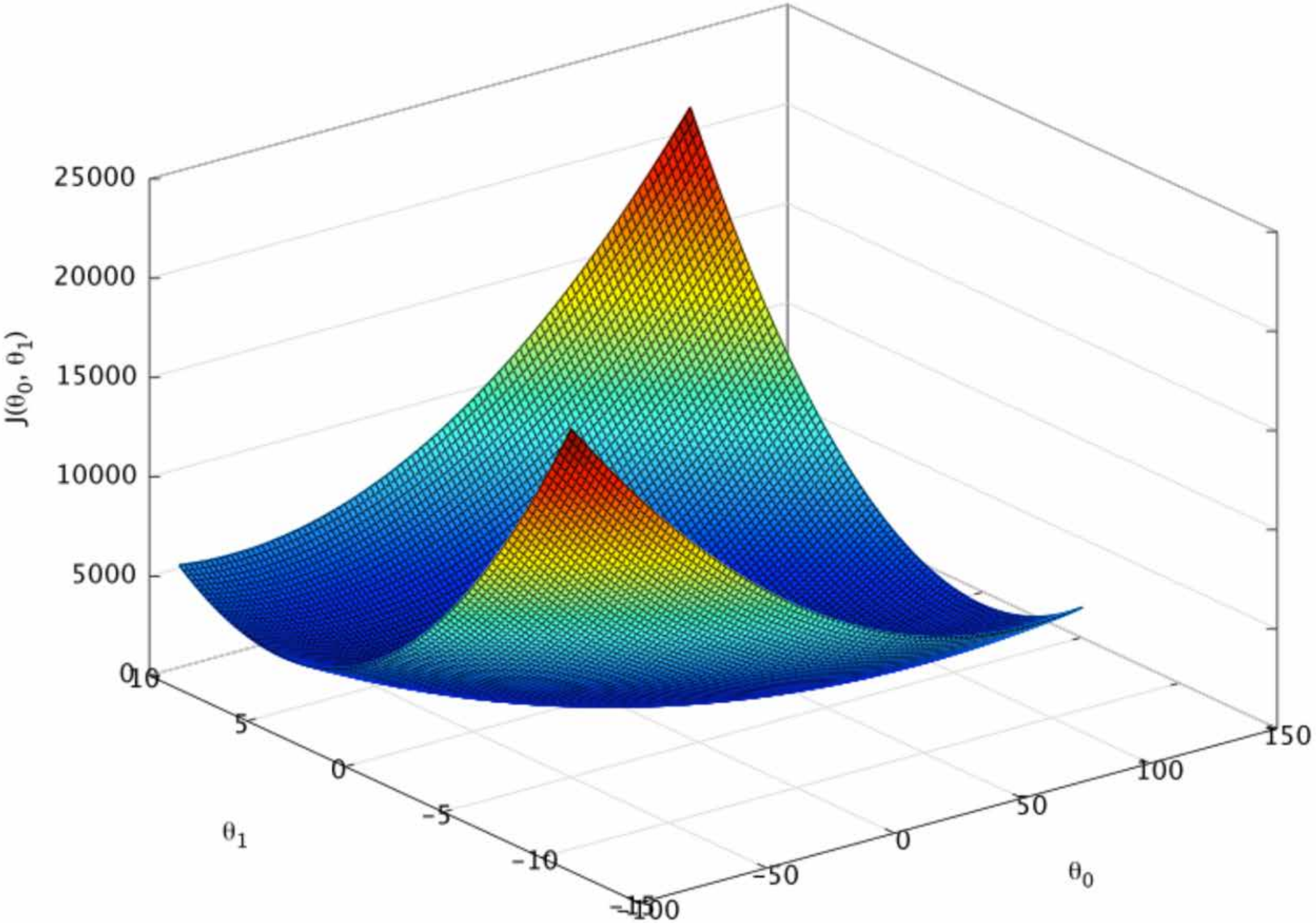
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(y\left(x_{t,i}\right) - y\right)^2$$

