

Regression

Given examples $\langle \mathbf{X}, \mathbf{y} \rangle$ a *regression task* is to predict

- a continuous \mathbf{y}
- from a vector of features \mathbf{x}

This differs from a *Classification* task (e.g., predicting the digit represented by an image)

- where the \mathbf{y} are *discrete* values

To be concrete: imagine we need to predict the Price \hat{y} of a house given only its Size \mathbf{x} .

We could imagine an approach similar to the KNN algorithm used for classification

- compare \mathbf{x} to each $\mathbf{x}^{(i)}$ in the training set \mathbf{X}
 - measure the "distance" from \mathbf{x} to $\mathbf{x}^{(i)}$ to come up with a weight
- predict \hat{y} as the weighted average of the \mathbf{y}^i

A strong criticism of KNN is that Θ , the parameters, comprised all m training examples

- large
- memorization versus generalization

The fact that \mathbf{y} is *continuous* rather than discrete opens the possibility of a *numerical* relationship between features \mathbf{x} and labels \mathbf{y} .

We will take advantage of this in our first Regression model.

Linear Regression

Our first predictor/estimator/model is called Linear Regression.

Linear Regression restricts the form of relationship between \mathbf{y} and \mathbf{x} to

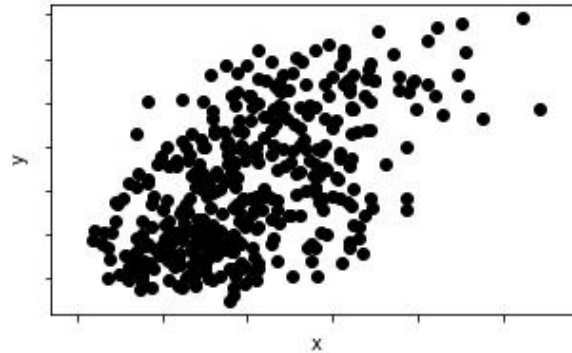
$$\hat{\mathbf{y}} = \Theta^T \cdot \mathbf{x}$$

That is: the predicted $\hat{\mathbf{y}}$ is a linearly-weighted (with weights from vector Θ) sum of features \mathbf{x} .

Anyone who has fit a straight line to a cloud of points has performed Linear Regression.

A straight line has intercept Θ_0 and slope Θ_1

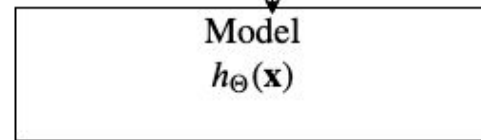
$$\hat{y} = \Theta_0 + \Theta_1 * x_1$$



$\langle \mathbf{X}, \mathbf{y} \rangle$

Training examples

$\mathbf{X} : m \times n$
 $\mathbf{y} : n \times 1$



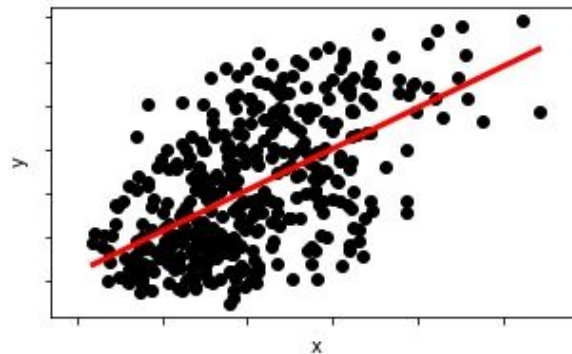
Fitting, training

Fit



Θ

$\Theta : n \times 1$



In our example

- we expect the Price to increase with Size \mathbf{x}_1
 - Θ_1 tells us how much each extra unit of Size increases the Price

Rather than writing the intercept Θ_0 as a separate term we can modify \mathbf{x} and Θ

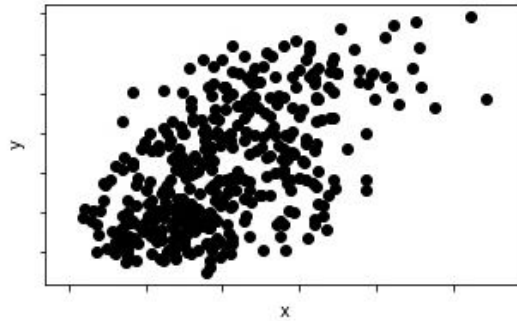
$$\begin{aligned}\Theta^T &= (\Theta_0, \Theta_1) \\ \mathbf{x}'^T &= (1, \mathbf{x}_1)\end{aligned}$$

so that the straight line may be written as

$$\hat{\mathbf{y}} = \Theta^T \cdot \mathbf{x}'$$

Because the size of Θ^T and \mathbf{x} must match

- we augmented \mathbf{x} with a "constant" feature 1
 - that corresponds to the intercept



$\langle \mathbf{X}, \mathbf{y} \rangle$

Training examples

$\mathbf{X} : m \times n$
 $\mathbf{y} : n \times 1$

Model

$$h_{\Theta}(\mathbf{x}) = \Theta_0 * \mathbf{x}_0 + \Theta_1 * \mathbf{x}_1$$

$$= \Theta^T \cdot \mathbf{x}$$

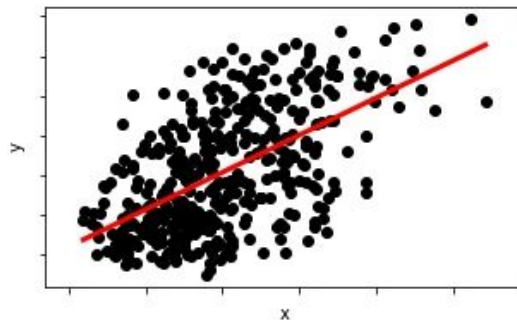
Fitting, training

$\Theta = [\Theta_0, \Theta_1] = [\text{intercept, slope}]$

$\Theta : (n + 1) \times 1$

$\mathbf{x}^{(i)} = [1, \mathbf{x}_1^{(i)}, \dots, \mathbf{x}_n] : (n + 1) \times 1$

Fit 



The real power of Linear Regression can be seen when there is more than 1 non-constant feature.

- Predict Price given features Size, Number of bedrooms, Number of bathrooms, Proximity to transportation
- Θ_j tells us how much each unit increase in feature \mathbf{x}_j affects Price.

The prediction \mathbf{y} is linear in each feature \mathbf{x}_j , hence the name *linear* regression

Anyone recognize this expression: $\Theta^T \cdot \mathbf{x}$?

It's our friend the dot product, as promised in the introductory lecture.

Watch out, this will be a regularly recurring character in our series.

Linear Regression in matrix form

We will typically augment \mathbf{x} with the leading "constant feature 1" to capture the intercept.

$$\begin{aligned}\Theta^T &= (\Theta_0, \Theta_1, \dots, \Theta_n) \\ \mathbf{x}'^T &= (1, \mathbf{x}_1, \dots, \mathbf{x}_n)\end{aligned}$$

We do this for each example in \mathbf{X} so that \mathbf{X} becomes

$$\mathbf{X}' = \begin{pmatrix} 1 & \mathbf{x}_1^{(1)} & \dots & \mathbf{x}_n^{(1)} \\ 1 & \mathbf{x}_1^{(2)} & \dots & \mathbf{x}_n^{(2)} \\ \vdots & \vdots & \dots & \vdots \\ 1 & \mathbf{x}_1^{(m)} & \dots & \mathbf{x}_n^{(m)} \end{pmatrix}$$

We sometimes refer to \mathbf{X} as the *design matrix*.

So we could simultaneously obtain our prediction for *all* training examples by the matrix product

$$\hat{\mathbf{y}} = \mathbf{X}'\Theta$$

Using matrix notation

- mimics an implementation using a language(such as numPy) with matrix arithmetic
- allows us to evaluate examples in parallel

Examples

Some examples

- Predict the Price of a stock given Earnings ($||\mathbf{x}|| = 1$)
- Predict the Price of a stock given Earnings, Dividend, and Sales ($||\mathbf{x}|| = 3$)

In [2]: `print("Done")`

Done