Machine Learning Intro

All knowledge is connected to all other knowledge. The fun is in making the connections.

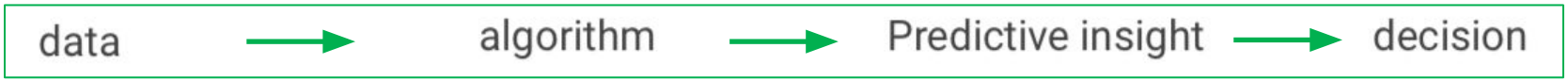
- Arthur Aufderheide

Jay Urbain, Ph.D.
Electrical Engineering and Computer Science Department
Milwaukee School of Engineering

What Is Machine Learning?

- Machine Learning is the science (and art) of programming computers so they can *learn from data*.

- Use standard algorithms to derive predictive insights from data and make repeated decisions.



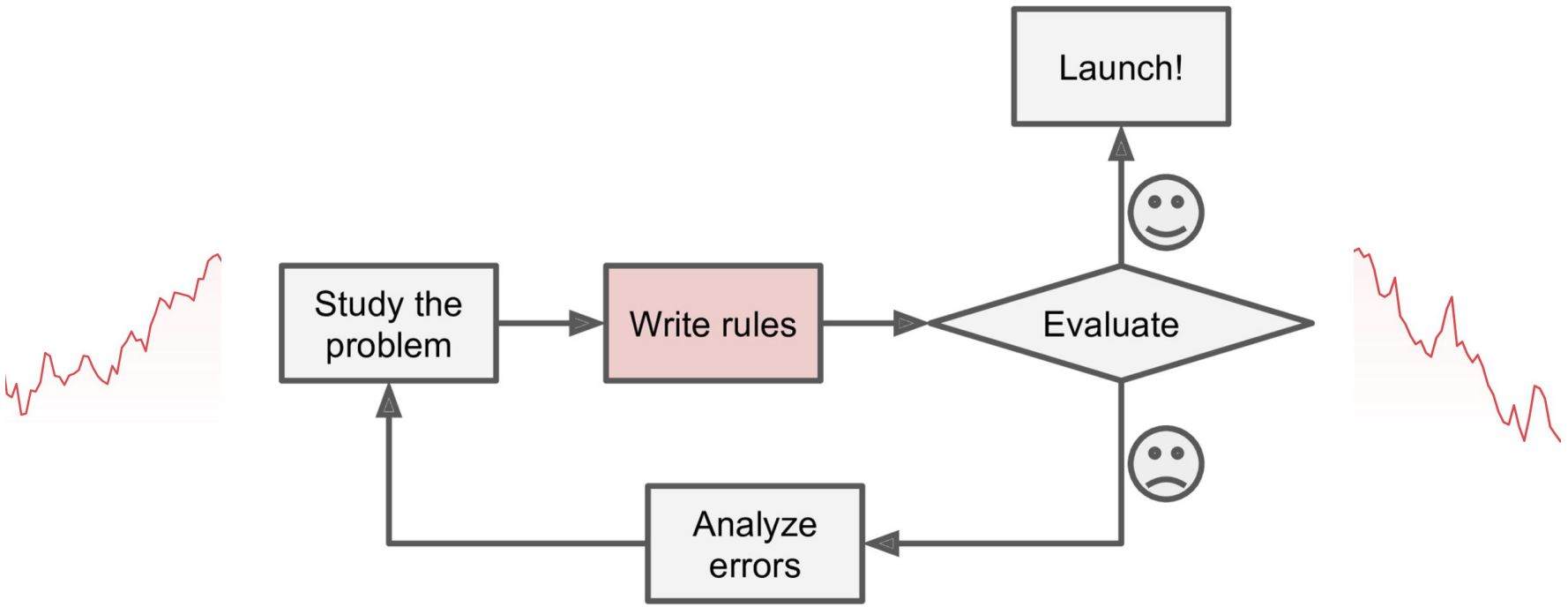
Production-ready Python frameworks:

- Scikit-Learn: easy to use ML library with a well defined API. Implements many Machine Learning algorithms efficiently.
- XGBoost, CatBoost, and LightGBM - gradient boosting libraries.
- PyTorch, TensorFlow/Keras, JAX: more complex libraries for distributed numerical computation.
  - Makes it possible to train and run very large neural networks efficiently by distributing the computations across potentially hundreds of multi-GPU servers.
- *inter alia …*

The main steps in a typical Machine Learning project:

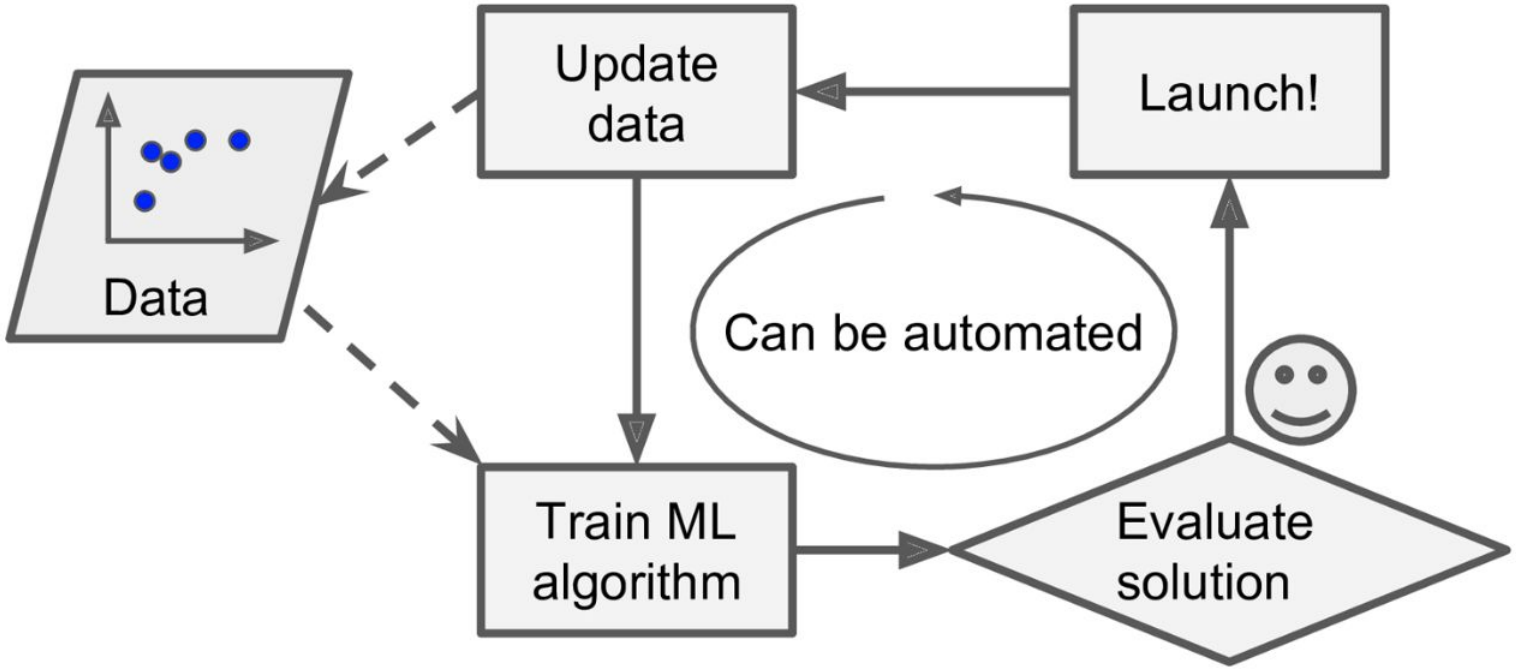
- Handling, cleaning, and preparing data.
- Selecting and engineering features.
  - Which features to use?
  - Normalization?
  - Reducing the dimensionality of the training data to fight the curse of dimensionality.
- Learning by fitting a model to data.
  - Selecting a model.
  - Tuning hyperparameters using cross-validation.
  - Optimizing a cost function
  - The main challenges of Machine Learning, in particular underfitting and overfitting (the bias/variance tradeoff).
- Common learning algorithms: Linear and Polynomial Regression, Logistic Regression, Decision Trees, Random Forests, Neural Networks, and Ensemble methods.

