Supervised Learning

Dan Lizotte

2017-09-19

Supervised Learning Framework (HTF 2, JWHT 2)

Training experience: a set of ***labeled examples*** of the form

$$\langle x_1, x_2, \ldots x_p, y \rangle,$$

where x_j are ***feature values*** and y is the ***output***

- Task: ***Given a new*** $x_1, x_2, \ldots x_p$***, predict*** y

What to learn: A ***function*** $f : \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_p \to \mathcal{Y}$, which maps the features into the output domain

- Goal: Make accurate ***future predictions*** (on unseen data)
- Plan: Learn to make accurate predictions on the training data

Wisconsin Breast Cancer Prognostic Data

Cell samples were taken from tumors in breast cancer patients before surgery and imaged; tumors were excised; patients were followed to determine whether or not the cancer recurred, and how long until recurrence or disease free.

Wisconsin data (continued)

- 198 instances, 32 features for prediction
- Outcome (R=recurrence, N=non-recurrence)
- Time (until recurrence, for R, time healthy, for N).

Radius.Mean	Texture.Mean	Perimeter.Mean	...	Outcome	Time
18.02	27.60	117.50		N	31
17.99	10.38	122.80		N	61
21.37	17.44	137.50		N	116
11.42	20.38	77.58		N	123
20.29	14.34	135.10		R	27
12.75	15.29	84.60		R	77
...	...	...	...	...	...

Terminology

Radius.Mean	Texture.Mean	Perimeter.Mean	...	Outcome	Time
18.02	27.60	117.50		N	31
17.99	10.38	122.80		N	61
21.37	17.44	137.50		N	116
11.42	20.38	77.58		N	123

Radius.Mean	Texture.Mean	Perimeter.Mean	...	Outcome	Time
20.29	14.34	135.10		R	27
12.75	15.29	84.60		R	77
...	...	...		...	...

- Columns are called *input variables* or ***features*** or *attributes*
- The outcome and time (which we are trying to predict) are called ***labels*** or *output variables* or *targets*
- A row in the table is called ***training example*** or *instance*
- The whole table is called ***(training) data set***.

Prediction problems

Radius.Mean	Texture.Mean	Perimeter.Mean	...	Outcome	Time
18.02	27.60	117.50		N	31
17.99	10.38	122.80		N	61
21.37	17.44	137.50		N	116
11.42	20.38	77.58		N	123
20.29	14.34	135.10		R	27
12.75	15.29	84.60		R	77
...	...	...		...	...

- The problem of predicting the recurrence is called ***(binary) classification***
- The problem of predicting the time is called ***regression***

More formally

- The ith training example has the form: $\langle x_{1,i}, \ldots x_{p,i}, y_i \rangle$ where p is the number of features (32 in our case).
- Notation $\mathbf{x}_i$ denotes a **column** vector with elements $x_{1,i}, \ldots x_{p,i}$.
- The training set D consists of n training examples
- We denote the $n \times p$ matrix of features by X and the size-n column vector of outputs from the data set by $\mathbf{y}$.
- In statistics, X is called the data matrix or the *design matrix.*
- $\mathcal{X}$ denotes space of input values
- $\mathcal{Y}$ denotes space of output values

Supervised learning problem

- Given a data set $D \subset (\mathcal{X} \times \mathcal{Y})^n$, find a function:

$$h : \mathcal{X} \to \mathcal{Y}$$

  such that $h(\mathbf{x})$ is a *"good predictor"* for the value of y.
- h is called a *predictive model* or *hypothesis*
- Problems are categorized by the type of output domain

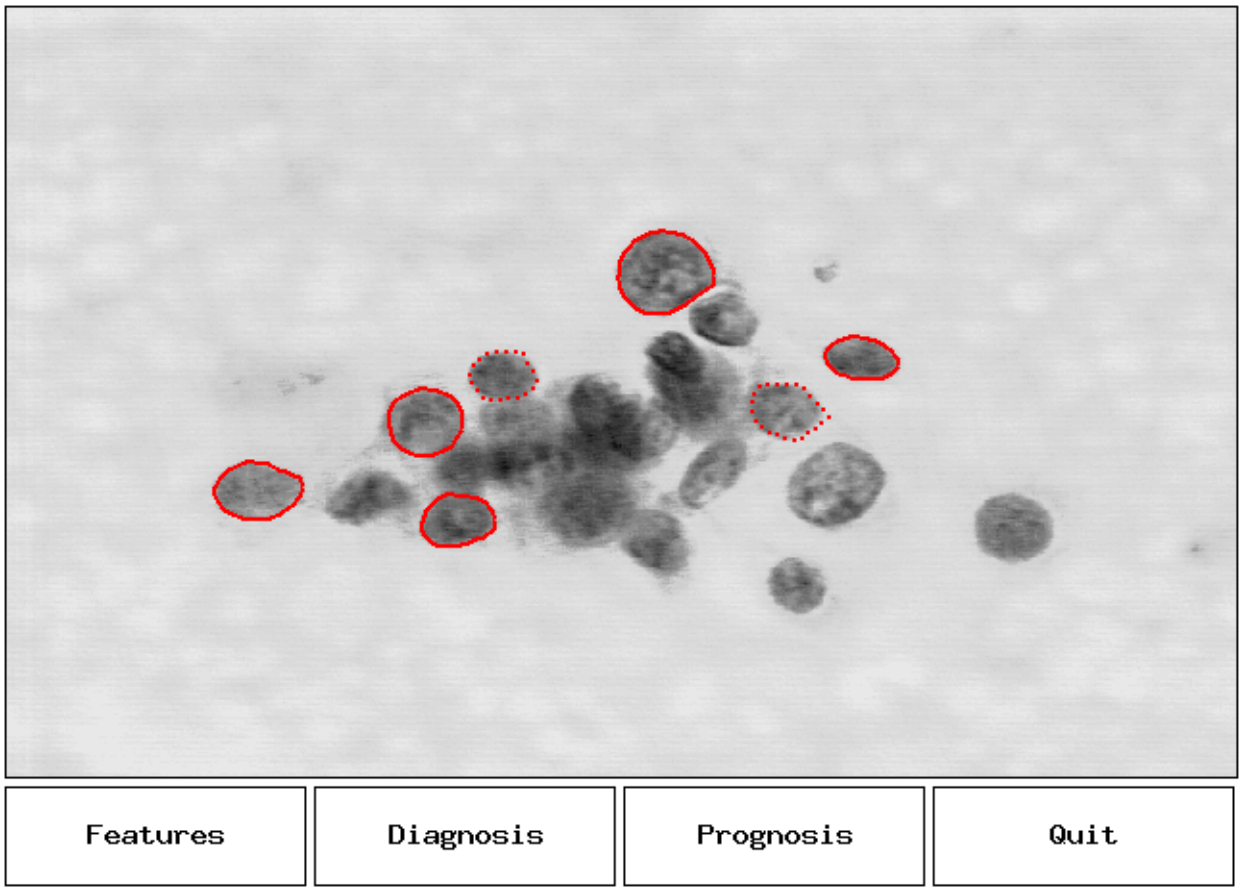


Figure 1: image

- If $\mathcal{Y} = \mathbb{R}$, this problem is called *regression*
- If $\mathcal{Y}$ is a finite discrete set, the problem is called *classification*
- If $\mathcal{Y}$ has 2 elements, the problem is called *binary classification*

