Regression

Some slides thanks to Oznur Tastan

- We saw how to do regression with decision trees and k-NN so far. These are non-parametric methods.
- There are also parametric approaches, such as linear regression, where you assume a parametric form (e.g. a line) ahead of time and then search for best fitting line.
- In these slides you will see linear, multiple linear and polynomial regression.

- We have talked about regression problems before, as the problem of estimating the function f(x) between an independent variable x and a dependent variable y.

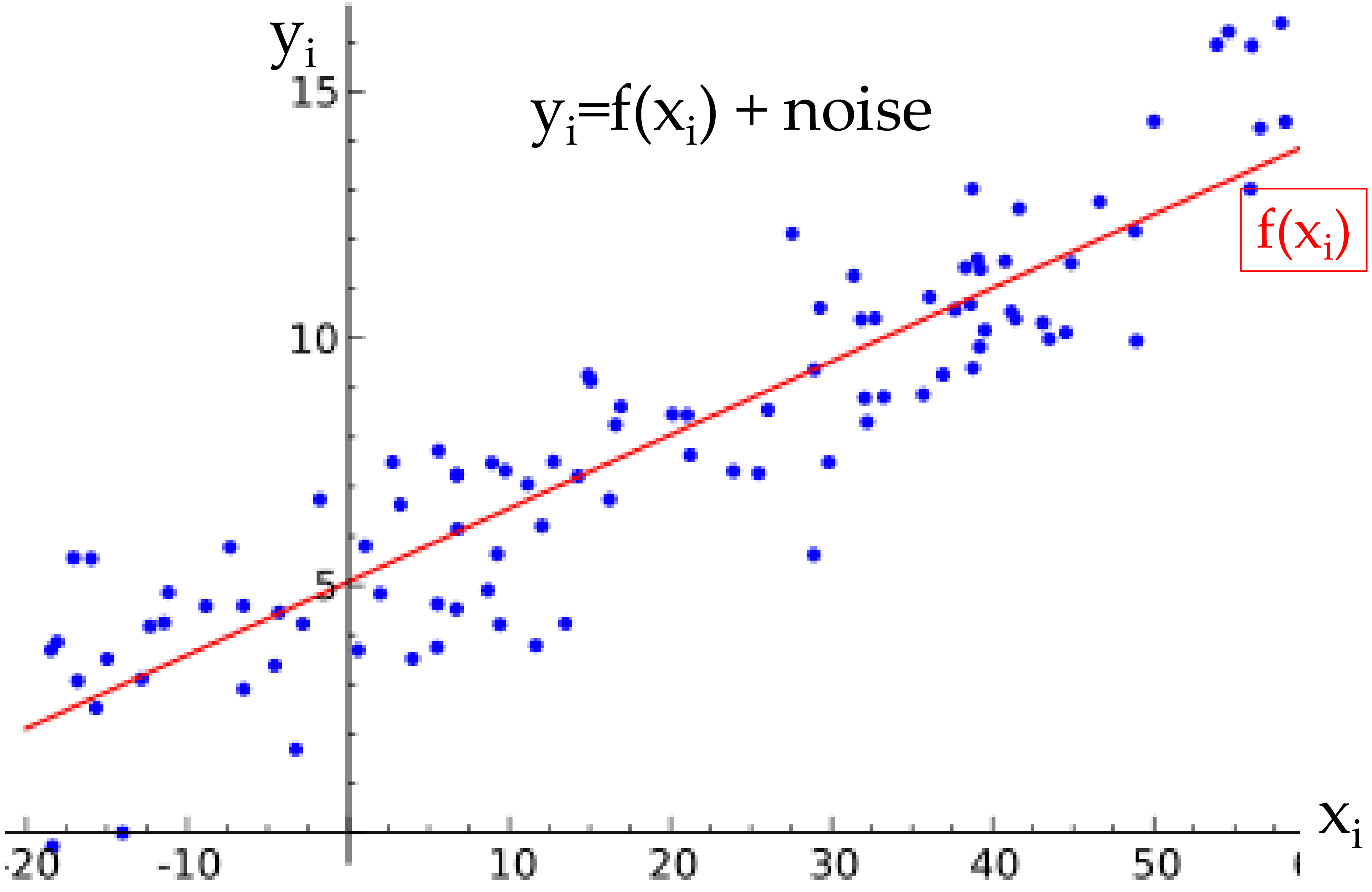
- We assume we have a dataset D={($\mathbf{x}_i$,y_i)} where