Traditional Development

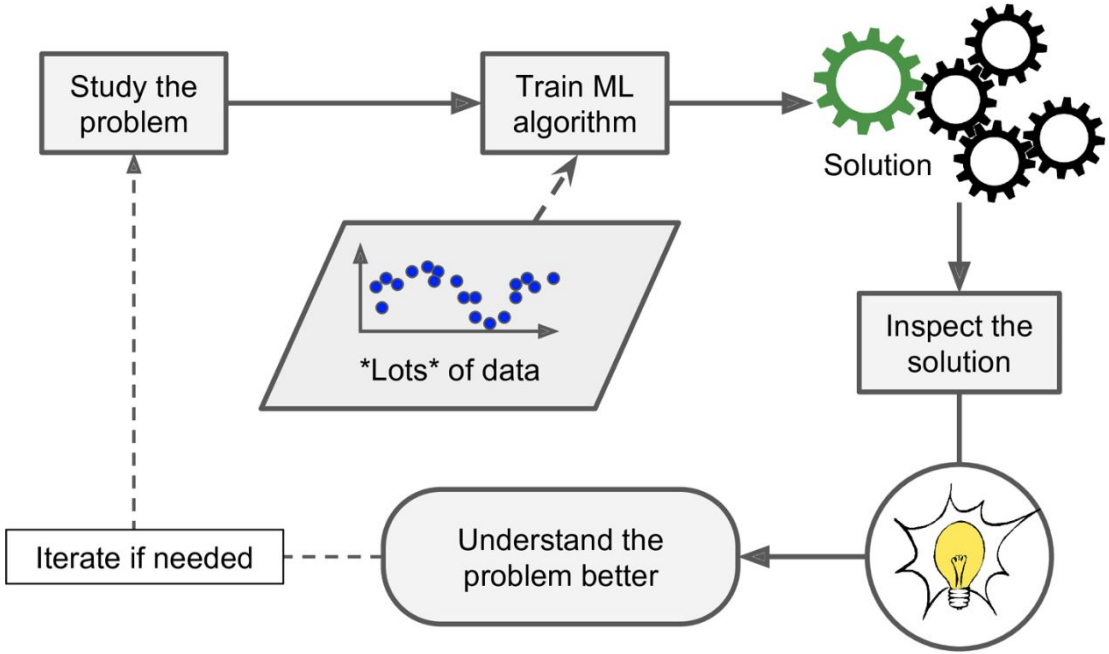


Machine Learning Approach

Automatically Adapting to Change with ML

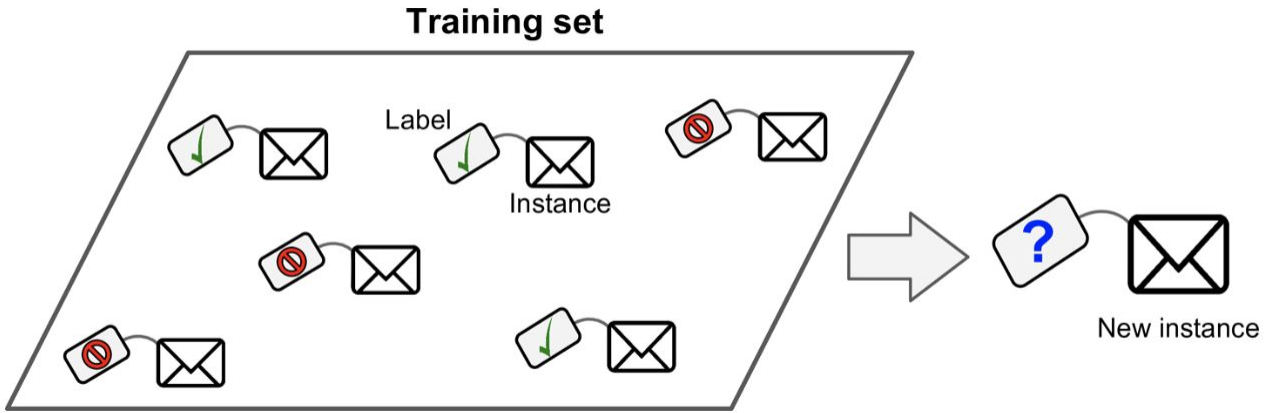


Machine Learning Helping Humans



Supervised Learning

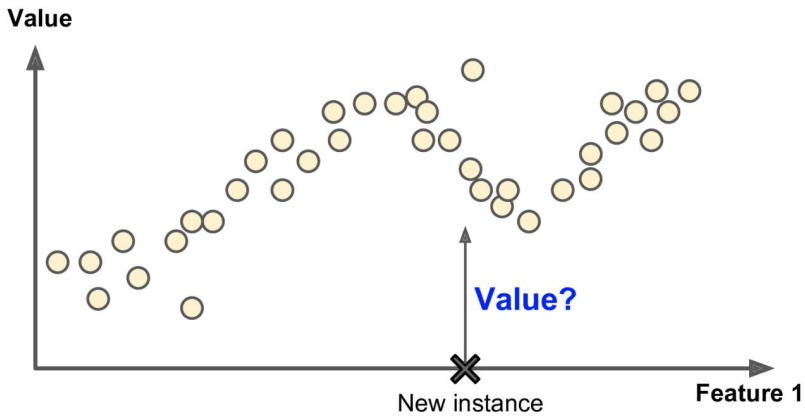
- In *supervised learning*, the training data you feed to the algorithm includes the desired solutions, called *labels.*



Supervised Learning

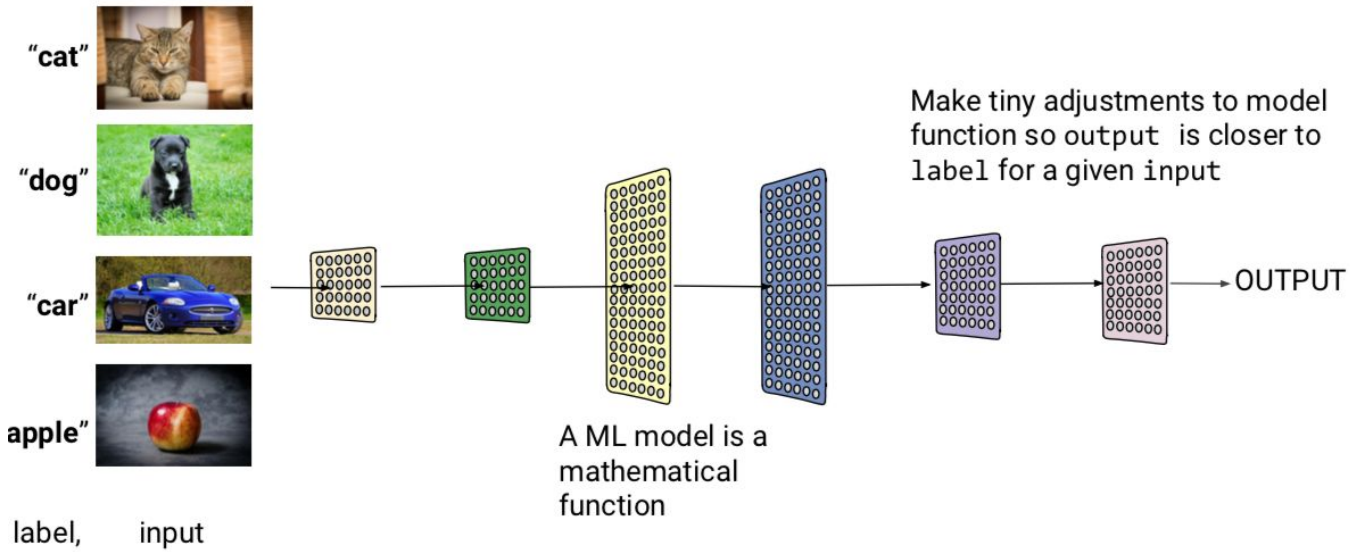
Here are some of the most important supervised learning algorithms:

- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Gradient Boosting
- Neural networks
- Probabilistic Graphical Models

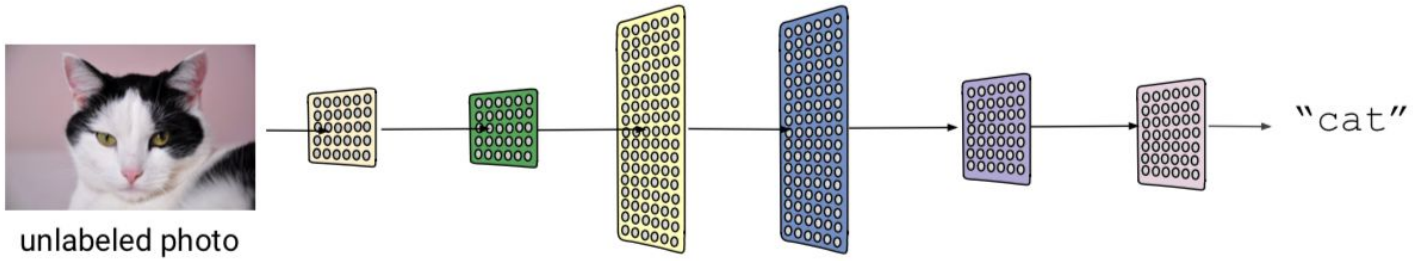


Regression

State 1: Train an ML model with examples



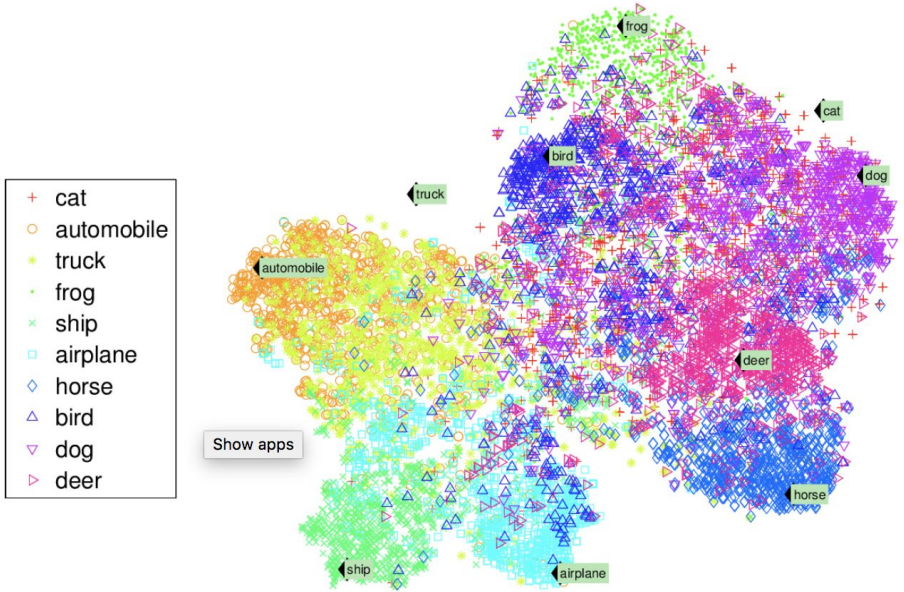
Stage 2: Predict with a trained model



Unsupervised Learning

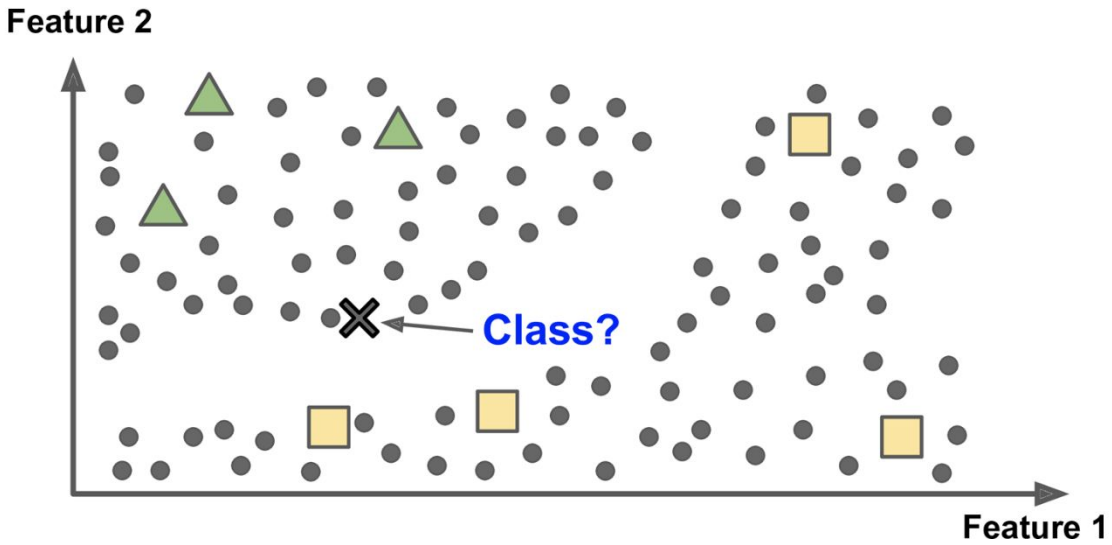
In *unsupervised learning*, the training data is unlabeled.