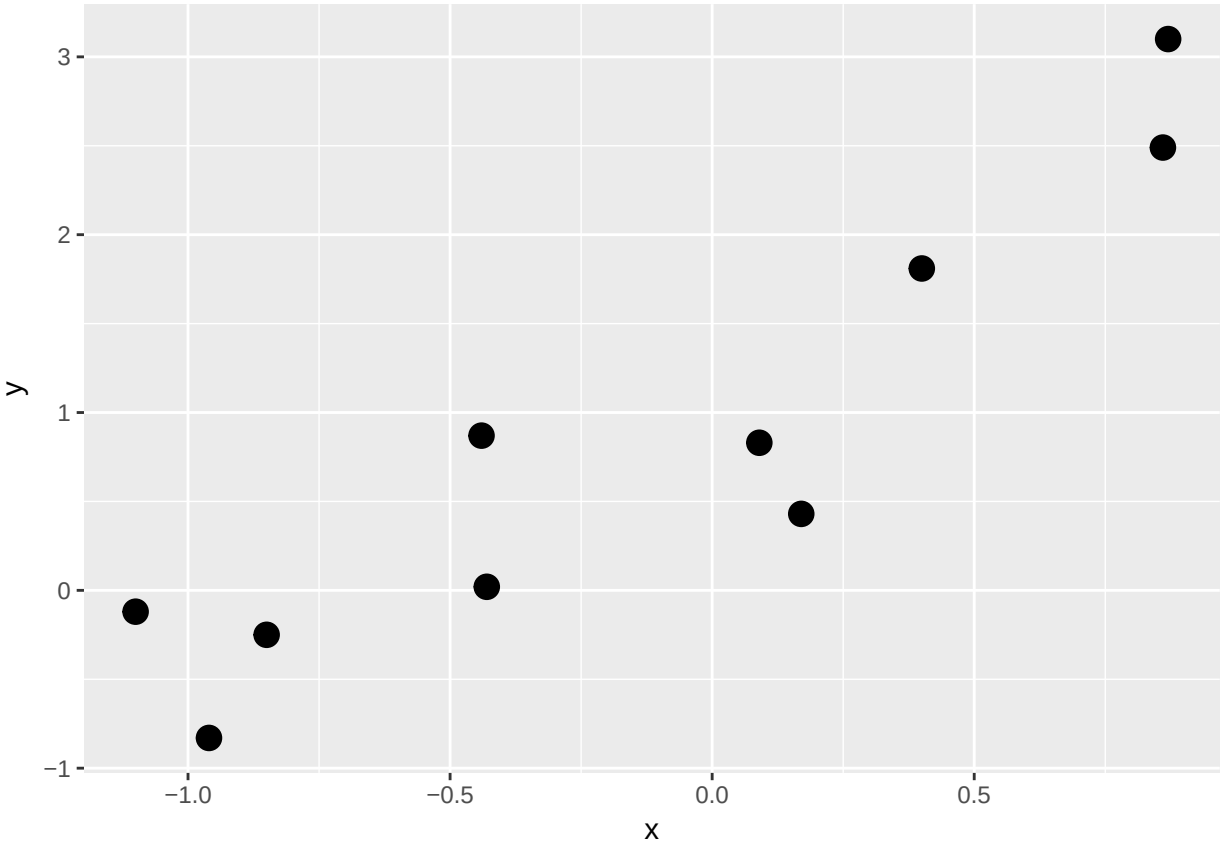
Steps to solving a supervised learning problem

1. Decide what the input-output pairs are.
2. Decide how to encode inputs and outputs.

   This defines the input space $\mathcal{X}$, and the output space $\mathcal{Y}$.

   (We will discuss this in detail later)
3. Choose model space/hypothesis class $\mathcal{H}$.
4. ...

Example: Choosing a model space



Linear hypothesis (HTF 3, JWHT 3)

- Suppose y was a linear function of $\mathbf{x}$:

$$h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots$$

- w_i are called *parameters* or *weights* (often β_i in stats books)
- Typically include an attribute $x_0 = 1$ (also called *bias term* or *intercept term*) so that the number of weights is $p + 1$. We then write:

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{i=0}^{p} w_i x_i = \mathbf{x}^{\mathsf{T}}\mathbf{w}$$

  where $\mathbf{w}$ and $\mathbf{x}$ are column vectors of size $p + 1$.
- The design matrix X is now n by $p + 1$.

Example: Design matrix with bias term

x0	x1	y
1	0.86	2.49
1	0.09	0.83
1	-0.85	-0.25
1	0.87	3.10
1	-0.44	0.87
1	-0.43	0.02
1	-1.10	-0.12
1	0.40	1.81
1	-0.96	-0.83
1	0.17	0.43

Models will be of the form

$$h_{\mathbf{w}}(\mathbf{x}) = x_0 w_0 + x_1 w_1 = w_0 + x_1 w_1$$

How should we pick $\mathbf{w}$*?*

Error minimization

- Intuitively, $\mathbf{w}$ should make the predictions of $h_{\mathbf{w}}$ close to the true values y_i on on the training data
- Define an *error function* or *cost function* to measure how much our prediction differs from the "true" answer on the training data
- Pick $\mathbf{w}$ such that the error function is minimized
- Hopefully, new examples are somehow "similar" to the training examples, and will also have small error.

How should we choose the error function?

Least mean squares (LMS)

- Main idea: try to make $h_{\mathbf{w}}(\mathbf{x})$ close to y on the examples in the training set
- We define a *sum-of-squares* error function

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$$

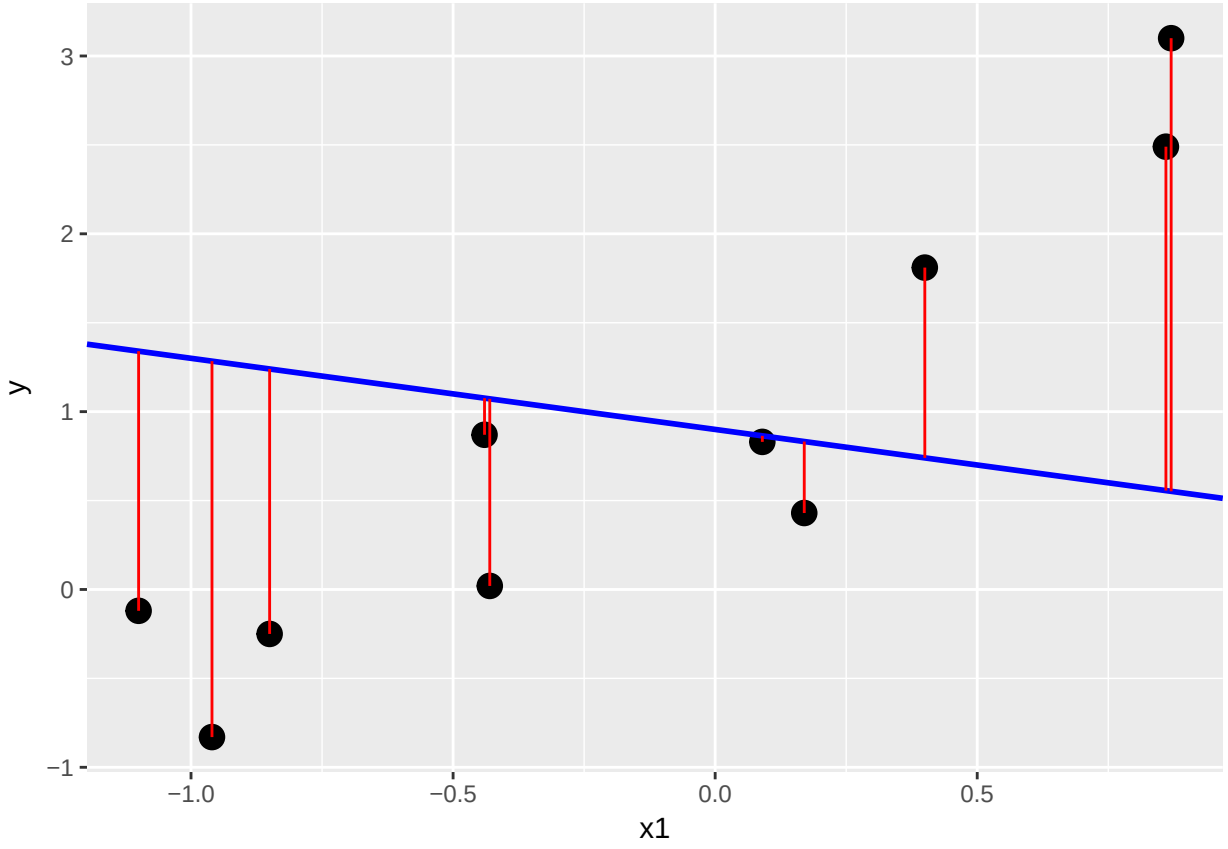
(the 1/2 is just for convenience)

- We will choose $\mathbf{w}$ such as to minimize $J(\mathbf{w})$
- One way to do it: compute $\mathbf{w}$ such that:

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = 0, \ \forall j = 0 \dots p$$