Linear Least Squares

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(y(x_{t,i}) - y\right)^2$$

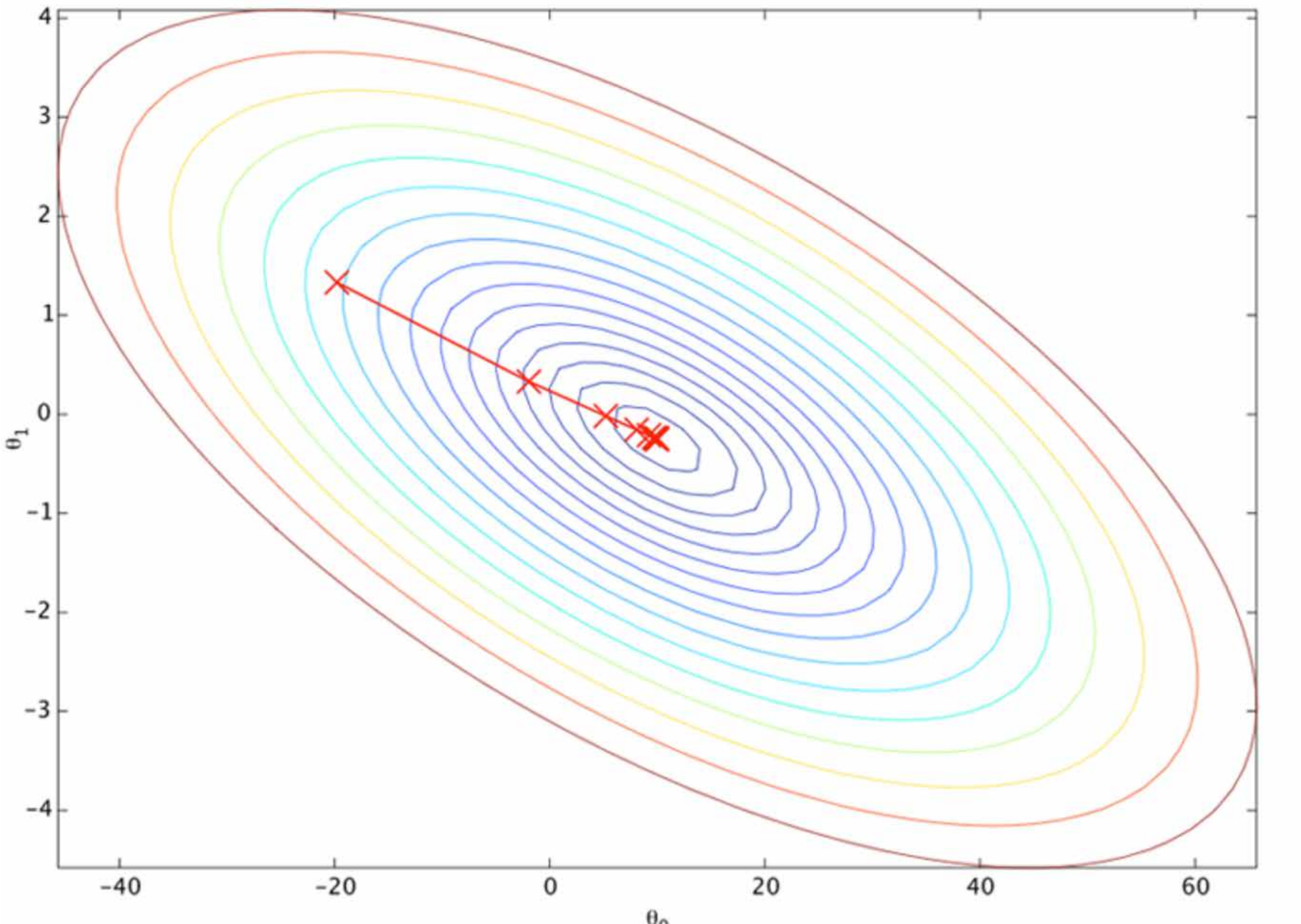
- the penalty for a bad guess goes up quadratically with the difference between the guess and the correct answer
- it acts as a very “strict” measurement of wrongness.
- The cost function computes an average penalty over all of the training examples.