$$y_i = f(\mathbf{x}_i) + \varepsilon_i$$

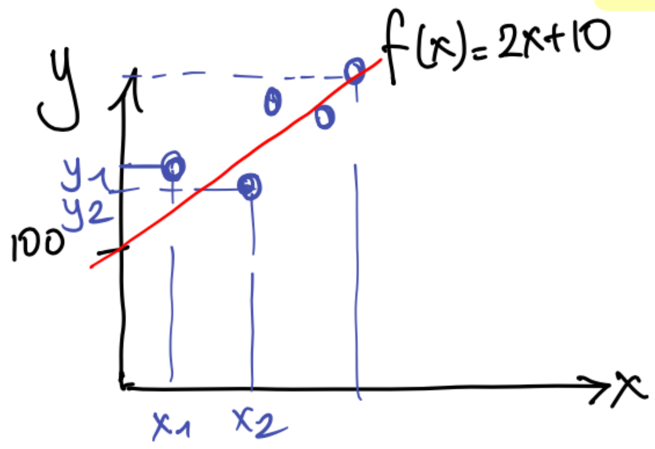
from which to learn f().

- We assume that ε_i is zero-mean noise.

Example Data



Review – Summary from Past Weeks



Regression: $\rightarrow$ underlying function

Assume: $y_i = f(x_i) + \varepsilon_i$ ε_i (zero-mean noise)

Problem: Find a model that outputs $y(x)$ which is an approximation to $f(x)$.

$\rightarrow y(x) = \hat{f}(x)$

$\rightarrow$ approximation $\hat{f}(x)$

$x \rightarrow \boxed{ML} \rightarrow y(x)$

$\rightarrow$ loss $= (y_i - y(x_i))^2$: squared loss

$\rightarrow \underset{\text{MSE}}{\text{cost}} = \frac{1}{N} \sum_i (y_i - y(x_i))^2)$

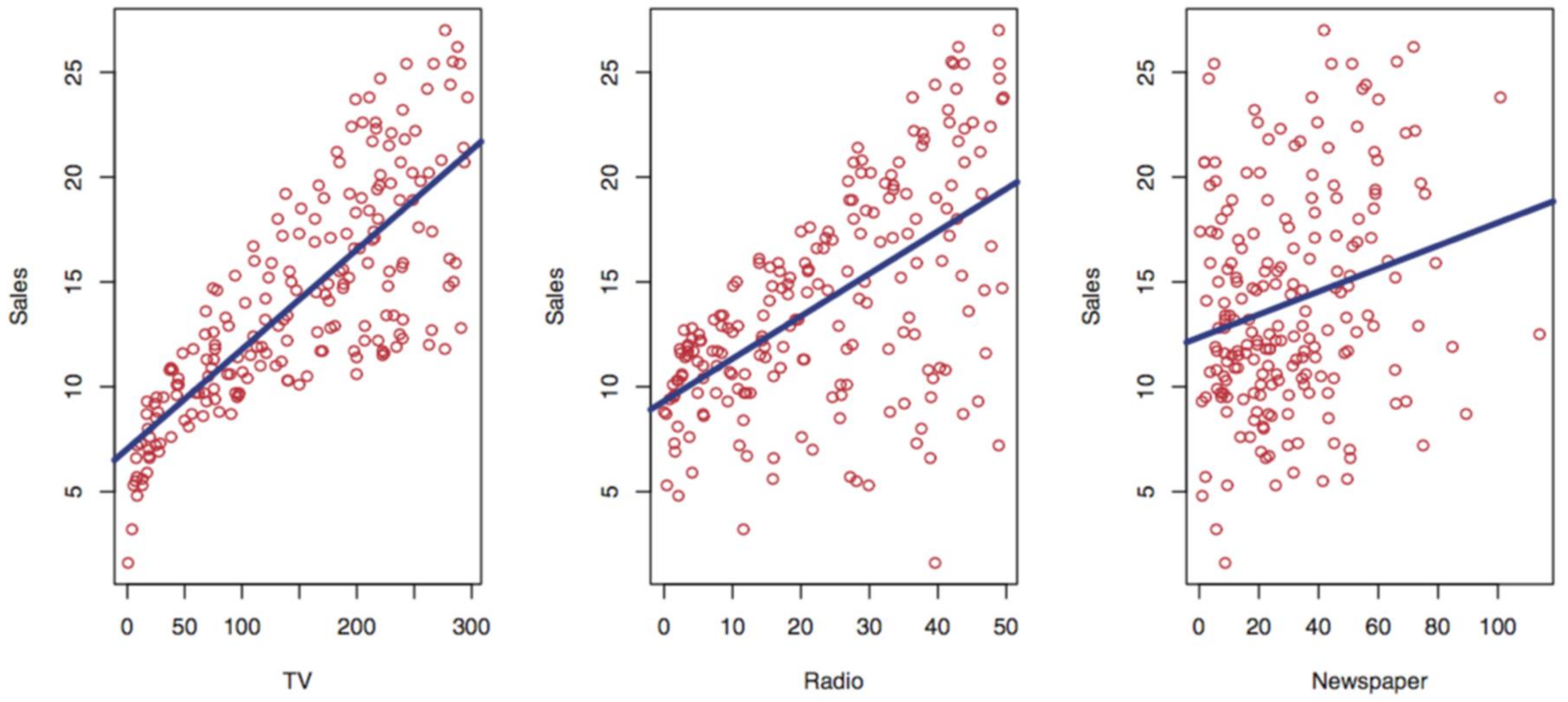
$\rightarrow$ MSE = Mean Squared Error

We will look at these in a bit more detail now…

Linear Regression:

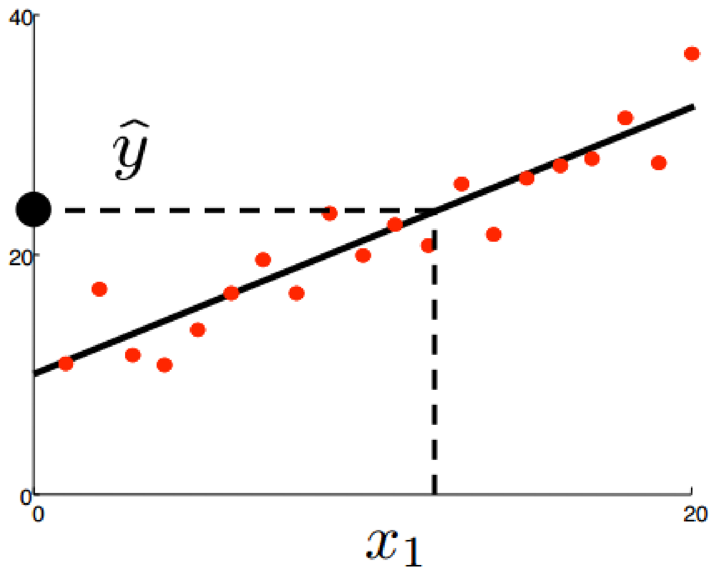
- We assume here that y_i are linearly dependent on x_i
- We find the parameters w_0 and w_1 using different approaches:

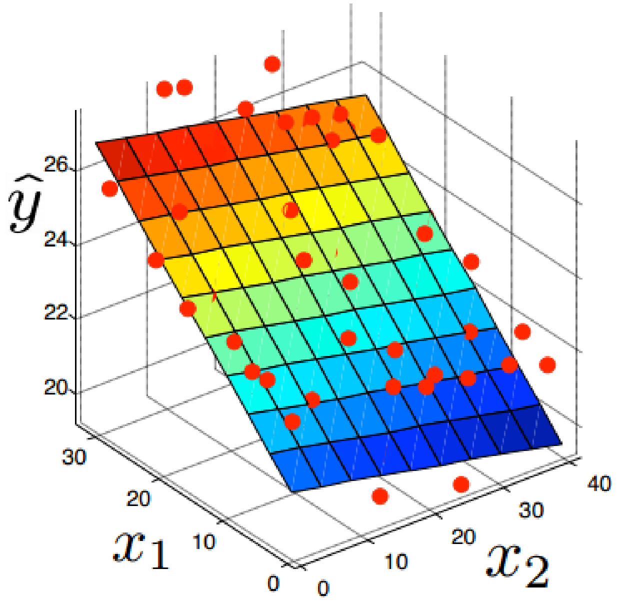
$$y = w_0 + w_1 b$$



What if there are multiple features?

- **Multiple Linear regression**





Prediction

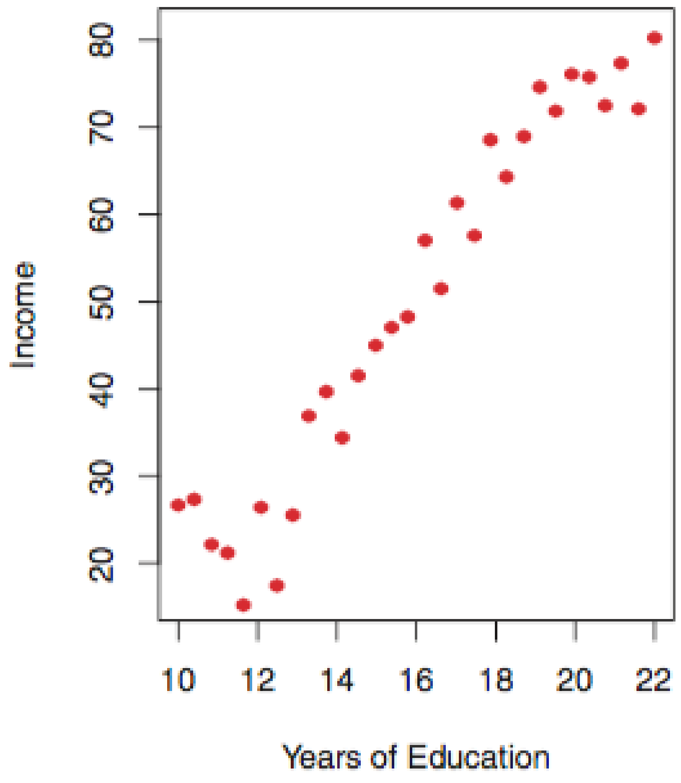
$$\hat{y} = w_0 + w_1 x_1$$

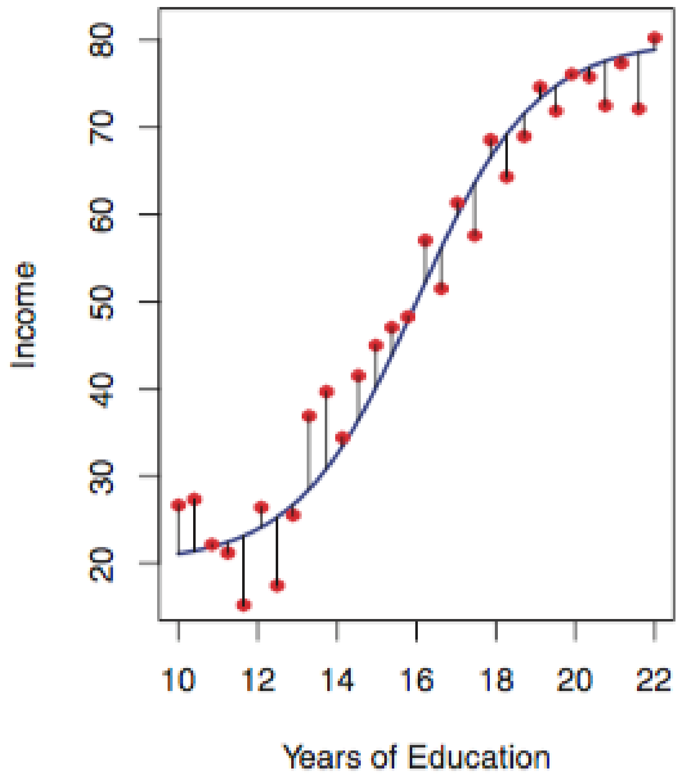
Prediction

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2$$

What if the relationship is not linear?

- **Polynomial regression**
  - If we have some reason to believe that the relationship between x_i and y_i is not linear, *or we want to allow for this possibility*, we can use polynomial regression.





Polynomial regression

- We can add power terms of x (as new features) and model as multiple linear regression:

$$y(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

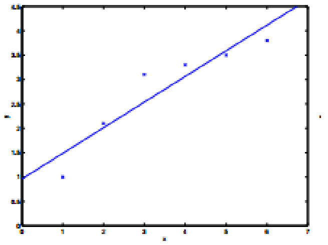
- Note the only difference that the new features is powers of the only explanatory variable x.

- Here M is the power

SUMMARY

Linear regression:

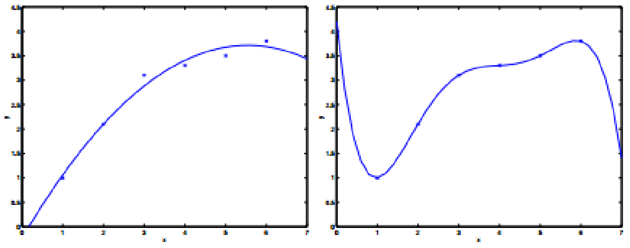
$$y(x) = w_0 + w_1 x$$



Polinomial regression (M is the degree of the polynom)

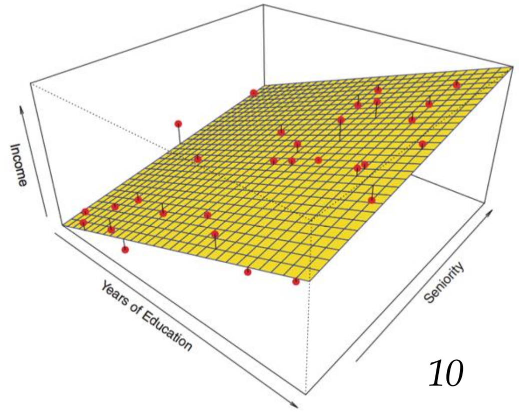
:

$$y(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$



Multiple linear regression (Data: {(**x**=[x$_1$ x$_2$ ... x$_k$],y)}):

$$y(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_k x_k$$



Extending Application of Linear Regression

- The inputs **X** for linear regression can be:
  - Original quantitative inputs
  - Transformation of quantitative inputs, e.g. log, exp, square root, square, etc.
  - Polynomial transformation
    - example: $y = w_0 + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3$
  - Interactions between variables
    - example: $x_3 = x_1 \cdot x_2$
  - Dummy coding of categorical inputs

- This allows use of linear regression techniques to fit much more complicated non-linear datasets.

Adding new features

- In the Credit Applicant problem (given the information about applicant, decide whether a Good credit candidate or not), you may be given salary, debt, and other info.
- In general, depending on the ML approach used, you may want to add new features based on existing ones:
    - F_{new} = debt / salary
    - F_{new} = #defaulted credits / age
    - F_{new} = #defaulted-credits^2
    - Powers of a single independent variable x in polynomial regression.

Feature selection and Extraction

- It is also possible to **select features** (and eliminate others) or **extract new features from existing ones** (and deleting existing ones)
  - **You can remove one feature if it is highly correlated with another one**
  - **You can leave the most statistically correlated features with the output label**
    - But maybe one feature is not very much correlated but contributes. Think about each pixel in MNIST.
    - SelectKBest in Sci-Kit-Learn can be used with a suite of different statistical tests to select a specific number of features.
  - Tree-based approaches (random forest etc) also have methods to measure feature importance (will see later)

Reminder

- Too few features
  - Underfitting
  - Large intrinsic error
- Too many features (Note: since it says “too many”, it means more than necessary – not just “many”)
  - Overfitting
  - Low train set error

We would like to:

- **have most/all the relevant features** (often we will not know or be able to include all, but we should try)
  - know the domain
- **hopefully not many irrelevant ones** (often we will not know this either)
  - use ML approaches that are robust to many irrelevant features if we suspect to have possibly too many/irrelevant attributes.