Example: $w_0 = 0.9, w_1 = -0.4$

```
## SSE: 21.510
```

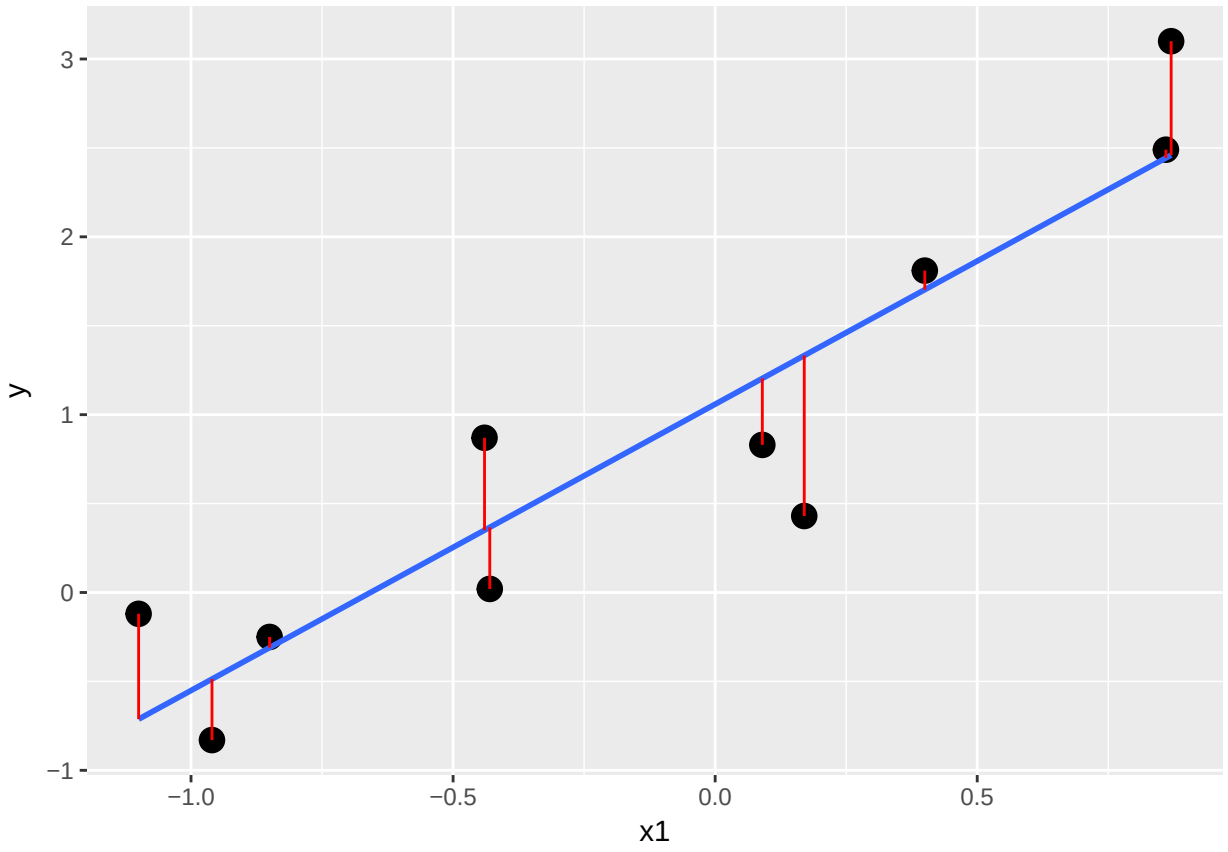


OLS Fit to Example Data

```
mod <- lm(y ~ x1, data=exb); print(mod$coefficients)
```

```
## (Intercept)          x1
##    1.058813    1.610168
```

```
## SSE: 2.240
```



Solving a supervised learning problem: optimisation-based approach

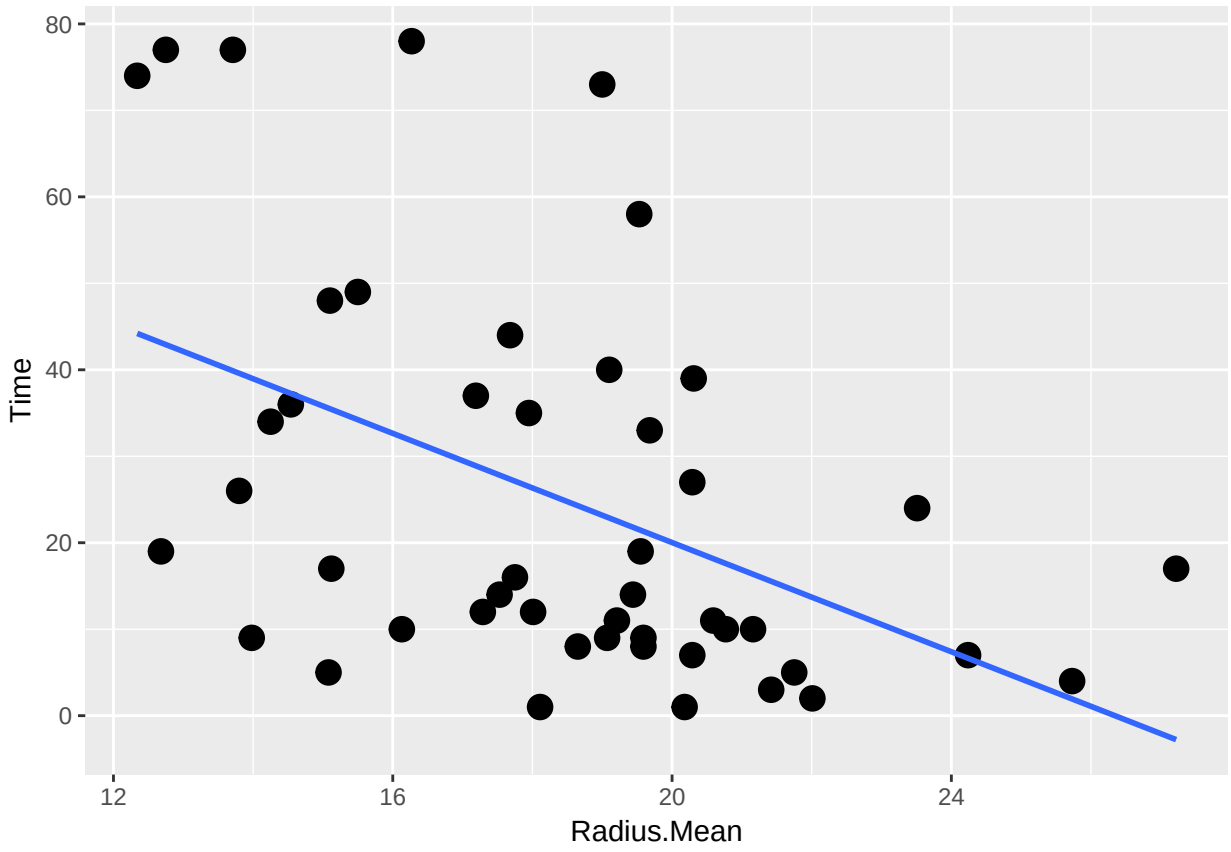
1. Decide what the input-output pairs are.
2. Decide how to encode inputs and outputs.

   This defines the input space $\mathcal{X}$, and the output space $\mathcal{Y}$.
3. Choose a class of models/hypotheses $\mathcal{H}$.
4. Choose an error function (cost function) to define the best model in the class
5. Choose an algorithm for searching efficiently through the space of models to find the best.

Recurrence Time from Tumor Radius

```
mod <- lm(Time ~ Radius.Mean, data=bc %>% filter(Outcome == 'R')); print(mod$coefficients)
```

```
## (Intercept) Radius.Mean
##   83.161238   -3.156896
```



Notation reminder

- Consider a function $J(u_1, u_2, \ldots, u_p) : \mathbb{R}^p \mapsto \mathbb{R}$ (for us, this will usually be an error function)
- The *gradient* $\nabla J(u_1, u_2, \ldots, u_p) : \mathbb{R}^p \mapsto \mathbb{R}^p$ is a function which outputs a vector containing the partial derivatives.
  That is:
$$\nabla J = \left\langle \frac{\partial}{\partial u_1} J, \frac{\partial}{\partial u_2} J, \ldots, \frac{\partial}{\partial u_p} J \right\rangle$$
- If J is differentiable and convex, we can find the global minimum of J by solving $\nabla J = \mathbf{0}$.
- The partial derivative is the derivative along the u_i axis, keeping all other variables fixed.

The Least Squares Solution (HTF 2.6, 3.2, JWHT 3.1)

- Recalling some multivariate calculus:

$$\begin{aligned}
\nabla_{\mathbf{w}} J &= \nabla_{\mathbf{w}} (X\mathbf{w} - \mathbf{y})^\mathsf{T} (X\mathbf{w} - \mathbf{y}) \\
&= \nabla_{\mathbf{w}} (\mathbf{w}^\mathsf{T} X^\mathsf{T} - \mathbf{y}^\mathsf{T})(X\mathbf{w} - \mathbf{y}) \\
&= \nabla_{\mathbf{w}} (\mathbf{w}^\mathsf{T} X^\mathsf{T} X\mathbf{w} - \mathbf{y}^\mathsf{T} X\mathbf{w} - \mathbf{w}^\mathsf{T} X^\mathsf{T}\mathbf{y} + \mathbf{y}^\mathsf{T}\mathbf{y}) \\
&= \nabla_{\mathbf{w}} (\mathbf{w}^\mathsf{T} X^\mathsf{T} X\mathbf{w} - 2\mathbf{y}^\mathsf{T} X\mathbf{w} + \mathbf{y}^\mathsf{T}\mathbf{y}) \\
&= 2X^\mathsf{T} X\mathbf{w} - 2X^\mathsf{T}\mathbf{y}
\end{aligned}$$