Plot of Cost Function

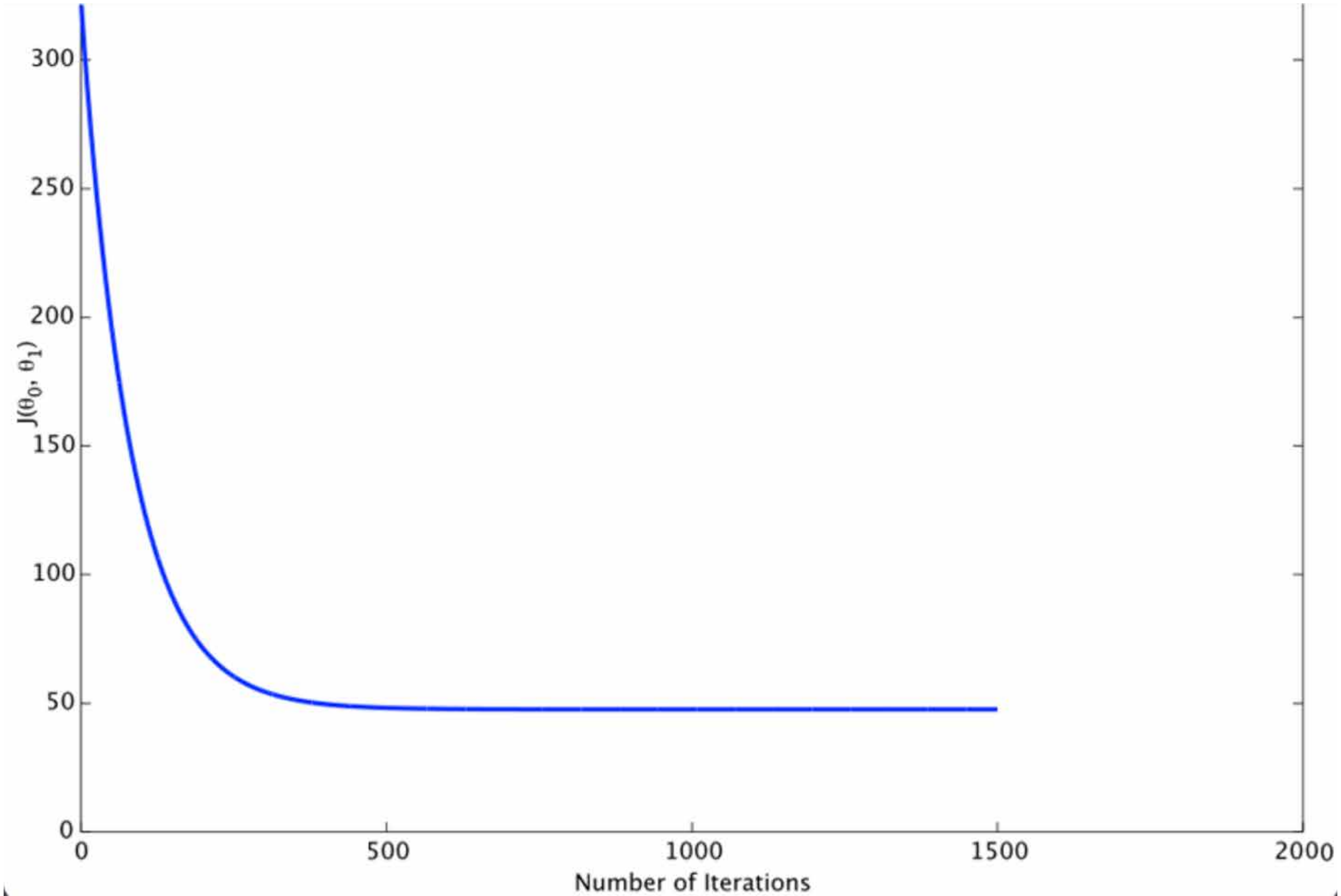


- We see that our goal is to find θ of or our predictor y(x) such that our cost function is as small as possible.
- Take the gradients in different parameter directions.
- Try to go down-hill.
- **This is called gradient descent.**

Gradient descent iterations