Learning the Coefficients for Linear and Multiple Linear Regression

Measure of Error – Loss versus Cost

- We can measure the prediction loss in terms of squared error. Loss on one example:

$$\text{Loss}(y,\hat{y}) = (y-\hat{y})^2$$

- Loss on *n* training examples (called Cost):

$$J_n(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(\mathbf{x}_i;\mathbf{w})\right)^2$$

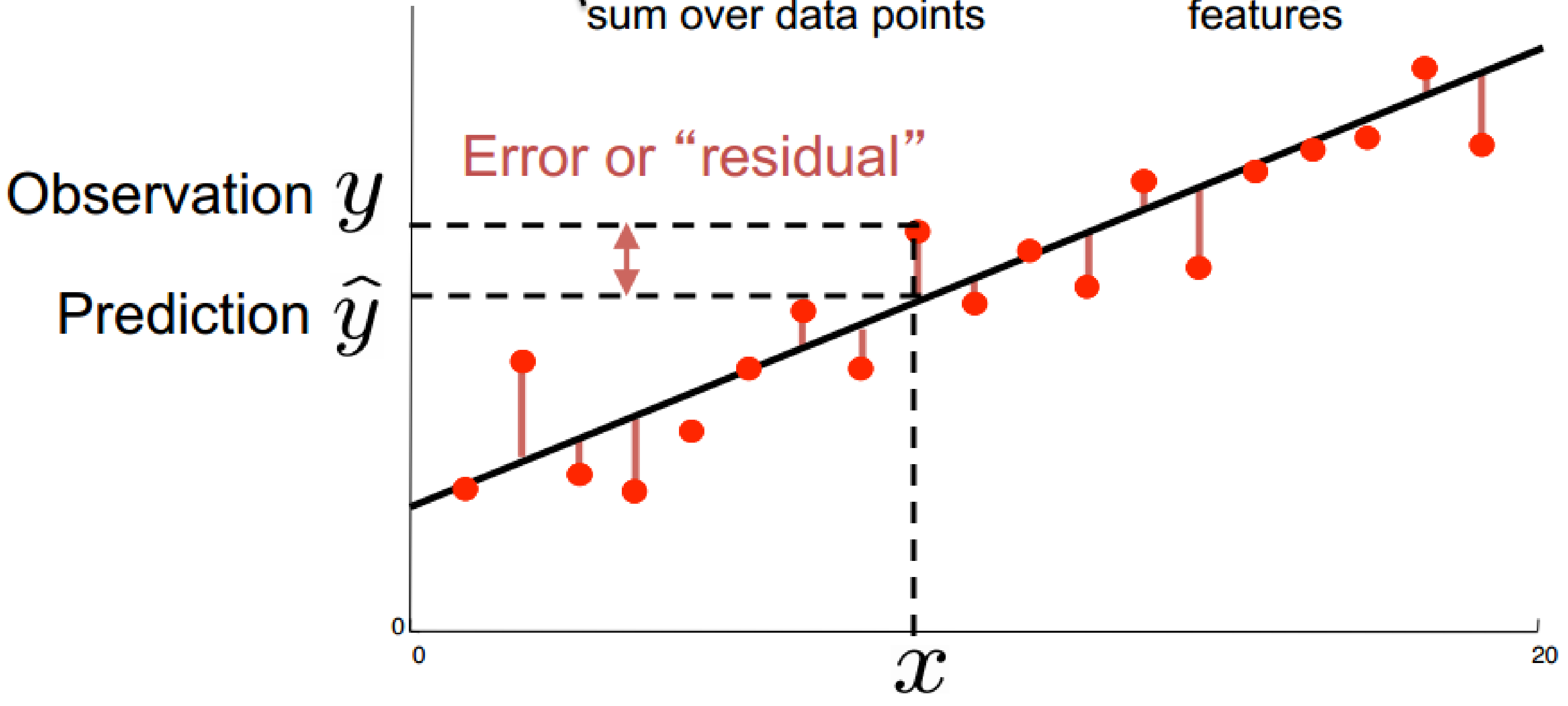
$$f:\mathcal{R}^d \rightarrow \mathcal{R} \qquad f(\mathbf{x};\mathbf{w}) = w_0 + w_1x_1 + \ldots w_dx_d$$

Ordinary Least Squares(OLS)

$$\text{total error} = \sum_i (y_i - \hat{y}_i)^2 = \sum_i \left(y_i - \sum_k w_k x_k^{(i)} \right)^2$$

sum over data points

features



Least Squares Linear Fit to Data

- Most popular estimation method is least squares:
  - Determine linear coefficients w_0, **w** that minimize sum of squared loss (SSL)
  - Use standard (multivariate) differential calculus:
    - differentiate total loss with respect to w_0, **w**
    - find zeros of each partial differential equation
    - solve for w_0, **w**

Minimize the Squared Loss

- Minimize the empirical squared loss:

$$\begin{aligned} J_n(\mathbf{w}) &= \frac{1}{n}\sum_{i=1}^{n}\big(y_i - f(\mathbf{x}_i;\mathbf{w})\big)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2 \quad \text{(1-dim)} \end{aligned}$$

Direct Minimization

- Minimize the empirical squared loss:

$$
\begin{aligned}
J_n(\mathbf{w}) &= \frac{1}{n}\sum_{i=1}^{n}\big(y_i - f(\mathbf{x}_i;\mathbf{w})\big)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2 \quad \text{(1-dim)}
\end{aligned}
$$

- To get the optimal parameter values, take partial derivatives and set to zero (requirement for minima/maxima)

$$
\begin{aligned}
\frac{\partial}{\partial w_1} J_n(\mathbf{w}) &= 0 \\
\frac{\partial}{\partial w_0} J_n(\mathbf{w}) &= 0
\end{aligned}
$$

Finding Optimal Parameters for Simple Linear Regression

$$
\begin{aligned}
\frac{\partial}{\partial w_1} J_n(\mathbf{w}) &= \frac{\partial}{\partial w_1} \frac{1}{n} \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i)^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial w_1} (y_i - w_0 - w_1 x_i)^2 \\
&= \frac{2}{n} \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i) \frac{\partial}{\partial w_1} (y_i - w_0 - w_1 x_i) \\
&= \frac{2}{n} \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i)(-x_i) = 0 \\
\frac{\partial}{\partial w_0} J_n(\mathbf{w}) &= \frac{2}{n} \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i)(-1) = 0
\end{aligned}
$$

Solving for w_0

$$\frac{\partial}{\partial w_0} J_n(\mathbf{w}) = \frac{2}{n} \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i)(-1) = 0$$

Complete the differentiation

Regression in Matrix Notation

Define:

- the data matrix **X** (row i contains the extended input vector for data sample $\mathbf{x}_i$ which is d+1 dimensional);
- the target vector **y** (row i contains the target y_i for the data sample $\mathbf{x}_i$ – it is N dimensional where N is the number of data points)
- weight vector β is d+1-dimensional

Then we have: $\mathbf{y} = \mathbf{X}\beta + \varepsilon$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} \ldots x_{1d} \\ 1 & x_{21} & x_{22} \ldots x_{2d} \\ \vdots & & \\ 1 & x_{N1} & x_{N2} \ldots x_{Nd} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \ldots \\ \beta_d \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \ldots \\ \mathbf{x}_N^T \end{bmatrix}$$

Minimize ||y−XB||

Matrix Form

By setting the derivatives of $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2/n$ to zero, we get the same optimality conditions as before, now expressed in a matrix form

which gives

pseudo-inverse

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- The solution is a linear function of the outputs y

- Pseudo-inverse is used when X may not be invertible (not square, or singular)

Numerical Solution

- Using the analytical form may not be feasible when the number of dimensions is large as we need to take matrix inverse.
- We can use other methods such as gradient descent approach

- Demo at:
- https://colab.research.google.com/drive/1lKt8Kb3IdOiCtlzZyw6UG7hqMBAShgBW?usp=sharing

Regression terminology

- For your reference, the following terms are used interchangeably in machine learning and statistics community

Y	X
Dependent variable	Independent variable
Explained variable	Explanatory variable
Target variable	Control variable
Labels	Features