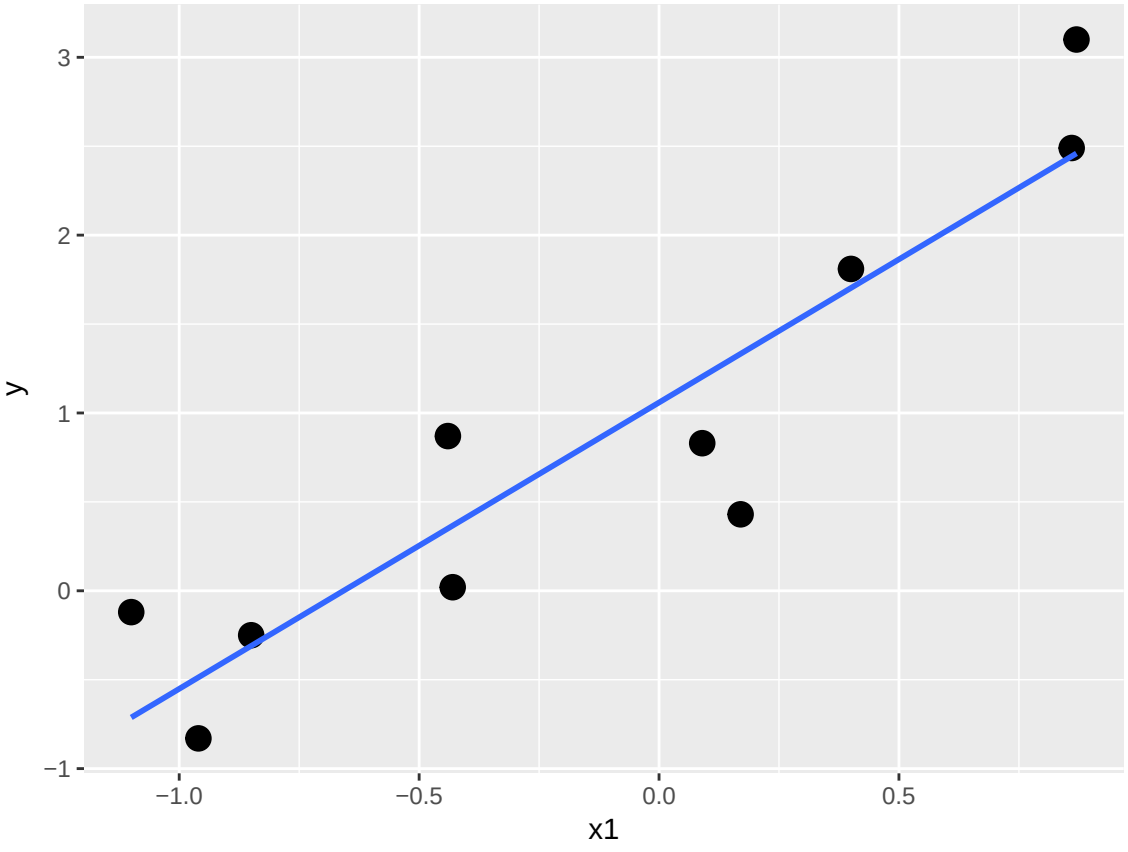
- Setting gradient equal to zero:

$$\begin{aligned} 2X^\mathsf{T}X\mathbf{w} - 2X^\mathsf{T}\mathbf{y} &= 0 \\ \Rightarrow X^\mathsf{T}X\mathbf{w} &= X^\mathsf{T}\mathbf{y} \\ \Rightarrow \mathbf{w} = (X^\mathsf{T}X)^{-1}X^\mathsf{T}\mathbf{y} & \end{aligned}$$

- The inverse exists if the columns of X are linearly independent.

Example of linear regression

x0	x1	y
1	0.86	2.49
1	0.09	0.83
1	-0.85	-0.25
1	0.87	3.10
1	-0.44	0.87
1	-0.43	0.02
1	-1.10	-0.12
1	0.40	1.81
1	-0.96	-0.83
1	0.17	0.43



$h_\mathbf{w}(\mathbf{x}) = 1.06 + 1.61x_1$

Data matrices

$$X = \begin{bmatrix} 1 & 0.86 \\ 1 & 0.09 \\ 1 & -0.85 \\ 1 & 0.87 \\ 1 & -0.44 \\ 1 & -0.43 \\ 1 & -1.10 \\ 1 & 0.40 \\ 1 & -0.96 \\ 1 & 0.17 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{bmatrix}$$

$X^\mathsf{T}X$

$$X^\mathsf{T}X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \end{bmatrix} \times \begin{bmatrix} 1 & 0.86 \\ 1 & 0.09 \\ 1 & -0.85 \\ 1 & 0.87 \\ 1 & -0.44 \\ 1 & -0.43 \\ 1 & -1.10 \\ 1 & 0.40 \\ 1 & -0.96 \\ 1 & 0.17 \end{bmatrix}$$

$$= \begin{bmatrix} 10 & -1.39 \\ -1.39 & 4.95 \end{bmatrix}$$

$X^\mathsf{T}\mathbf{y}$

$$X^\mathsf{T}\mathbf{y} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \end{bmatrix} \times \begin{bmatrix} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{bmatrix}$$

$$= \begin{bmatrix} 8.34 \\ 6.49 \end{bmatrix}$$

Solving for w

$$\mathbf{w} = (X^\mathsf{T}X)^{-1}X^\mathsf{T}\mathbf{y} = \begin{bmatrix} 10 & -1.39 \\ -1.39 & 4.95 \end{bmatrix}^{-1} \begin{bmatrix} 8.34 \\ 6.49 \end{bmatrix} = \begin{bmatrix} 1.06 \\ 1.61 \end{bmatrix}$$

So the best fit line is $y = 1.06 + 1.61x$.

Linear regression summary

- The optimal solution (minimizing sum-squared-error) can be computed in polynomial time in the size of the data set.
- The solution is $\mathbf{w} = (X^\mathsf{T}X)^{-1}X^\mathsf{T}\mathbf{y}$, where X is the data matrix augmented with a column of ones, and $\mathbf{y}$ is the column vector of target outputs.
- A very rare case in which an analytical, exact solution is possible

Is linear regression enough?

- Linear regression should be the **first thing** you try for real-valued outputs!
- ... but it is sometimes not expressive enough.
- Two possible solutions:
  1. Explicitly transform the data, i.e. create additional features
     - Add cross-terms, higher-order terms
     - More generally, apply a transformation of the inputs from $\mathcal{X}$ to some other space $\mathcal{X}'$, then do linear regression in the transformed space
  2. Use a different model space/hypothesis class
- Idea (1) and idea (2) are two views of the strategy. Today we focus on the first approach

Polynomial fits (HTF 2.6, JWHT 7.1)

- Suppose we want to fit a higher-degree polynomial to the data.
  (E.g., $y = w_0 + w_1 x_1 + w_2 x_1^2$.)
- Suppose for now that there is a single input variable $x_{i,1}$ per training sample.
- How do we do it?

Answer: Polynomial regression

- Given data: $(x_{1,1}, y_1), (x_{1,2}, y_2), \ldots, (x_{1,n}, y_n)$.
- Suppose we want a degree-d polynomial fit.
- Let $\mathbf{y}$ be as before and let

$$X = \begin{bmatrix} 1 & x_{1,1} & x_{1,1}^2 & \ldots & x_{1,1}^d \\ 1 & x_{1,2} & x_{1,2}^2 & \ldots & x_{1,2}^d \\ \vdots & & \vdots & \vdots & \vdots \\ 1 & x_{1,n} & x_{1,n}^2 & \ldots & x_{1,n}^d \end{bmatrix}$$

- We are **making up features** to add to our design matrix
- Solve the linear regression $X\mathbf{w} \approx \mathbf{y}$.

Example of quadratic regression: Data matrices

$$X = \begin{bmatrix} 1 & 0.86 & 0.75 \\ 1 & 0.09 & 0.01 \\ 1 & -0.85 & 0.73 \\ 1 & 0.87 & 0.76 \\ 1 & -0.44 & 0.19 \\ 1 & -0.43 & 0.18 \\ 1 & -1.10 & 1.22 \\ 1 & 0.40 & 0.16 \\ 1 & -0.96 & 0.93 \\ 1 & 0.17 & 0.03 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{bmatrix}$$

$X^\mathsf{T}X$

$$X^\mathsf{T}X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\ 0.75 & 0.01 & 0.73 & 0.76 & 0.19 & 0.18 & 1.22 & 0.16 & 0.93 & 0.03 \end{bmatrix} \times \begin{bmatrix} 1 & 0.86 & 0.75 \\ 1 & 0.09 & 0.01 \\ 1 & -0.85 & 0.73 \\ 1 & 0.87 & 0.76 \\ 1 & -0.44 & 0.19 \\ 1 & -0.43 & 0.18 \\ 1 & -1.10 & 1.22 \\ 1 & 0.40 & 0.16 \\ 1 & -0.96 & 0.93 \\ 1 & 0.17 & 0.03 \end{bmatrix}$$

$$= \begin{bmatrix} 10 & -1.39 & 4.95 \\ -1.39 & 4.95 & 1.64 \\ 4.95 & 1.64 & 4.11 \end{bmatrix}$$