Cost function vs iterations



Classification Problems

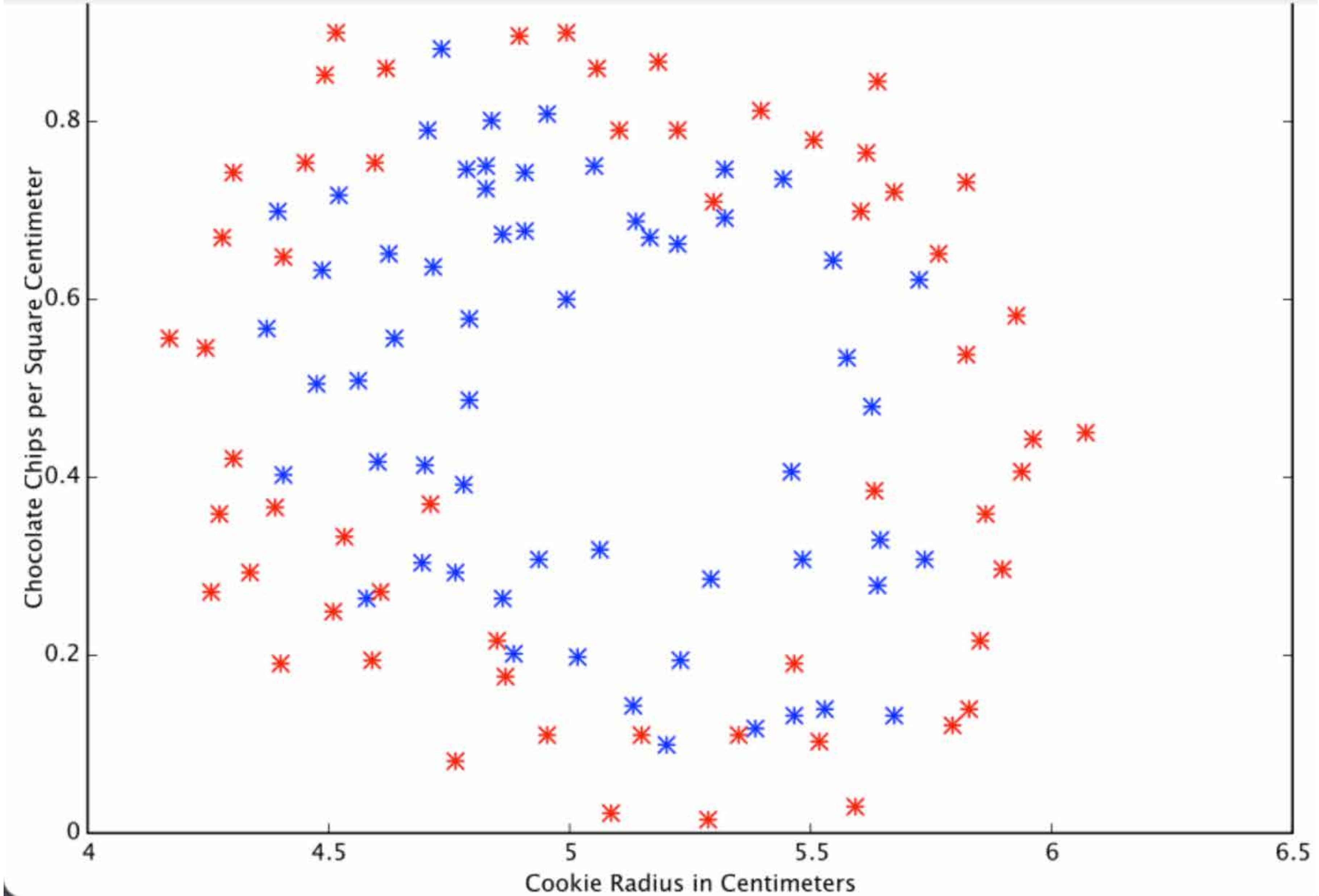
- Under supervised ML, two major subcategories are:
  - Regression machine learning systems:
    Systems where the value being predicted falls somewhere on a continuous spectrum. These systems help us with questions of “How much?” or “How many?”