- Clustering
  - K-Means
  - DBSCAN
  - Hierarchical Cluster Analysis (HCA)
- Anomaly detection and novelty detection
  - One-class SVM
  - Isolation Forest
- Visualization and dimensionality reduction
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally-Linear Embedding (LLE)
  - t-distributed Stochastic Neighbor Embedding (t-SNE)
- Association rule learning
  - Apriori, Eclat



Semi-supervised Learning

- Some algorithms can deal with partially labeled training data, usually a lot of unlabeled data and a little bit of labeled data.



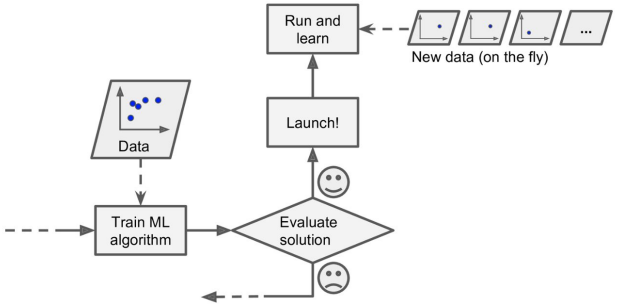
Reinforcement Learning

Reinforcement Learning is a very different.

- The learning system, called an *agent* in this context, can observe the environment, select and perform actions, and get *rewards* in return (or *penalties* in the form of negative rewards).
- It must then learn by itself what is the best strategy, called a *policy*, to get the most reward over time.
- A policy defines what action the agent should choose when it is in a given situation.

Online Learning

- In *online learning*, you train the system incrementally by feeding it data instances sequentially, either individually or by small groups called *mini-batches*.
- Each learning step is fast and cheap, so the system can learn about new data on the fly.
- Online learning is great for systems that receive data as a continuous flow (e.g., stock prices) and need to adapt to change rapidly or autonomously.
  - E.g., Kalman Filtering
- Good for restricted resources.



Framing a (supervised) ML Problem

Cast this as Machine Learning problem:

- What is being predicted?
- What data is needed?

Cast the ML problem as a software problem:

- What is the API for the problem during prediction?
- Who will use this service? How are they doing it today?

Now, cast it in the framework of a data problem:

- What are some key actions to collect, analyze, predict, and react to the data/predictions (different input features might require different actions)

Bad Algorithm

- The only way to know how well a model will generalize to new cases is to actually try it out on new cases.
- Split your data into two sets: the *training set* and the *test set*.
- Train your model using the training set, and test it using the test set.
- The error rate on new cases is called the *generalization error* (or *out-of-sample error*), and by evaluating your model on the test set, you get an estimate of this error. This value tells you how well your model will perform on instances it has never seen before.
- If the training error is low (i.e., your model makes few mistakes on the training set) but the generalization error is high, it means that your model is overfitting the training data.

Linear regression with gradient descent

Machine Learning:
Jay Urbain, PhD

Credits: Nando de Freitas, Oxford; Andrew Ng, Stanford; Hastie and Tibshirani, Stanford

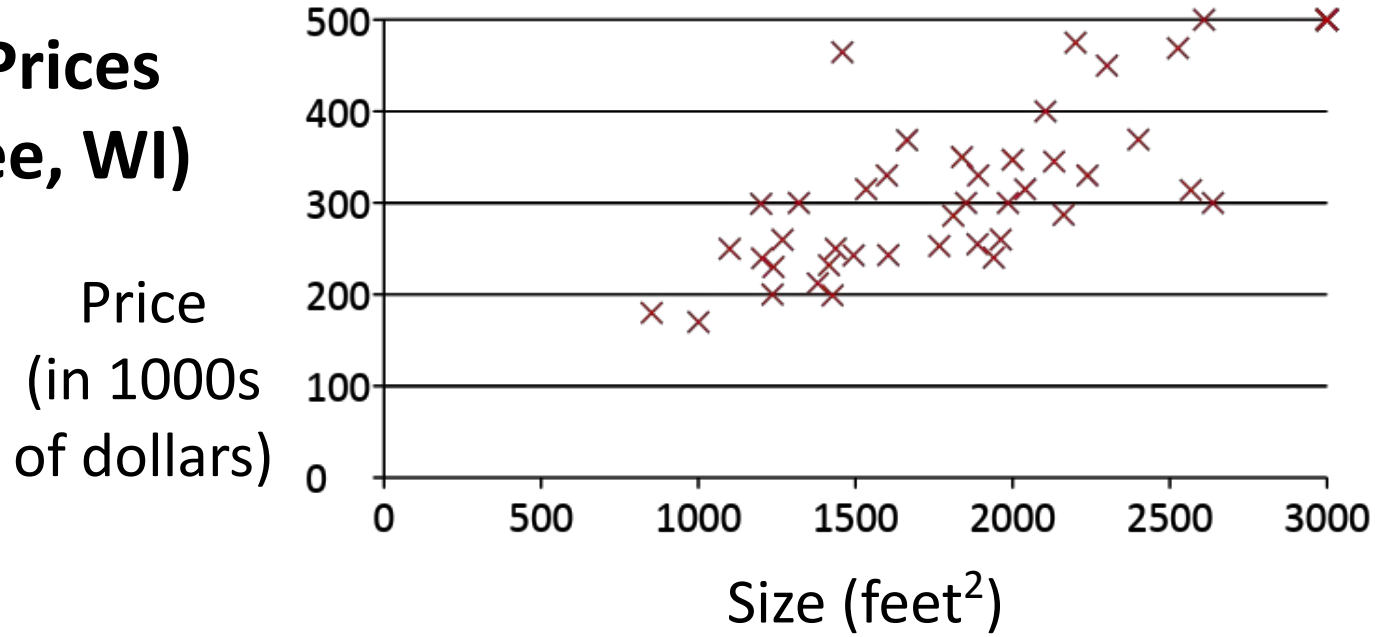
Linear Regression with Gradient Descent

- Linear Regression
  - Hypothesis formulation, hypothesis space
- Optimizing Cost with Gradient Descent
- Using multiple input features with Linear Regression

Notes on:

- Feature Scaling
- Nonlinear Regression

Housing Prices (Milwaukee, WI)



Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real-valued output

Training set of housing prices (Milwaukee, WI)

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Notation:

- **m** = Number of training examples
- **x**'s = "input" variable / features
- **y**'s = "output" variable / "target" variable

$X^{(1)} = 2104$

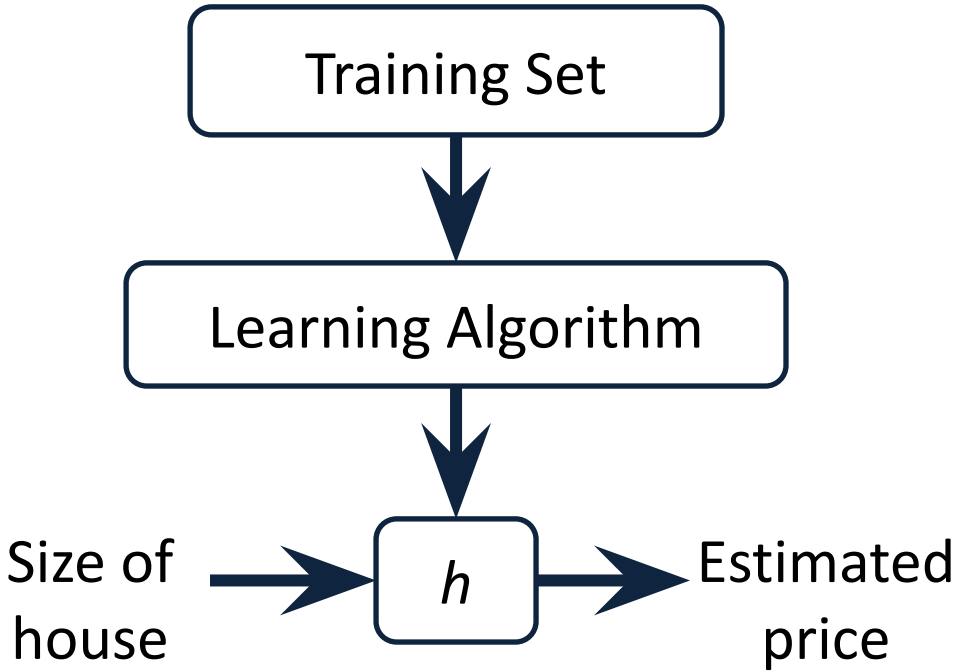
$X^{(2)} = 1416$

$\cdots$

$Y^{(1)} = 460$

(x, y) – one training example

$(x^{(1)}, y^{(1)})$ – i^{th} training example



h – hypothesis to learn (parameterized model)
h maps from x to y

How do we represent *h* ?

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Linear regression with one variable.
(Univariate linear regression)

Training Set

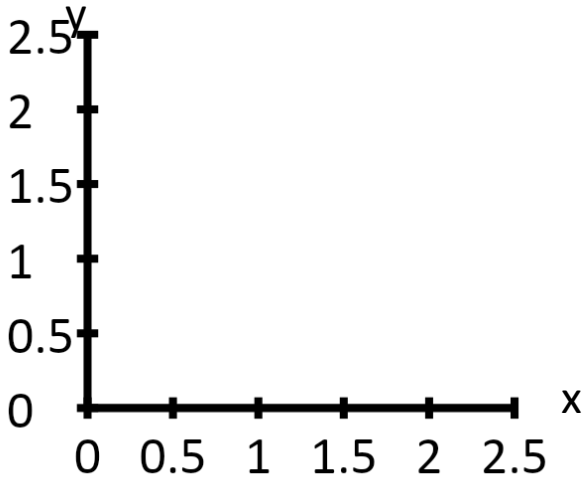
Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

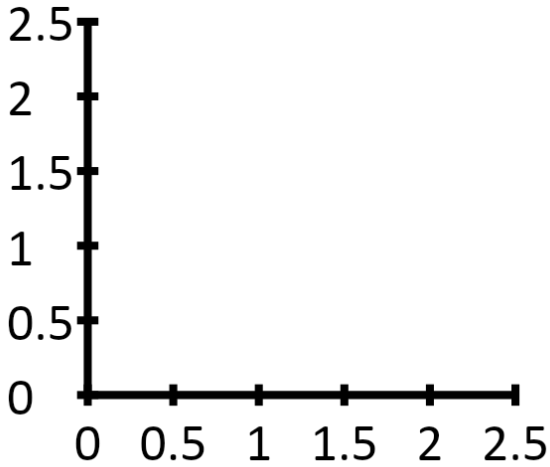
θ_i's: Parameters

How to choose θ_i's ?

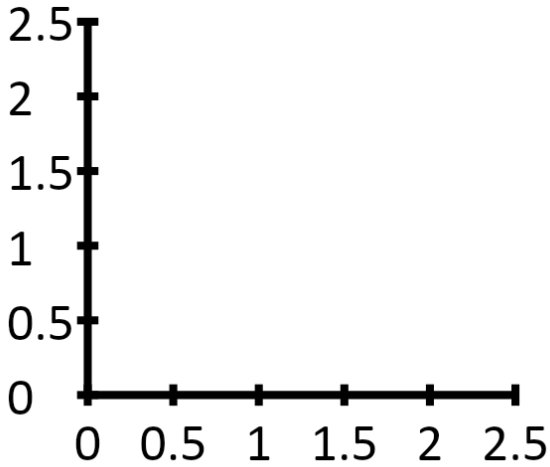
$$h_\theta(x) = \theta_0 + \theta_1 x$$



$\theta_0 = 1.5$
$\theta_1 = 0$



$\theta_0 = 0$
$\theta_1 = 0.5$



$\theta_0 = 1$
$\theta_1 = 0.5$

Draw h(x)

Simplified

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$h_\theta(x) = \theta_1 x$$

Parameters:

$$\theta_0, \theta_1$$

$$\theta_1$$

Cost Function:

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

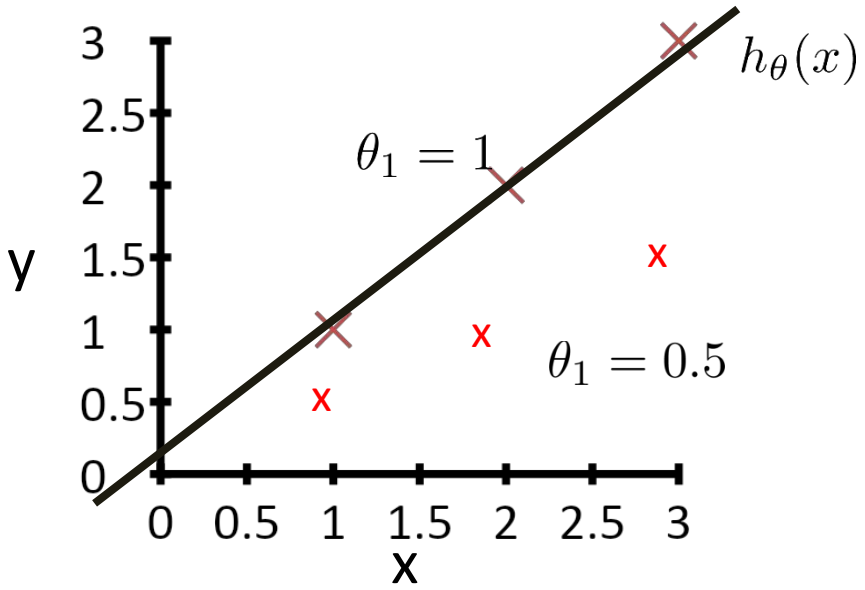
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$

$h_\theta(x)$

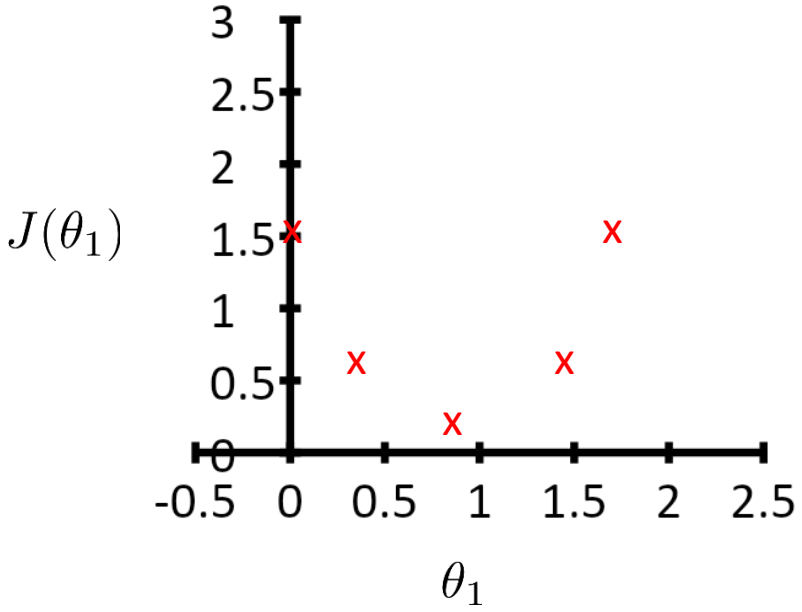
(for fixed θ_1, this is a function of x)



$$h_\theta(x) = \theta_1 x$$

$J(\theta_1)$

(function of the parameter θ_1)



$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Note: $h(x^{(i)})-y^{(i)}=0$, $J^{(i)}= 1/(2m)*(0^2+0^2+0^2)=0$

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Fitting data well: least squares cost function

In regression, you almost always want to fit the data well

- smallest average distance to points in training data
  (h(x) close to y for (x, y) in training data)

Cost function ***J***:

Number of training instances

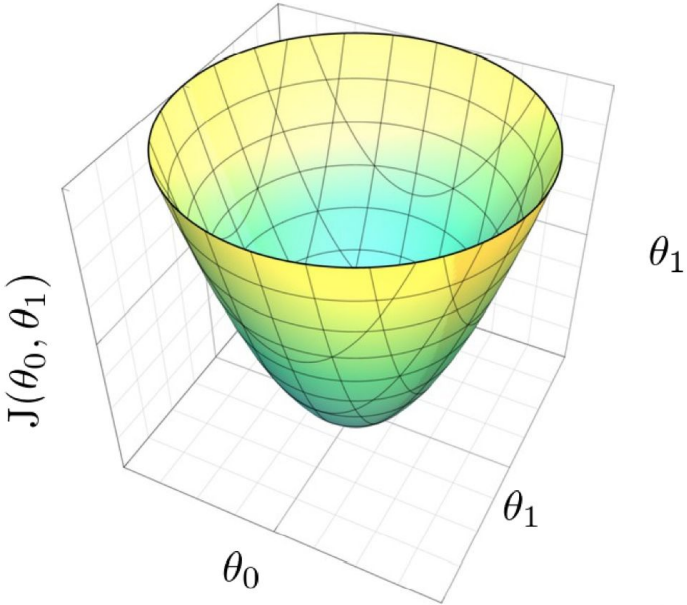
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$
$$= \frac{1}{2m} \sum_{i=1}^{m} \left((\theta_0 + \theta_1 x^{(i)}) - y^{(i)}\right)^2$$

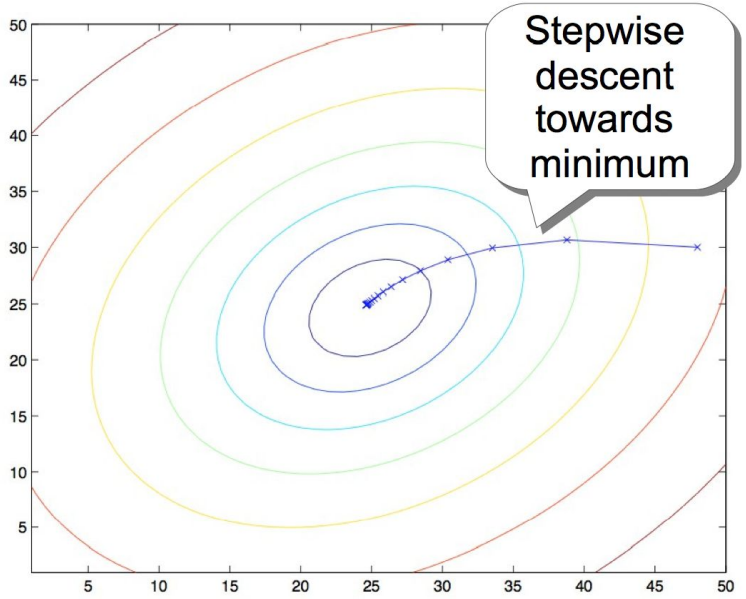
Squaring

- Penalty for positive and negative deviations the same
- Penalty for large deviations stronger

Optimizing Cost with Gradient Descent

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





Stepwise descent towards minimum

Derivatives work only for few parameters

$$\underset{\theta_0 \theta_1}{minimize}\, J(\theta_0, \theta_1) = \underset{\theta}{arg\ min} J(\theta)$$

Have some function $J(\theta_0, \theta_1)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

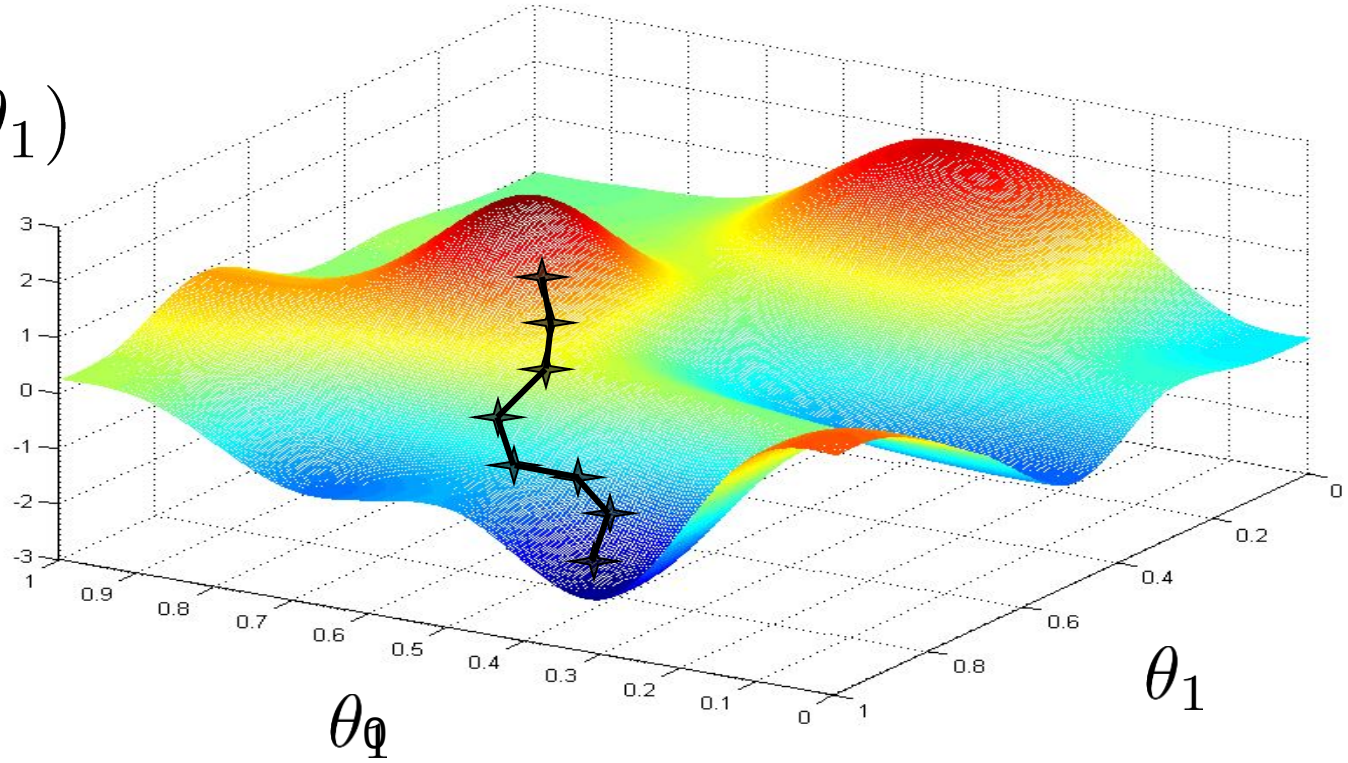
Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

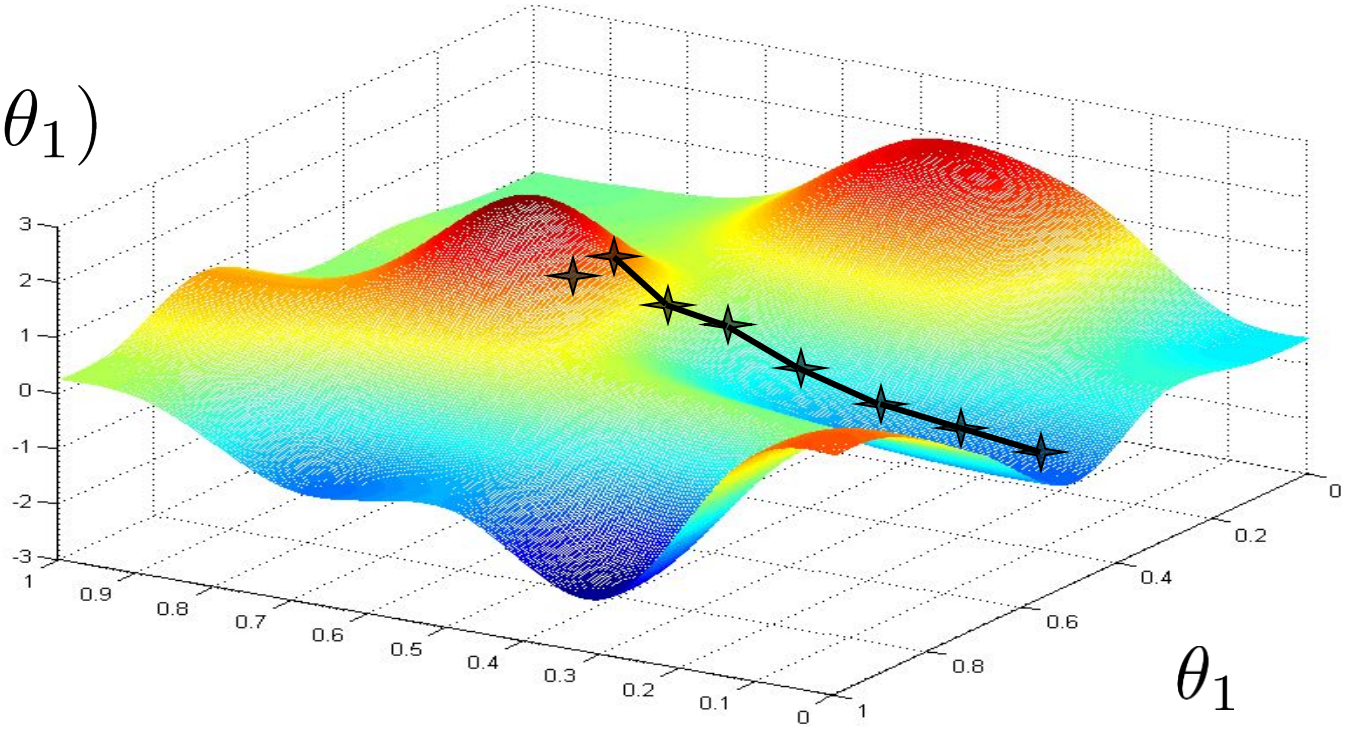
  until we hopefully end up at a minimum

How?

$J(\theta_0, \theta_1)$



$J(\theta_0, \theta_1)$



θ_1

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (\text{for } j = 0 \text{ and } j = 1)$$

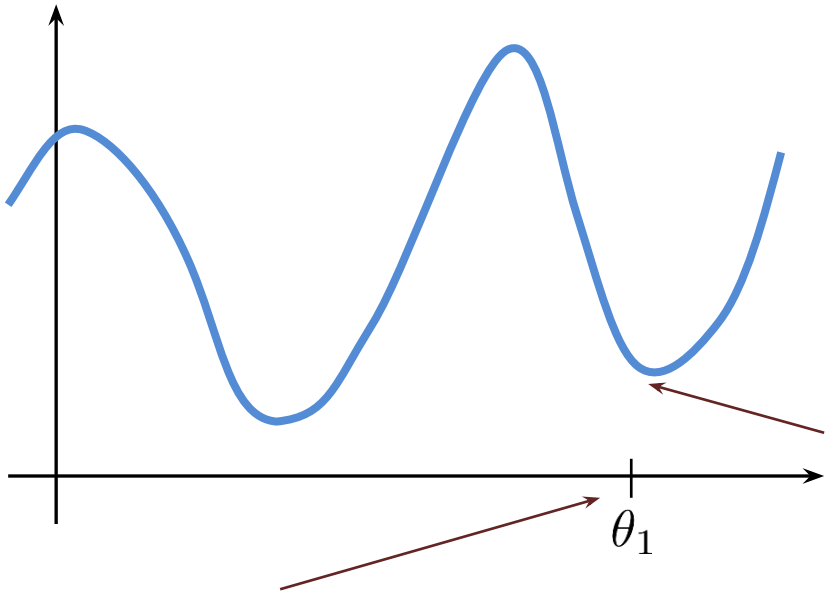
}

Correct: Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

Incorrect:

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_1 := \text{temp1}$$



θ_1 at local optima

Current value of θ_1

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

Linear regression with one variable

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

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

Gradient descent algorithm

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

}

update θ_0 and θ_1 simultaneously

“Batch” versus “Online” versus “Minibatch” Gradient Descent

“Batch”: Each step of gradient descent uses all the training examples.

“Online”: Each step of gradient descent uses *a* training example at a time.

“Minibatch”: Smaller sampled batches of training examples.

Multiple features (variables).

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

$\}$ (simultaneously update for every $j = 0, \ldots, n$)

Gradient Descent

Previously (n=1):

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial \theta_0} J(\theta)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$$

(simultaneously update θ_0, θ_1)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

(simultaneously update θ_j for $j = 0, \ldots, n$)

$\}$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_2^{(i)}$$

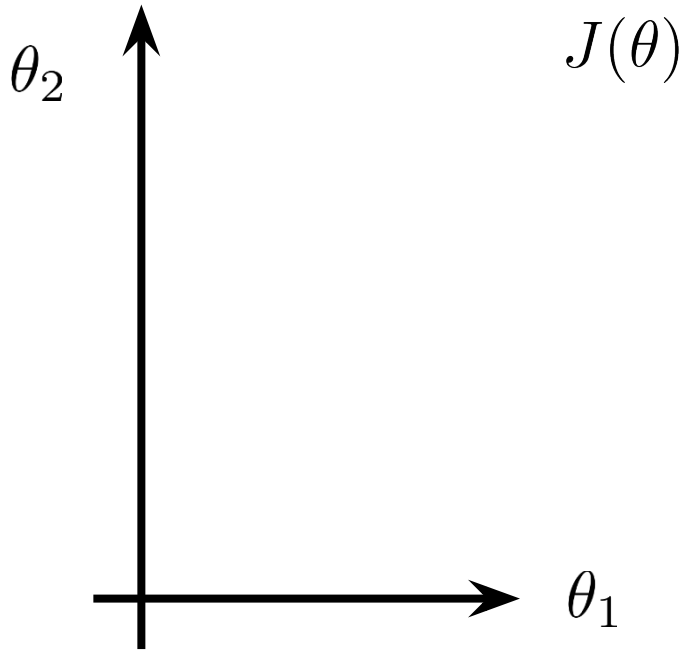
$\ldots$

Feature Scaling

Idea: Make sure features are on a similar scale.

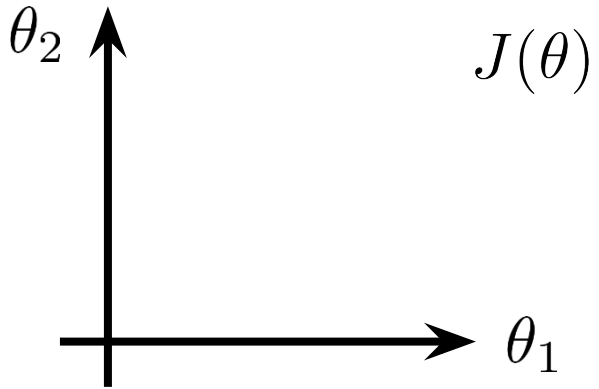
E.g. x_1 = size (0-2000 feet^2)

x_2 = number of bedrooms (1-5)



$$x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

$$x_2 = \frac{\text{number of bedrooms}}{5}$$



Mean normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$).

E.g. $x_1 = \frac{size - 1000}{2000}$ $x_2 = \frac{\#bedrooms - 2}{5}$

$$-0.5 \leq x_1 \leq 0.5, -0.5 \leq x_2 \leq 0.5$$

Z-score (standard score) normalization

$$z = \frac{x - \mu}{\sigma}$$

Standard score is the signed number of standard deviations by which the value of an observation or data point is above the mean value.

Min-max (0-1) normalization

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

Problem: considering different cities for opening a new outlet. The chain already has trucks in various cities and you have data for profits and populations from the cities. You would like to use this data to help you select which city to expand to next.

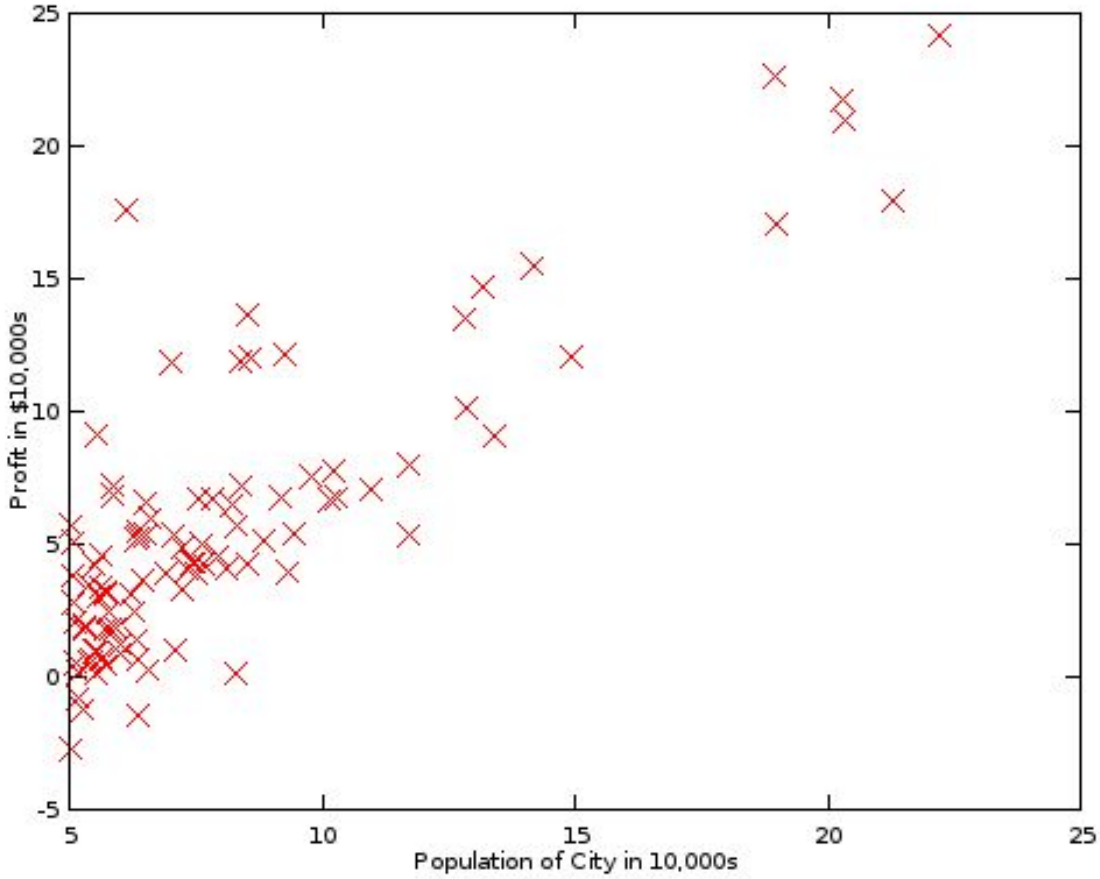
X =

1.0000	6.1101
1.0000	5.5277
1.0000	8.5186
1.0000	7.0032
1.0000	5.8598
1.0000	8.3829
1.0000	7.4764
1.0000	8.5781
1.0000	6.4862
1.0000	5.0546
1.0000	5.7107
1.0000	14.1640
1.0000	5.7340

y =

17.59200
9.13020
13.66200
11.85400
6.82330
11.88600
4.34830
12.00000
6.59870
3.81660
3.25220
15.50500
3.15510

Visualize data

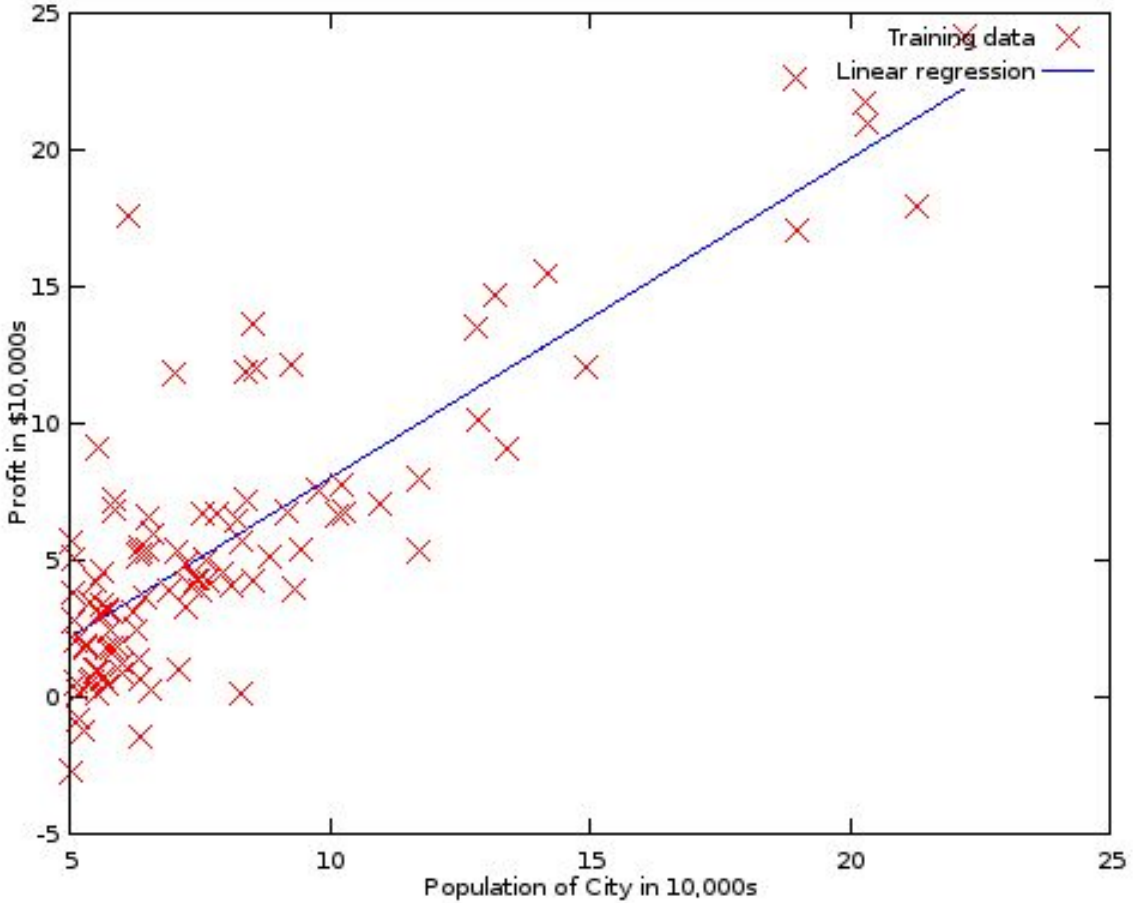


Run linear regression

Theta found by gradient descent: -3.630291 1.166362

For population = 35,000, we predict a profit of 4519.767868

For population = 70,000, we predict a profit of 45342.450129



Assessing the accuracy of model coefficients

Linear regression with residual term. Represents what we can't explain with our model.

$$Y = \beta_0 + \beta_1 X + \epsilon.$$

RSS measures the amount of variability that is left unexplained after performing the regression

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

TSS (Total sum of squares) measures the total variance when measuring the response y.

$$\text{TSS} = \sum (y_i - \bar{y})^2$$

R^2 amount of variance explained by our model

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

The RSE is an estimate of the standard deviation of ε. It is basically the average amount that the response will deviate from the true regression line.

$$\text{RSE} = \sqrt{\frac{1}{n-2}\text{RSS}} = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}.$$

Notes on:

- Feature Scaling
- Nonlinear Regression
- Optimizing Cost using derivatives

Demo gradient descent algorithm
gradient_descent_assignment_solution.ipynb

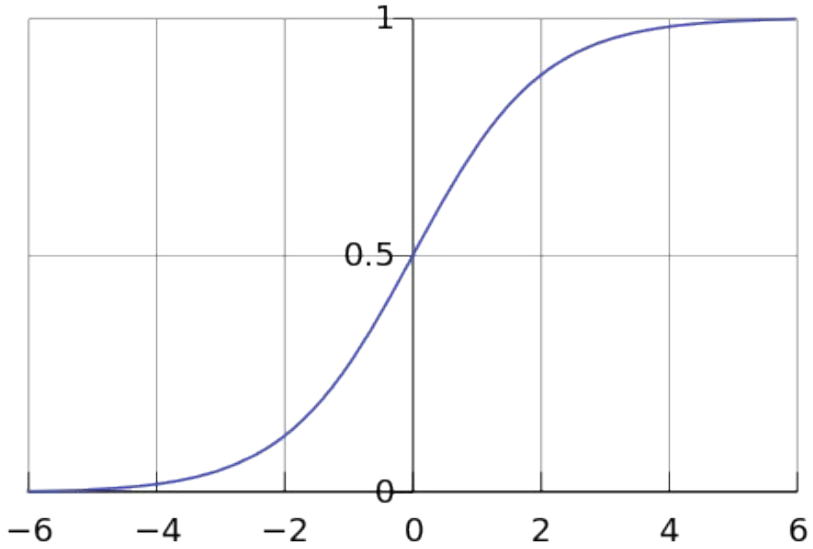


Image: Wikipedia

Logistic regression Classification

Data Mining:
Jay Urbain, PhD

Credits: Andrew Ng, Stanford; Hastie and Tibshirani, Stanford

Classification

Email: Spam / Not Spam?
Online Transactions: Fraudulent (Yes / No)?
Tumor: Malignant / Benign ?

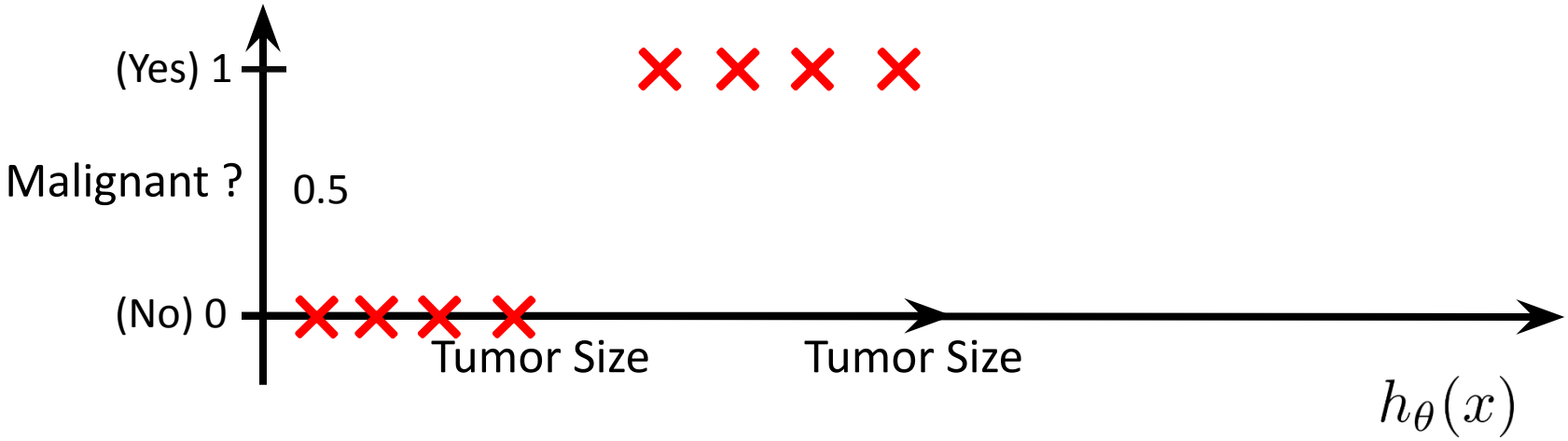
$y \in \{0, 1\}$

0: “Negative Class” (e.g., benign tumor)
1: “Positive Class” (e.g., malignant tumor)

Multinomial classification: y {0, 1, 2, 3,…}

Linear Regression for classification

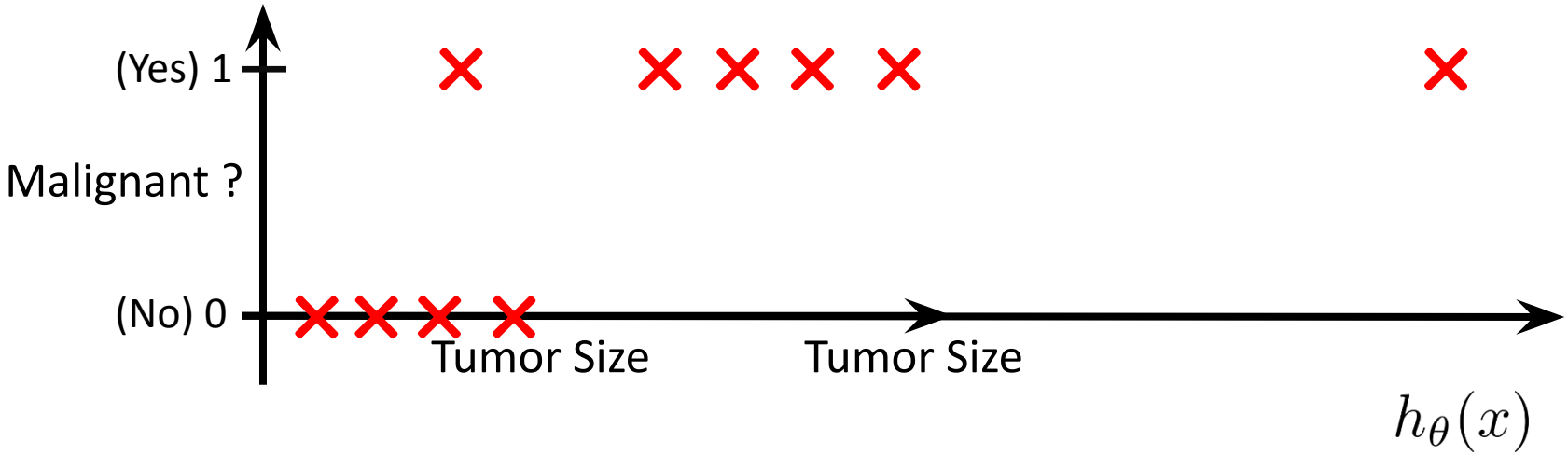
- Linear regression can be used for classification in domains with numeric attributes.
  - Perform a regression for each class, set output to 1 for instances that belong to the class, and 0 for those that do not.
  - The result is a linear expression for each class.
  - Then, given a test instance of an unknown class, calculate the value of each linear expression and choose the one that is **largest**.
  - *Form of multinomial linear regression.*
  - Problems: output is not a proper probability, assumes errors are not statistically significant.



Threshold classifier output $h_\theta(x)$ at 0.5:

If $h_\theta(x) \geq 0.5$, predict “y = 1”

If $h_\theta(x) < 0.5$, predict “y = 0”



Threshold classifier output $h_\theta(x)$ at 0.5:

If $h_\theta(x) \geq 0.5$, predict “y = 1”

else $h_\theta(x) < 0.5$, predict “y = 0”

Classification: y = 0 or 1

$h_\theta(x)$ can be > 1 or < 0

With regression

Logistic *Regression*: $0 \leq h_\theta(x) \leq 1$

Want proper probability

Classification

Logistic Regression Model: hypothesis representation

Want $0 \leq h_\theta(x) \leq 1$

$$h_\theta(x) = \theta^T x$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

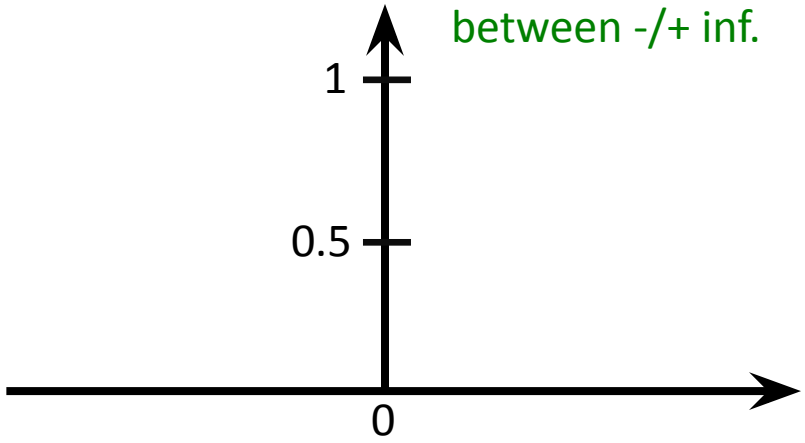
Sigmoid function

==

Logistic function

Replaces original target variable which can not be approximated easily with a nonlinear function.