Evaluation

- We can use MSE (Mean Squared Error) or better, use RMSE (RootMeanSquared Error) to compare model performances on the same data set.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i}(y_i - f_i)^2}$$

- Both MSE and RMSE dependon the scale of the original data, so **comparison across different data** sets is not meaningful
  - But we can use them for model selection using the same validation set and features

- There is also commonly mentioned R^2 measure
  - "How much better is our model than a baseline model predicting average y?"
  - But do not use RSME for model selection since R^2 increases with increasing model complexity

- Lets calculate the **average y** of the data:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

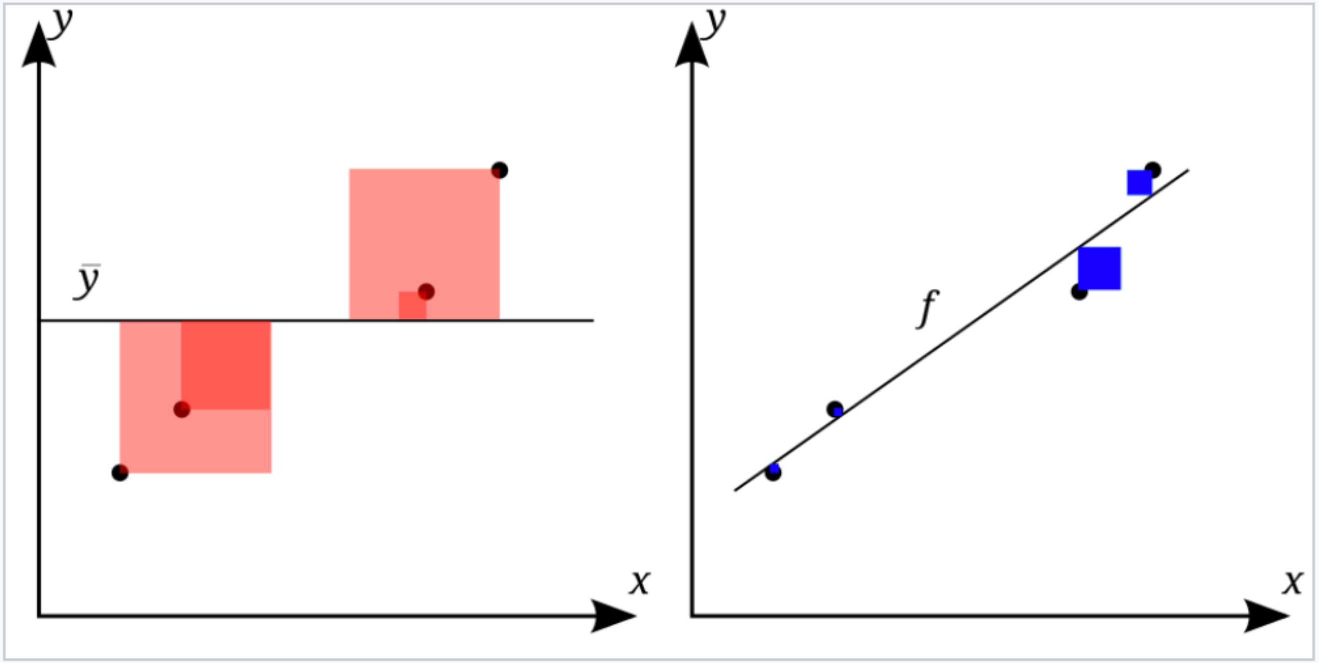
- Calculate the **sum of squares of residuals**
  - total sum of differences between prediction and actual targets

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

- Calculate the **total sum of squares**

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2$$

- Coefficient of determination is $R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$



$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

The better the linear regression (on the right) fits the data in comparison to the simple average (on the left graph), the closer the value of R^2 is to 1. The areas of the blue squares represent the squared residuals with respect to the linear regression. The areas of the red squares represent the squared residuals with respect to the average value.

Image credit: Wikipedia

- **If the predicted values exactly match the observed values:**
  - SS$_{res}$ is 0 and R^2 is 1

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

- **If the model always predicts *y-average*:**
  - $R^2 = 0$.

- In general, R^2 ranges between 0 and 1 (occasionally even negative)