  - Classification machine learning systems:
    Systems where we seek a yes-or-no prediction, such as “Is this tumor cancerous?”, “Does this cookie meet our quality standards?”, “Does this picture contain a dog?”, “Which brand of automobile is in this picture?”, “Which digit does this hand-writing image represent?”, and so on.

How do we classify?

- It turns out that most of the theory is the same for classification.
- We need some modifications in our **predictor** and in our **cost function**.
- Let’s see it with an example: A cookie quality testing study!
  There are some “good cookies” labeled as y=1 (shown by blue),
  there are some “bad cookies” labeled as y=0 (shown by red)

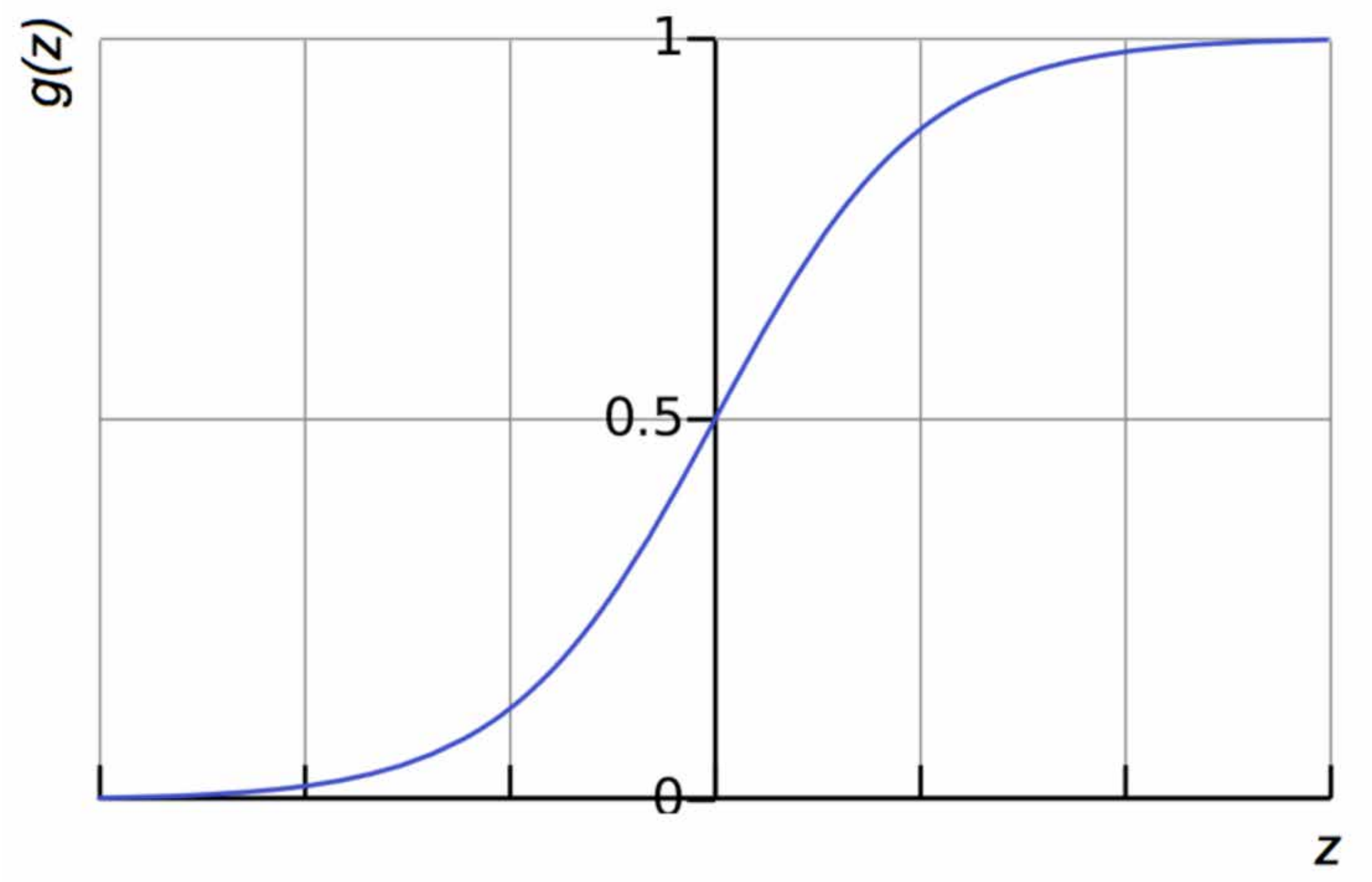
Cookie Quality Test (Good and Bad cookies)



