Linear and Logistic Regression

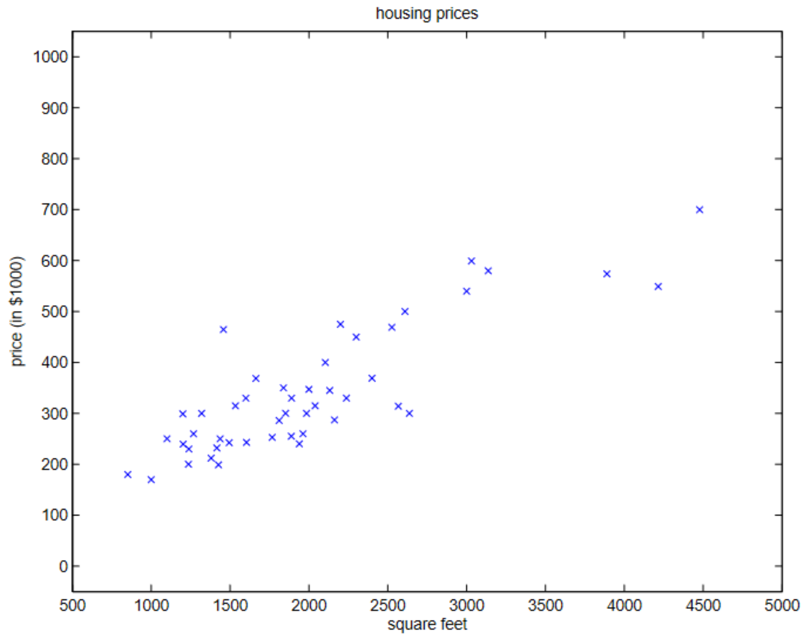
Amirhossein Abaskohi
University of Tehran ACM Summer School 2021

Supervised learning

- Let’s start by talking about a few examples of supervised learning problems. Suppose we have a dataset giving the living areas and prices of 47 houses from Portland, Oregon:

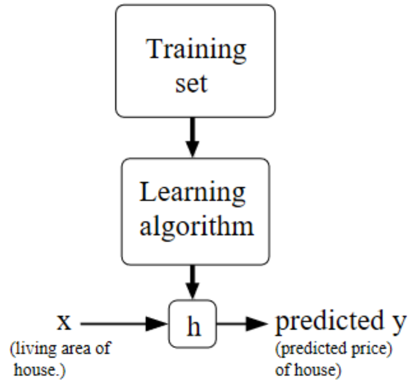
Living area(feet^2)	Price(1000$)
2104	400
1600	330
2400	369
...	...

Plottting data



Predicting house prices

- Given data like this, how can we learn to predict the prices of other houses in Portland, as a function of the size of their living areas?



Regression and classification

When the target variable that we're trying to predict is continuous, such as in our housing example, we call the learning problem a **regression** problem.

When y can take on only a small number of discrete values (such as if, given the living area, we wanted to predict if a dwelling is a house or an apartment, say), we call it a **classification** problem.

Linear Regression

- To make our housing example more interesting, let’s consider a slightly richer dataset in which we also know the number of bedrooms in each house:

#bedrooms	Living area(feet^2)	Price(1000$)
3	2104	400
3	1600	330
3	2400	369
	...	...

Linear Regression

To perform supervised learning, we must decide how we’re going to represent functions/hypotheses h in a computer. As an initial choice, let’s say we decide to approximate y as a linear function of x:

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

Here the θs are the parameters (also called weights).

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

Linear Regression

Now, given a training set, how do we pick, or learn, the parameters θ? One reasonable method seems to be to make h(x) close to y, at least for the training examples we have. To formalize this, we will define a function that measures, for each value of the θ's, how close the h(x(i))'s are to the corresponding y(i)'s. We define the **cost function**:

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

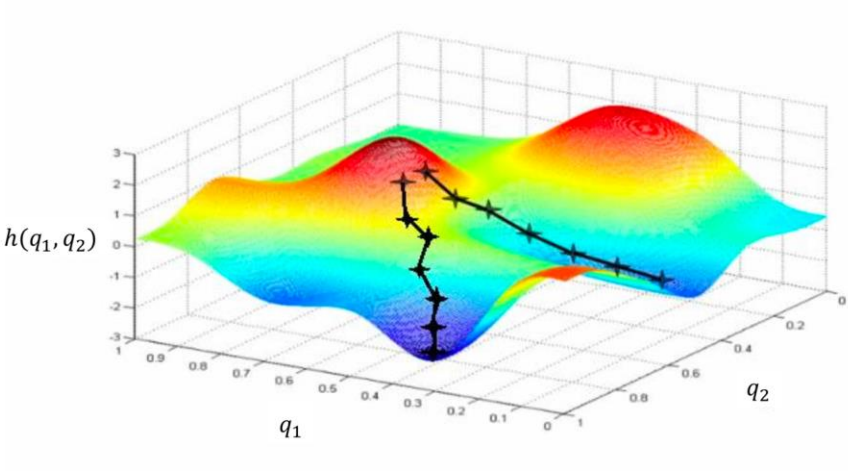
LMS algorithm

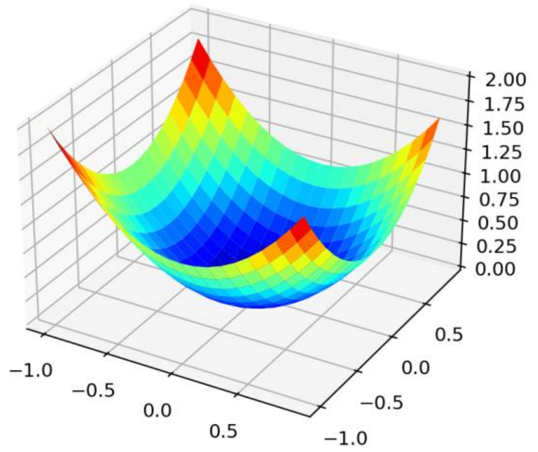
We want to choose θ so as to minimize J(θ). To do so, lets use a search algorithm that starts with some "initial guess" for θ, and that repeatedly changes θ to make J(θ) smaller, until hopefully we converge to a value of θ that minimizes J(θ). Let's consider **gradient descent** algorithm.

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

LMS algorithm

Non-convex Example





LMS algorithm

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(x) - y\right)^2 \\
&= 2 \cdot \frac{1}{2} \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(h_\theta(x) - y\right) \\
&= \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= \left(h_\theta(x) - y\right) x_j
\end{aligned}$$

LMS algorithm

For a single training example, this gives the update rule:

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

The rule is called the LMS update rule (LMS stands for “least mean squares”), and is also known as the Widrow-Hoff learning rule.

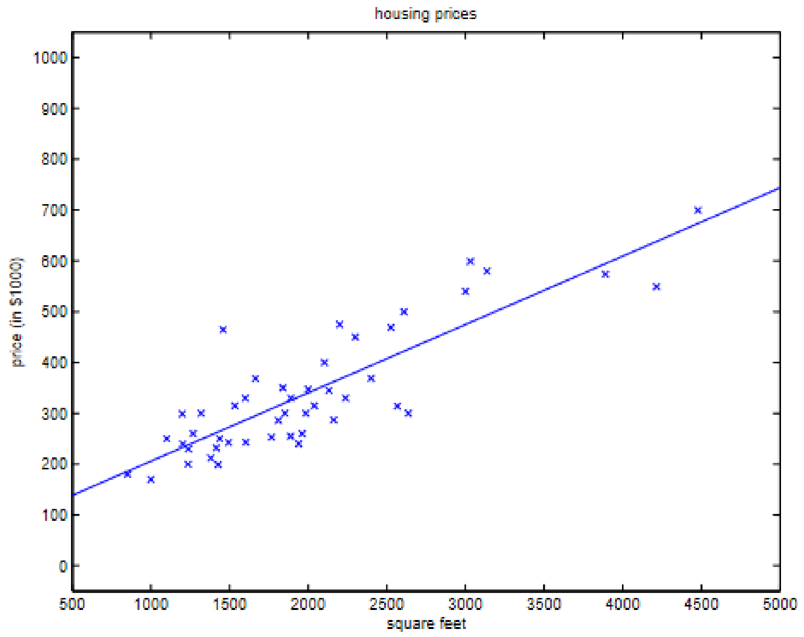
Gradient descent

```
Repeat until convergence {
    $\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$    (for every $j$).
}
```

So, this is simply gradient descent on the original cost function J. This method looks at every example in the entire training set on every step, and is called batch **gradient descent**.

Gradient descent can be susceptible to local minima in general.

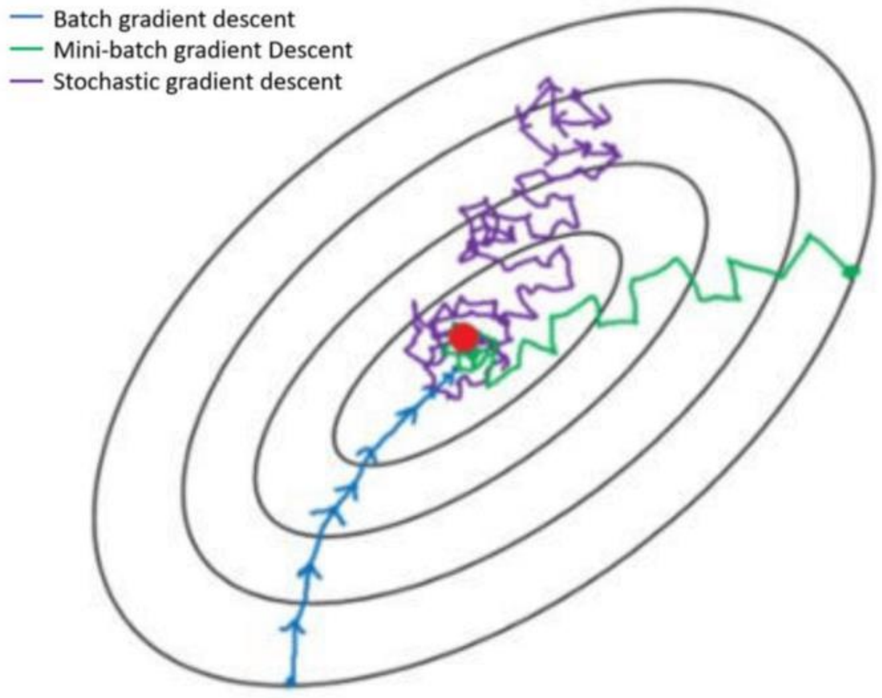
Linear regression



Stochastic gradient descent(SGD)

```
Loop {
    for i=1 to m, {
        $\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$    (for every $j$).
    }
}
```

Gradient descent



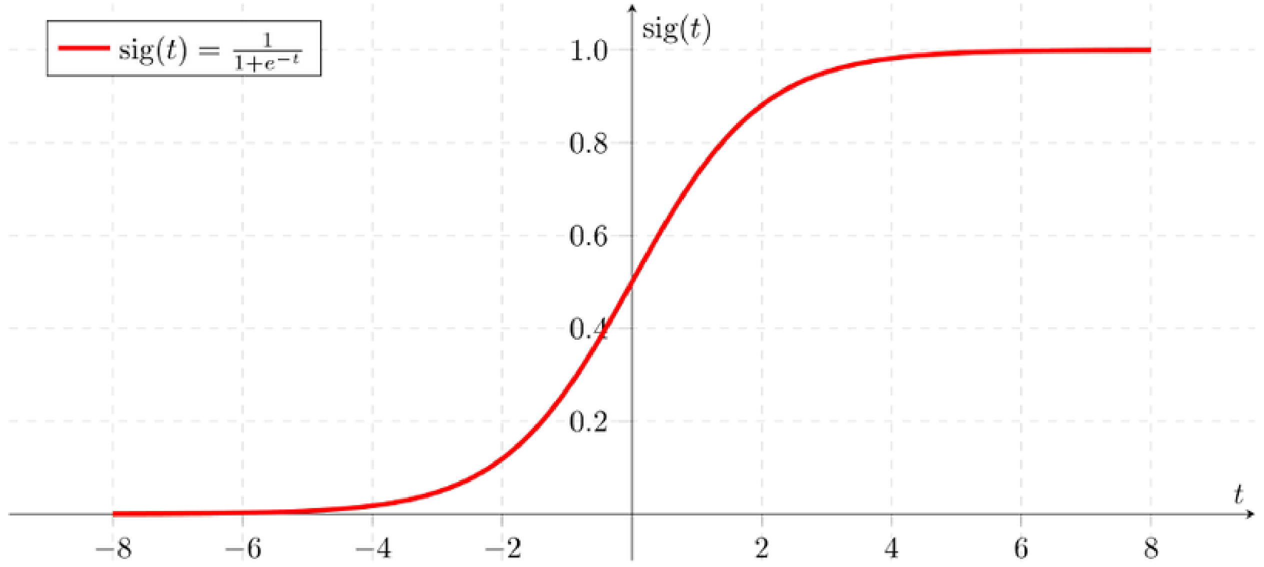
Logistic Regression

- This justifies the name ‘logistic regression’. Data is fit into linear regression model, which then be acted upon by a logistic function predicting the target categorical dependent variable.
- Types of Logistic Regression:
  - Binary Logistic classification
  - Multinomial Logistic Regression

Decision Boundary

- To predict which class a data belongs, a threshold can be set. Based upon this threshold, the obtained estimated probability is classified into classes.
- Say, if predicted_value ≥ 0.5, then classify email as spam else as not spam.
- Decision boundary can be linear or non-linear. Polynomial order can be increased to get complex decision boundary.

Sigmoid function



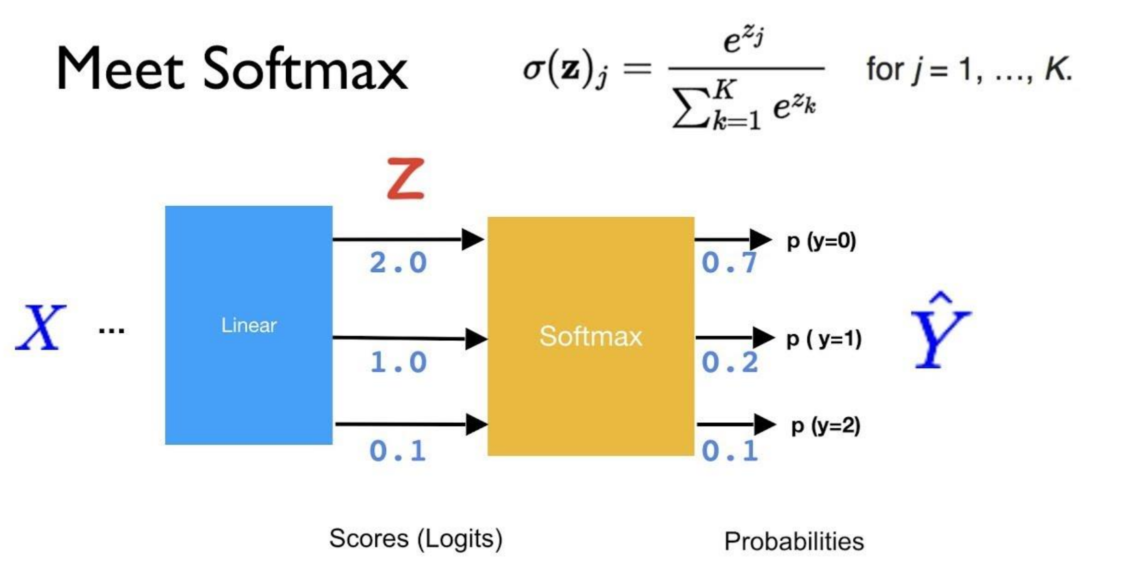
Cost function

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

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

$$m = number\ of\ samples$$

Softmax and multi-class classification


Meet Softmax
$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$ for $j = 1, \ldots, K$.
Z
X
...
Linear
2.0
1.0
0.1
Softmax
0.7
0.2
0.1
p (y=0)
p (y=1)
p (y=2)
$\hat{Y}$
Scores (Logits)
Probabilities