$X^\mathsf{T}\mathbf{y}$

$$X^\mathsf{T}\mathbf{y} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\ 0.75 & 0.01 & 0.73 & 0.76 & 0.19 & 0.18 & 1.22 & 0.16 & 0.93 & 0.03 \end{bmatrix} \times \begin{bmatrix} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{bmatrix}$$

$$= \begin{bmatrix} 8.34 \\ 6.49 \\ 3.60 \end{bmatrix}$$

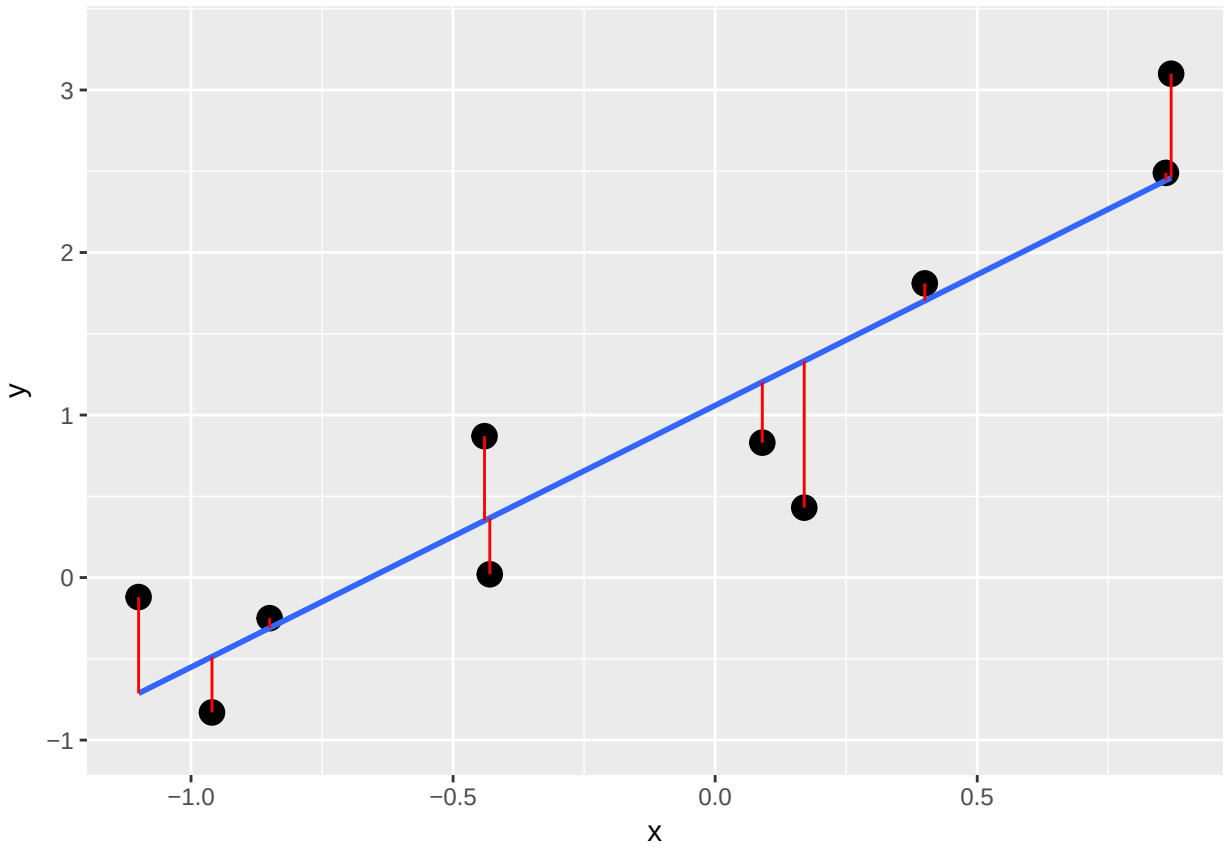
Solving for w

$$\mathbf{w} = (X^\mathsf{T}X)^{-1}X^\mathsf{T}\mathbf{y} = \begin{bmatrix} 10 & -1.39 & 4.95 \\ -1.39 & 4.95 & 1.64 \\ 4.95 & 1.64 & 4.11 \end{bmatrix}^{-1} \begin{bmatrix} 3.60 \\ 6.49 \\ 8.34 \end{bmatrix} = \begin{bmatrix} 0.74 \\ 1.75 \\ 0.69 \end{bmatrix}$$

So the best order-2 polynomial is $y = 0.74 + 1.75x + 0.69x^2$.

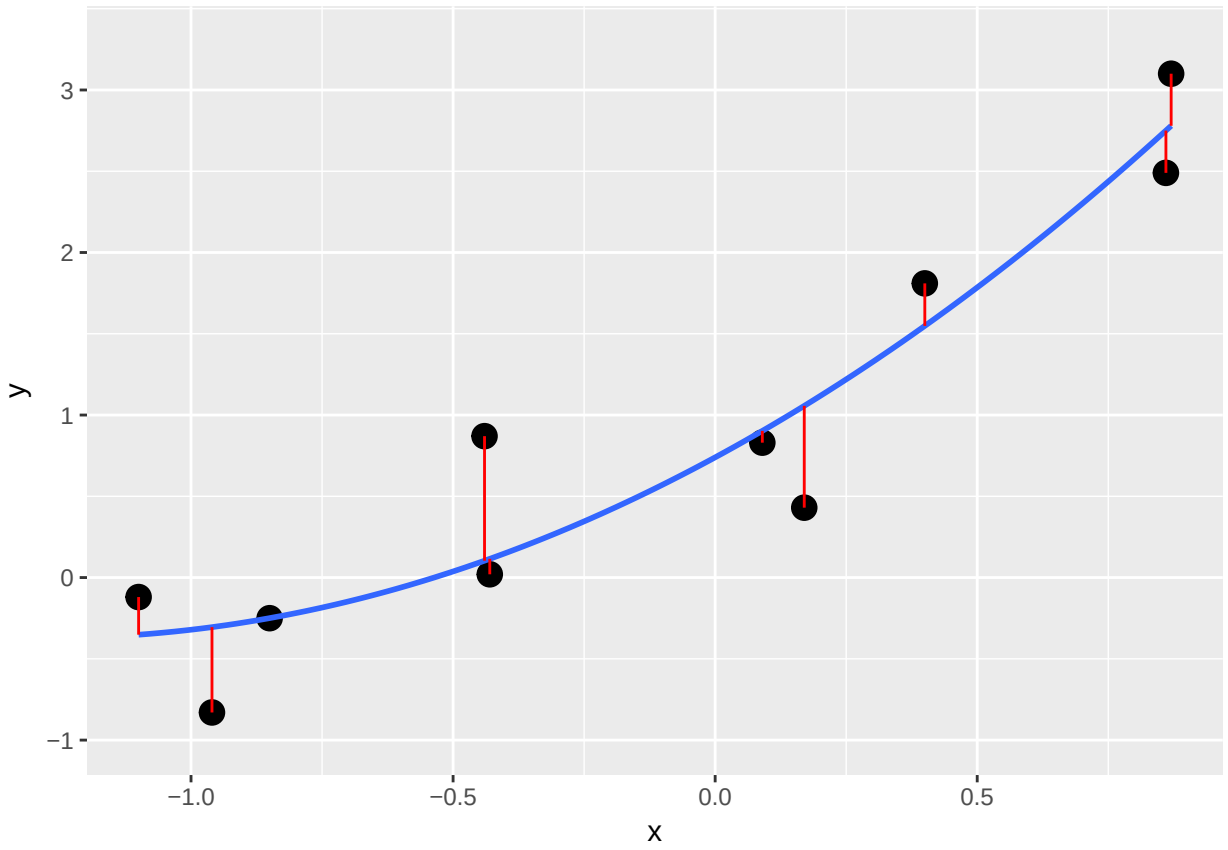
Data and linear fit

```
## (Intercept)           x
##         1.1         1.6
```



Data and quadratic fit

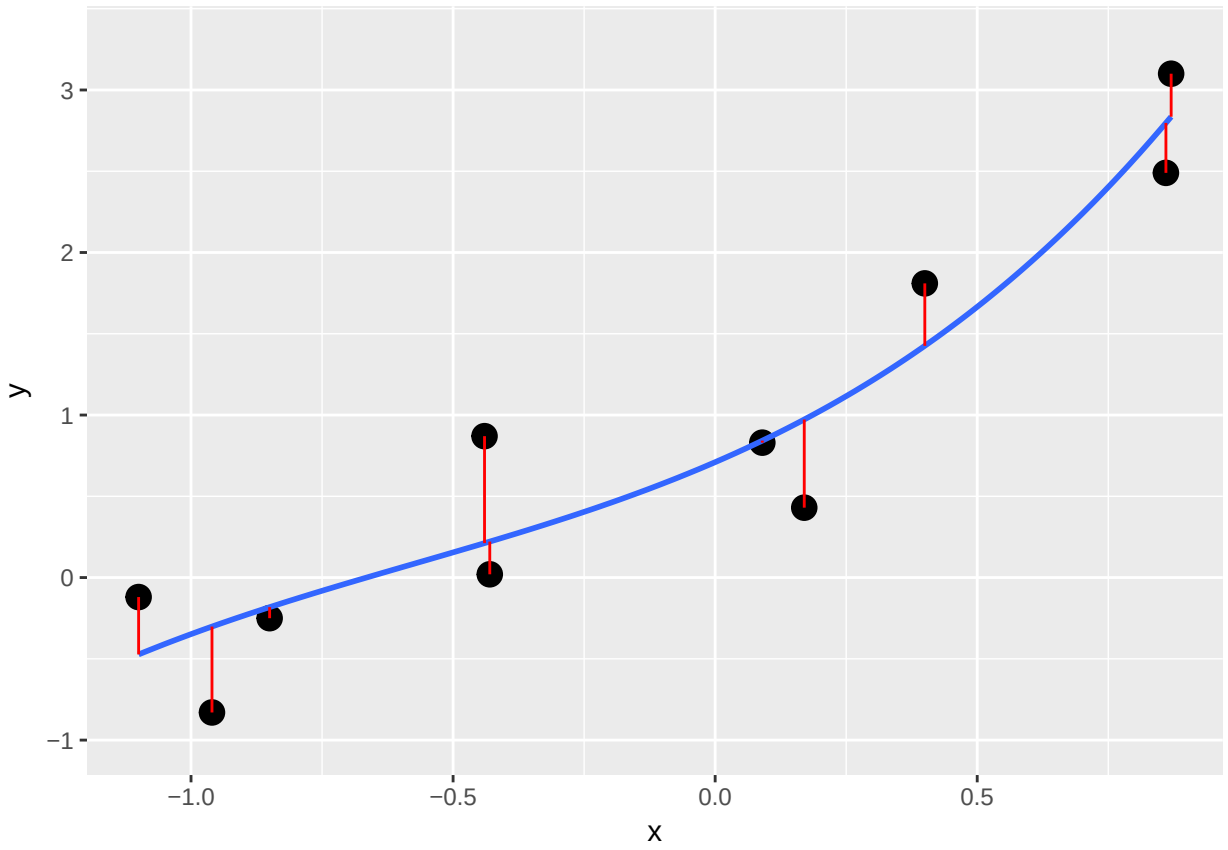
```
## (Intercept)           x      I(x^2)
##        0.74        1.75        0.69
```



Is this a better fit to the data?

Order-3 fit

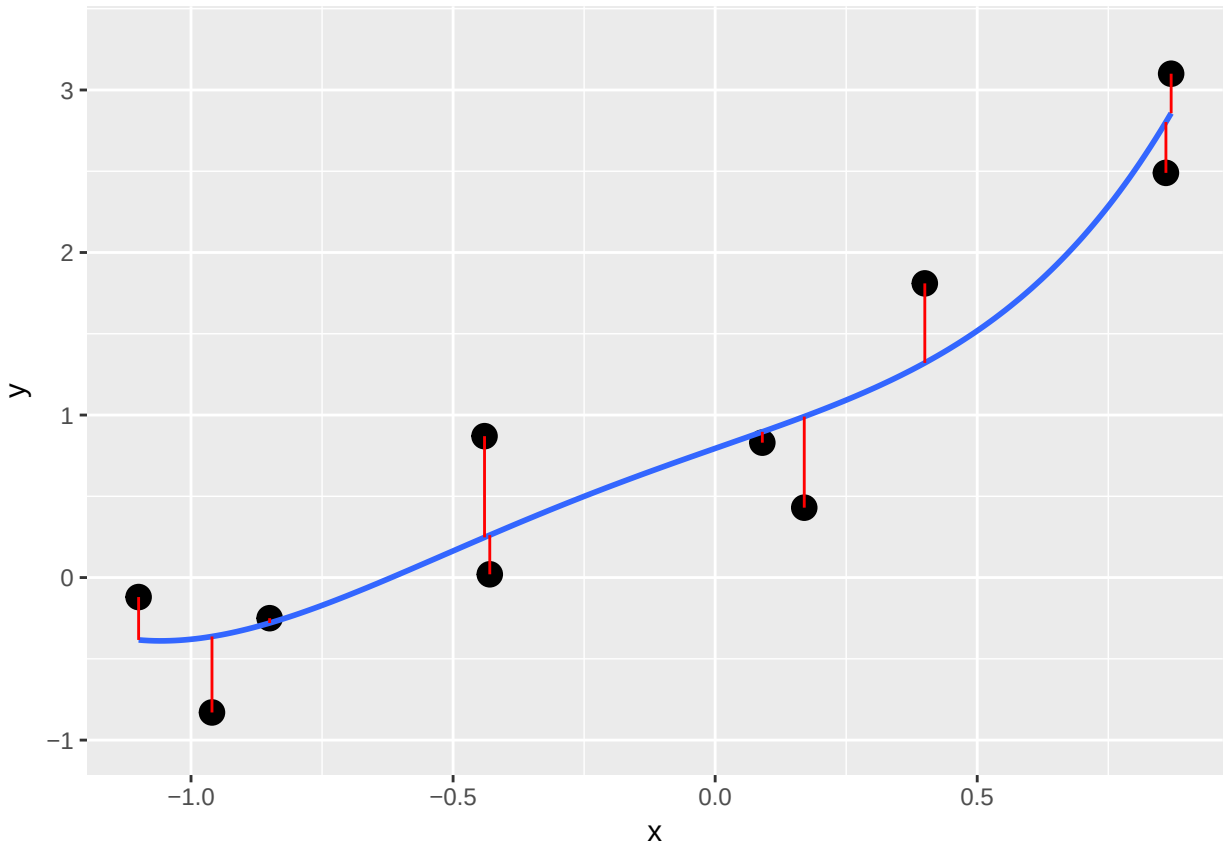
```
## (Intercept)           x      I(x^2)      I(x^3)
##        0.71        1.39        0.80        0.46
```



Is this a better fit to the data?

Order-4 fit

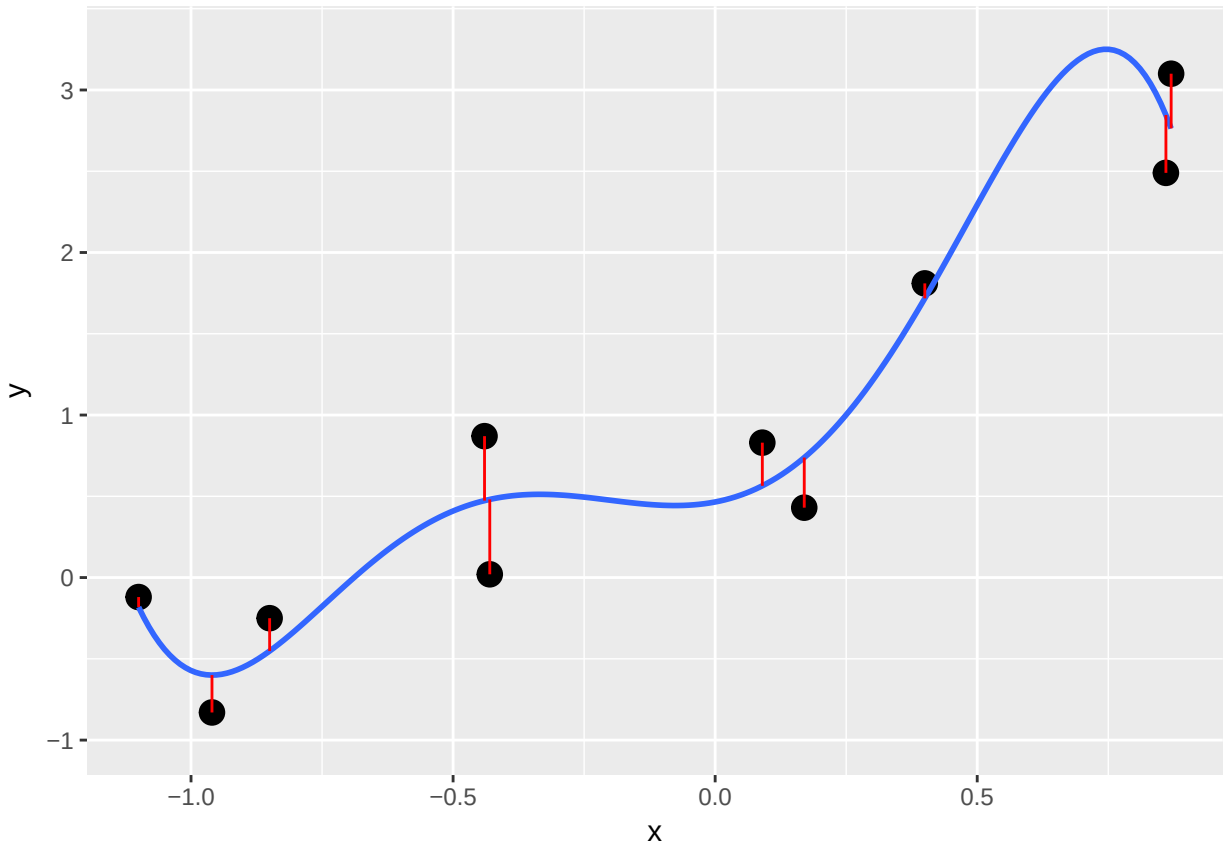
```
## (Intercept)           x      I(x^2)      I(x^3)      I(x^4)
##       0.795       1.128      -0.039       0.905       0.898
```



Is this a better fit to the data?