How do we classify?

- In classification, a regression predictor is not very useful. What we usually want is a predictor that makes a guess somewhere between 0 and 1.
- A prediction of:
  - 1 means “very confident guess that the cookie is perfectly mouthwatering”
  - 0 means “high confidence that the cookie is an embarrassment”
  - 0.6 means “Man, that’s a tough call, but I’m gonna say yes, and sell it!”
  - 0.5 means complete uncertainty.
- It turns out there’s a nice function that captures this behavior well!
  - Sigmoid function!

Sigmoid Function



z is some representation of our inputs and coefficients, such as:

$$z = \theta_0 + \theta_1 x$$

so that our predictor becomes:

$$h(x) = g(\theta_0 + \theta_1 x)$$

Notice that the sigmoid function transforms our output into the range between **0 and 1**.

Cost function for classification

- Logic behind designing a cost function for classification is different.
- we ask “what does it mean for a guess to be wrong?”
  - This time rule of thumb is if you cannot guess correctly then we are **completely and utterly** wrong! It’s all or none. Not in between.
  - Since you can’t be more wrong than absolutely wrong, the penalty in this case is **enormous**.
  - Alternatively if the we guessed correctly, our cost function should not add any cost for each time this happens.

Cost function for classification

- If our guess was right, but we weren't completely confident ($y = 1$, but $h(x) = 0.8$), this should come with a small cost.
- if our guess was wrong but we weren't completely confident ($y = 1$ but $h(x) = 0.3$), this should come with some significant cost, but not as much as if we were completely wrong.
- This is captured by the **log** function such that:

$$cost = \begin{cases} -\log\big(h(x)\big) & if\ y = 0 \\ -\log\big(1 - h(x)\big) & if\ y = 1 \end{cases}$$

Cost function for classification

- Again, the cost function gives us the average cost over all of our training examples.
- So here we’ve described how the predictor h(x) and the cost function differ between regression and classification, but gradient descent still works fine.
- A classification predictor can be visualized by drawing the boundary line; i.e., the barrier where the prediction changes from a “yes” (a prediction greater than 0.5) to a “no” (a prediction less than 0.5).

Decision boundary (classification boundary)

With a well-designed system, our cookie data can generate a classification boundary as below: