## Regression

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

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

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

ullet where the  $oldsymbol{y}$  are discrete values

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

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

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

A strong criticism of KNN is that  $\Theta$ , the parameters, comprised all m training examples

- large
- memorization versus generalization

The fact that y is continuous rather than discrete

ullet opens the possibility of a *numerical* relationship between features  ${f x}$  and labels  ${f y}$ .

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  ${\bf y}$  and  ${\bf x}$  to

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

That is: the predicted  $\hat{\mathbf{y}}$  is a linearly-weighted (with weights from vector  $\Theta$ ) 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  $\Theta_0$  and slope  $\Theta_1$ 

$$\hat{\mathbf{y}} = \Theta_0 + \Theta_1 * \mathbf{x}_1$$

In our example

- ullet we expect the Price to increase with Size  ${f x}_1$ 
  - ullet  $\Theta_1$  tells us how much each extra unit of Size increases the Price

Rather than writing the intercept  $\Theta_0$  as a separate term we can modify  ${f x}$  and  $\Theta$ 

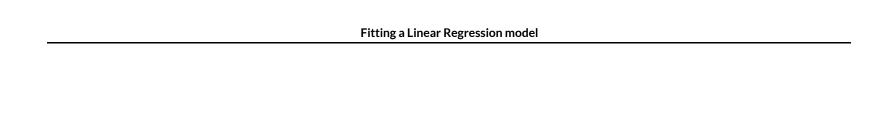
$$egin{array}{lll} \Theta^T &=& (\Theta_0,\Theta_1) \ \mathbf{x}'^T &=& (1,\mathbf{x}_1) \end{array}$$

so that the straight line may be written as

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

Because the size of  $\Theta^T$  and  ${f x}$  must match

- ullet we augmented  ${f x}$  with a "constant" feature 1
  - that corresponds to the intercept



The real power of Linear Regression can be seen when there is more than one nonconstant feature.

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

The prediction y is linear in each feature  $x_j$ , 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.

$$egin{array}{lll} \Theta^T &=& (\Theta_0,\Theta_1,\ldots,\Theta_n) \ \mathbf{x}'^T &=& (1,\mathbf{x}_1,\ldots,\mathbf{x}_n) \end{array}$$

We do this for each example in  ${f X}$  so that  ${f X}$  becomes

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

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

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

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

Done