Input variables can range between -/+ inf.



$$h_\theta(x) = g(\ \theta^T x\ )$$

Interpretation of Hypothesis Output

$h_\theta(x)$ = estimated probability that y = 1 on input x

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$$h_\theta(x) = 0.7$$

Tell patient that 70% chance of tumor being malignant.

Generates proper probability. Use threshold to select classification.

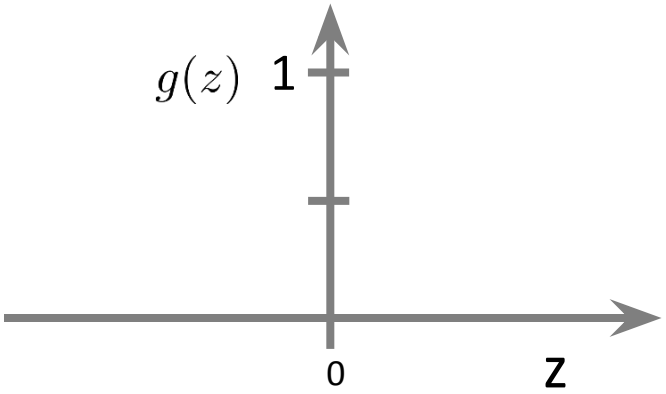
"probability that y = 1, given x, parameterized by θ"

$$P(y = 0|x;\theta) + P(y = 1|x;\theta) = 1$$
$$P(y = 0|x;\theta) = 1 - P(y = 1|x;\theta)$$

Logistic regression decision boundary

$$h_\theta(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$



predict "$y = 1$" if $h_\theta(x) \geq 0.5$

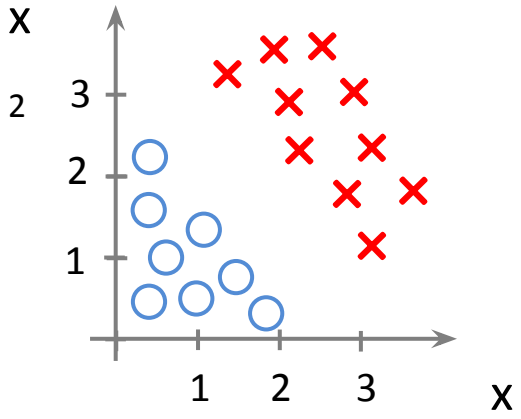
g(z) >= 0.5 when z >=0
g(z) < 0.5 when z < 0

$$h_\theta(x) = \theta^T x$$

1/(1 + EXP(-(0.0001))) => 0.500025

predict "$y = 0$" if $h_\theta(x) < 0.5$

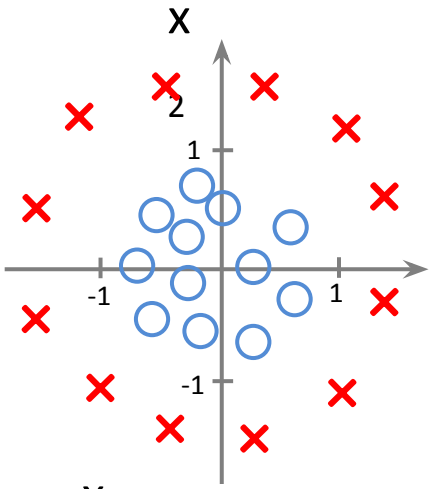
Decision Boundary



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

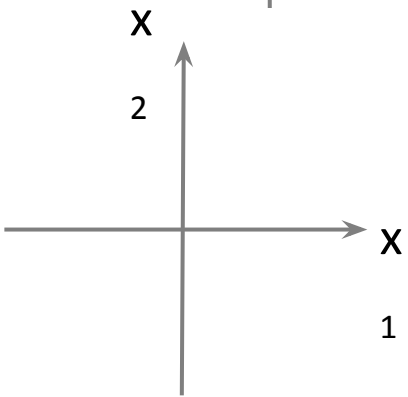
Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

Non-linear decision boundaries



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

m examples $x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}$ $x_0 = 1, y \in \{0, 1\}$

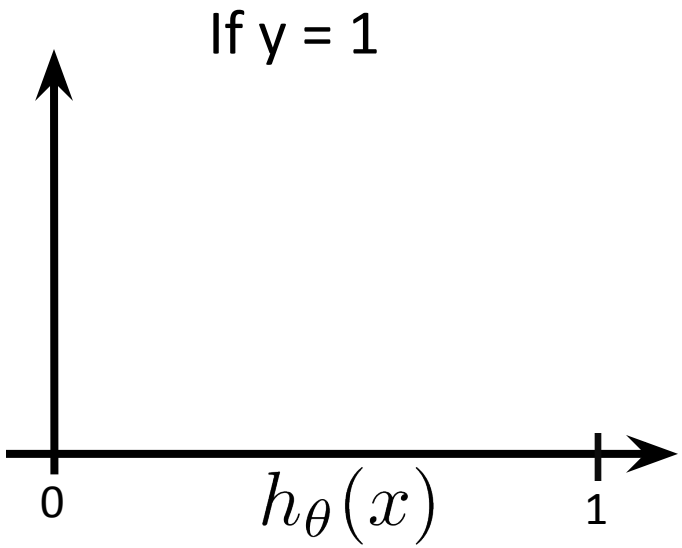
$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

How to choose parameters θ ?

Logistic regression cost function

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



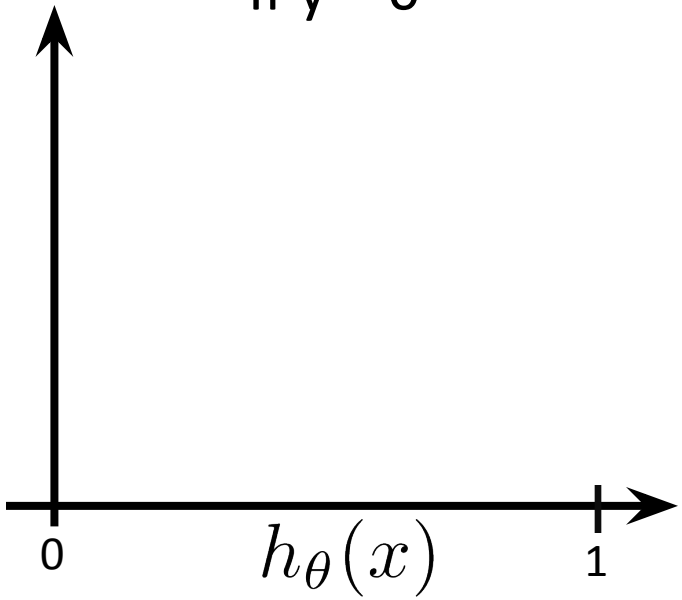
$\text{Cost} = 0$ if $y = 1, h_\theta(x) = 1$
But as $h_\theta(x) \to 0$
$Cost \to \infty$

Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

Logistic regression cost function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

If y = 0



Logistic regression cost function with gradient Descent

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or 1 always

Logistic regression cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

To fit parameters θ:

$$\min_\theta J(\theta)$$

To make a prediction given new: x

Output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

Logistic regression cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m} [\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

To fit parameters θ:

$$\min_\theta J(\theta)$$

To make a prediction given new x:

Output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

Gradient Descent

$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$\}$ (simultaneously update all θ_j)

Gradient Descent

$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$\}$

Algorithm looks identical to linear regression!

Softmax

- Generalization of the logistic function that "squashes" a ***K***-dimensional vector into a range [0,1]
- Softmax can be used in multiclass classification methods.
- In multinomial logistic regression, the input to the function is the result of ***K*** distinct linear functions, and the predicted probability for the ***j'th*** class given a sample vector x and a weighting vector w is:

$$P(y = j|\mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$

Sigmoid

2 classes

$$\text{out} = P(Y=\text{class1}|X)$$

SoftMax

k>2 classes

$$\text{out} = \begin{bmatrix} P(Y=\text{class1}|X) \\ P(Y=\text{class2}|X) \\ P(Y=\text{class3}|X) \\ \vdots \\ P(Y=\text{classk}|X) \end{bmatrix}$$

Loss Function Example

One common loss for classification: **cross entropy (CE)**. Label y is a categorical vector (one-hot encoding).

$y =$

0	0	1	0	0

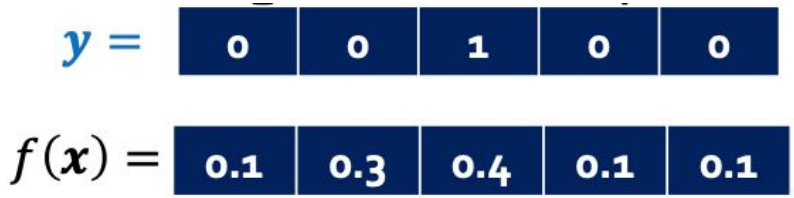
f(x) = Softmax(g(x)) $f(\boldsymbol{x})_i = \frac{e^{g(x)_i}}{\sum_{j=1}^{C} e^{g(x)_j}}$

Where ***C*** is the number of classes. E.g.,

$f(\boldsymbol{x}) =$

0.1	0.3	0.4	0.1	0.1

Loss Function Example: CE



$$CE(y, f(x)) = -\sum_{i=1}^{C}(y_i log(f(x)_i))$$

- y_i and $f(x)_i$ are the actual and predicted values of the i-th class.
- Intuition:the lower the loss, the closer the prediction is to the one-hot ground truth.
- Total loss over all training examples: $L = \sum_{(x,y)\in T} CE(y, f(x))$

T: training set containing paris of data and labels (x, y)