Order-5 fit

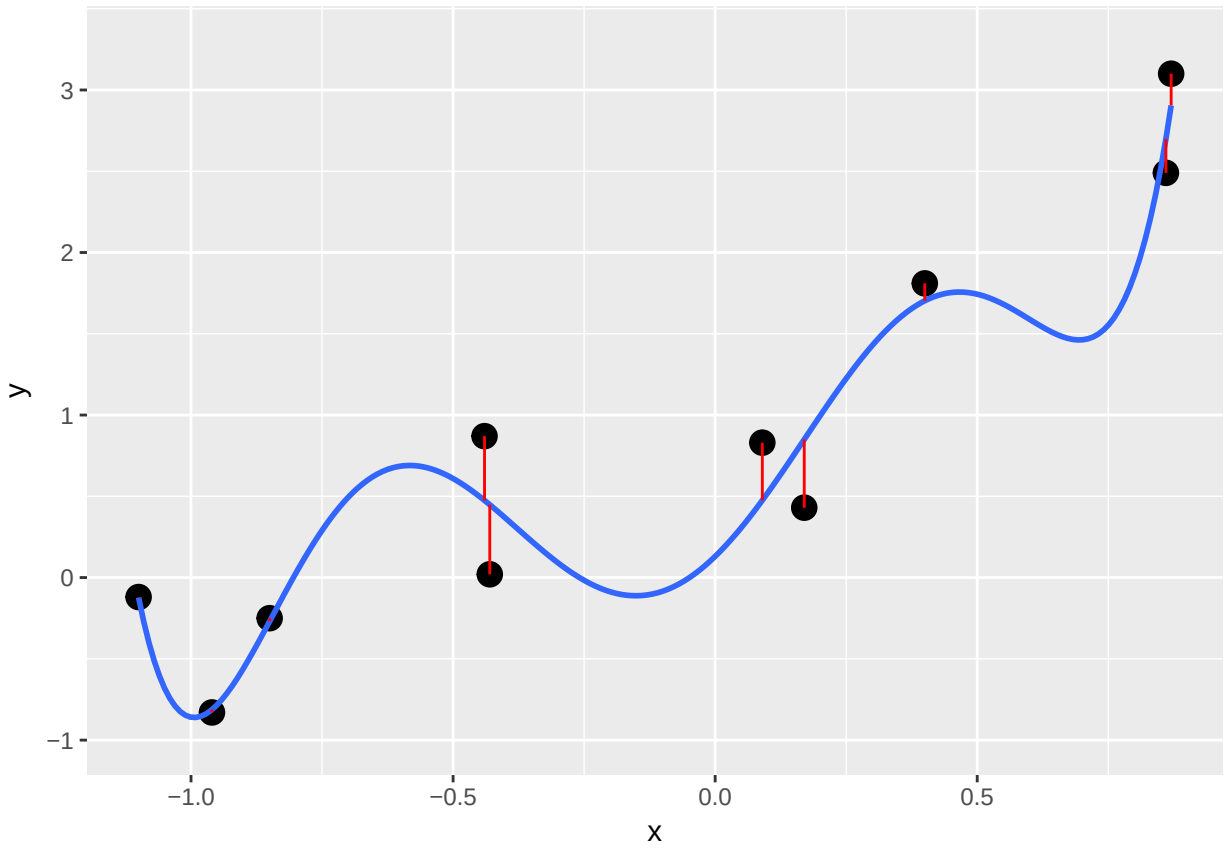
```
## (Intercept)           x      I(x^2)      I(x^3)      I(x^4)      I(x^5)
##        0.47        0.62        4.86        6.75       -5.25       -6.72
```



Is this a better fit to the data?

Order-6 fit

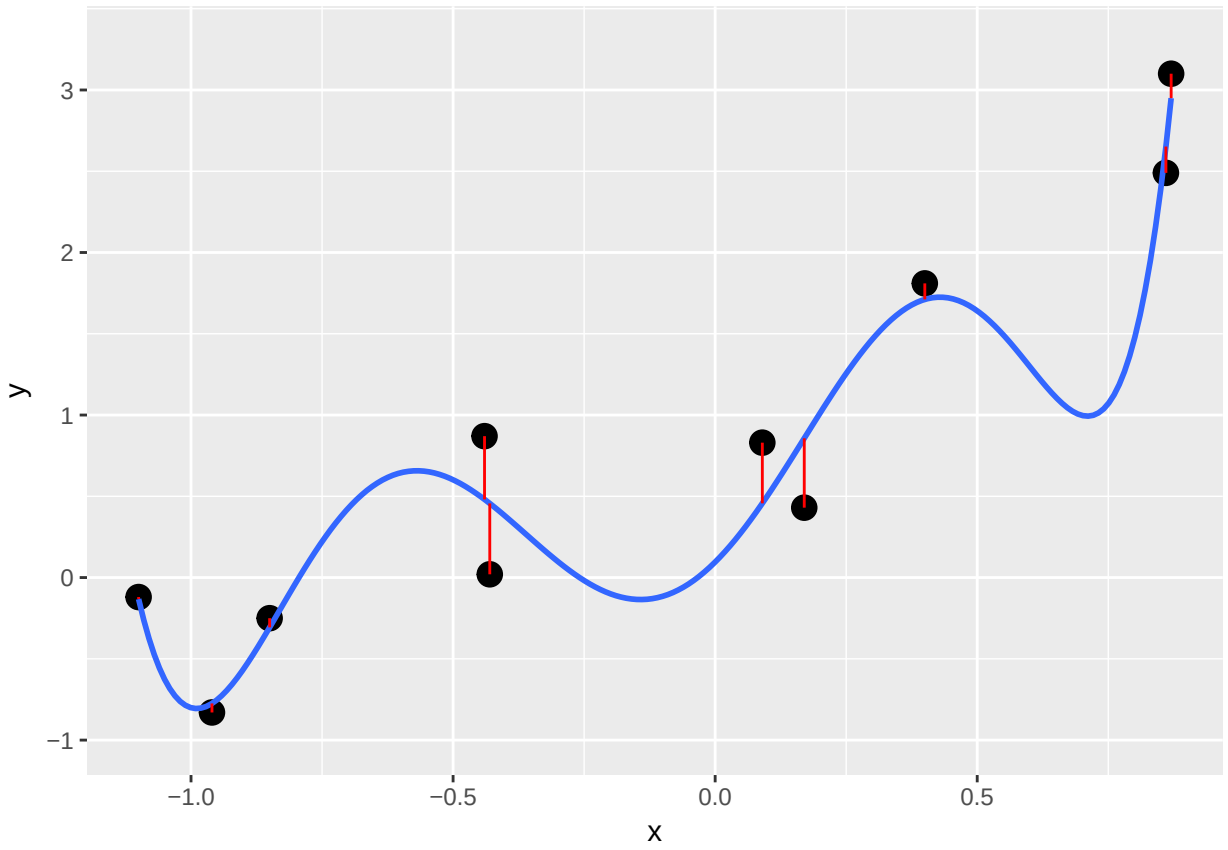
```
## (Intercept)           x      I(x^2)      I(x^3)      I(x^4)      I(x^5)      I(x^6)
##        0.13        3.13        8.99      -11.11      -23.83       12.52       18.38
```



Is this a better fit to the data?

Order-7 fit

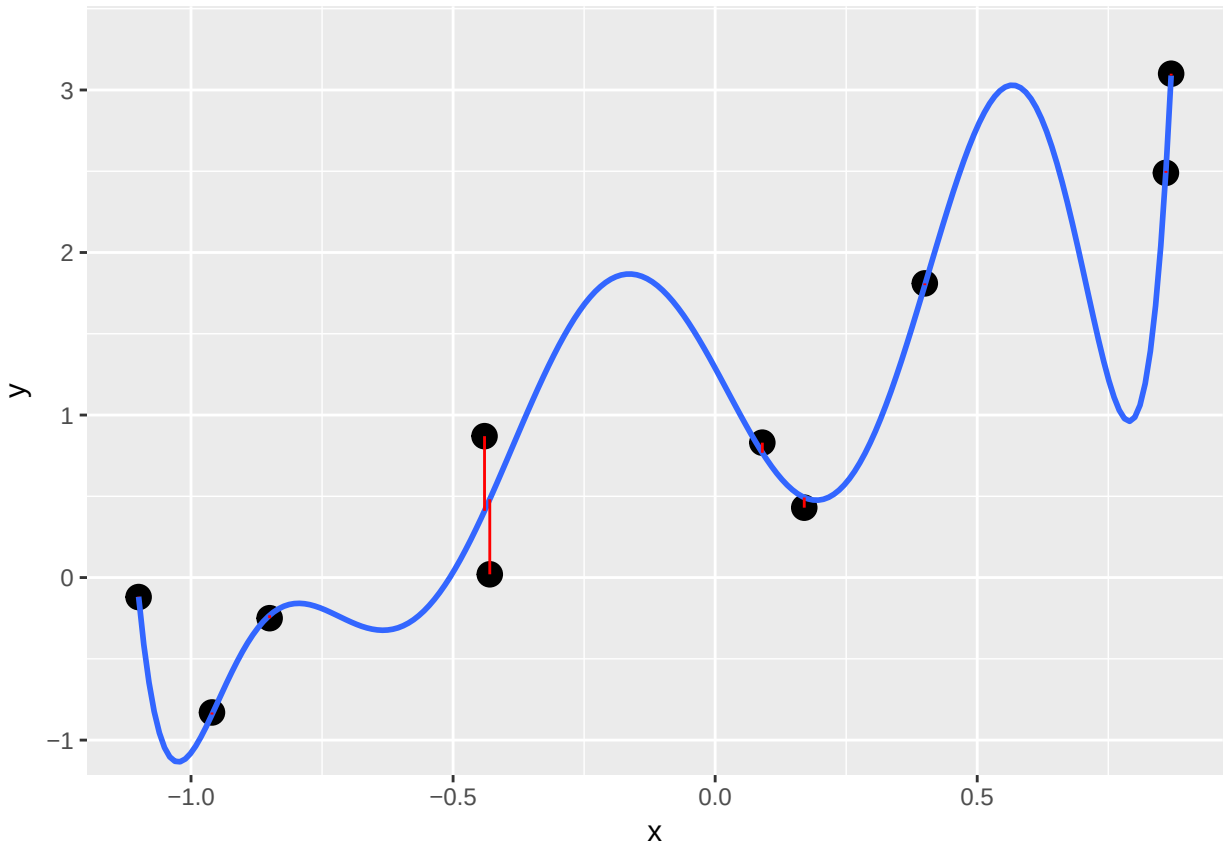
```
## (Intercept)           x      I(x^2)      I(x^3)      I(x^4)      I(x^5)      I(x^6)      I(x^7)
##       0.096       3.207      10.193     -11.078     -30.742       8.263      25.527       5.483
```



Is this a better fit to the data?

Order-8 fit

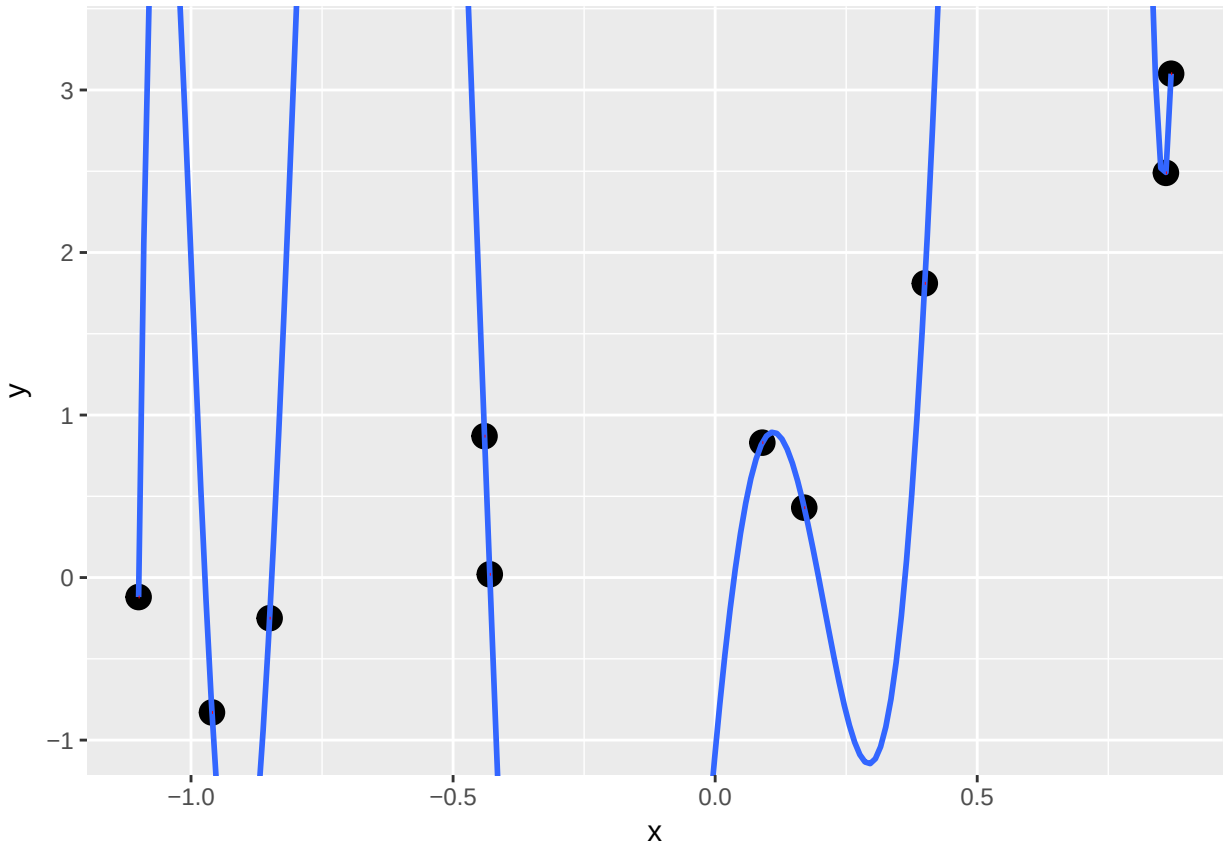
```
## (Intercept)           x      I(x^2)      I(x^3)      I(x^4)      I(x^5)      I(x^6)      I(x^7)
##         1.3        -5.9        -5.1        69.9        48.8      -172.0      -131.9       123.3
```



Is this a better fit to the data?

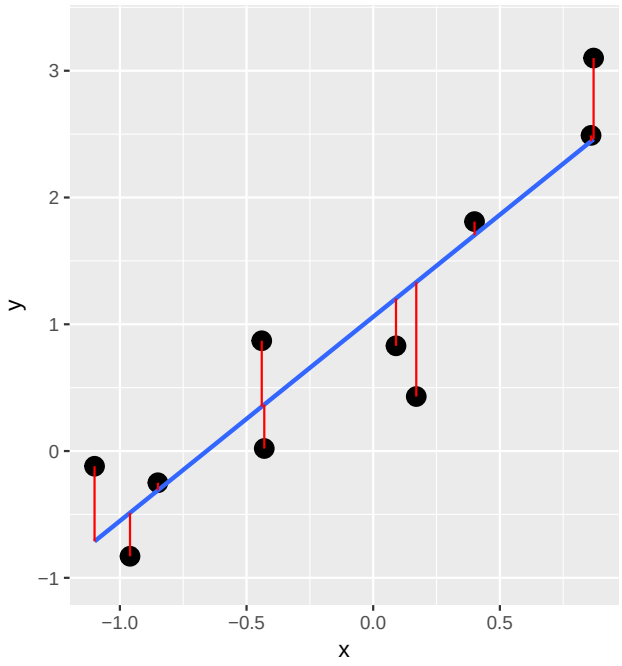
Order-9 fit

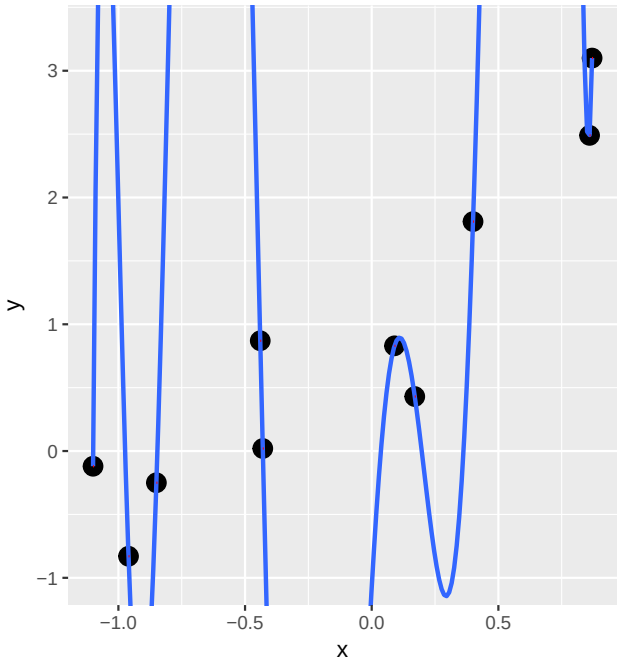
```
## (Intercept)           x      I(x^2)      I(x^3)      I(x^4)      I(x^5)      I(x^6)      I(x^7)
##        -1.1        34.8      -127.9      -379.9      1186.9      1604.8     -2475.4     -2627.6
```



Is this a better fit to the data?

Evaluating Performance





Which do you prefer and why?