

CS195-5 : Introduction to Machine Learning

Lecture 2

Greg Shakhnarovich

September 8, 2006

Revised October 24th, 2006

Today

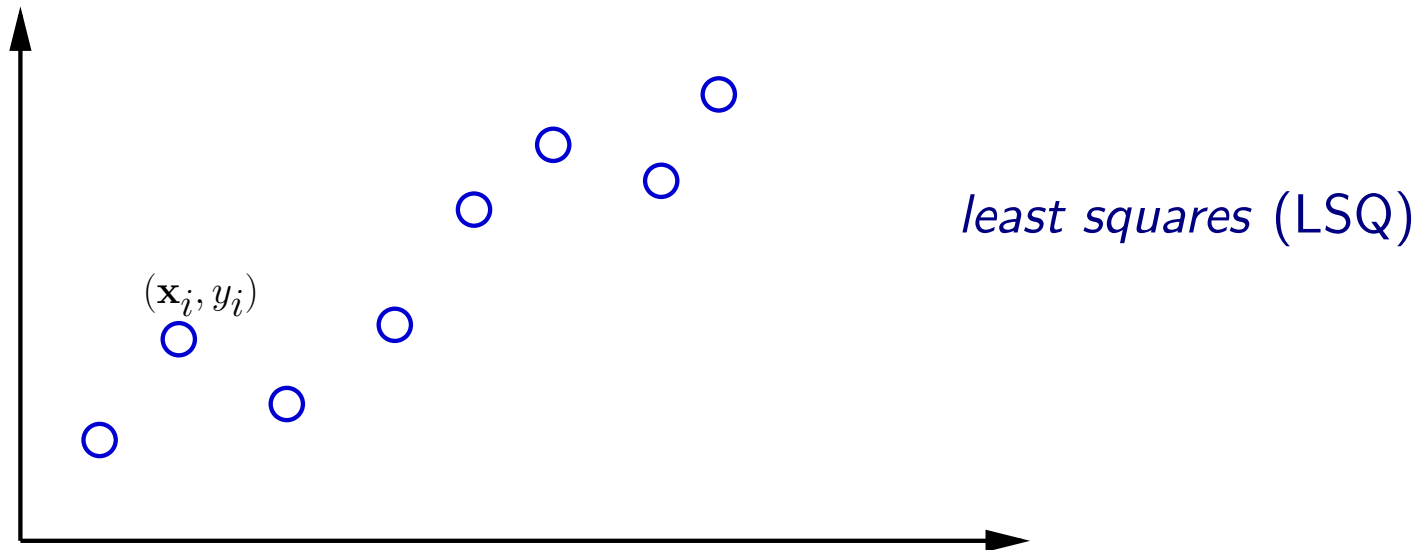
- Some announcements
- Fundamental concepts in statistical learning:
 - Empirical loss
 - Learning via empirical loss minimization
 - Empirical loss versus expected loss: overfitting and generalization
- Linear models for regression; least squares.

Announcements

- Today and Monday: in CIT 368
- Mailing list: subscribe (link on website)
- Machine Learning Reading Group: meeting every Wednesday, 1pm, CIT 506.

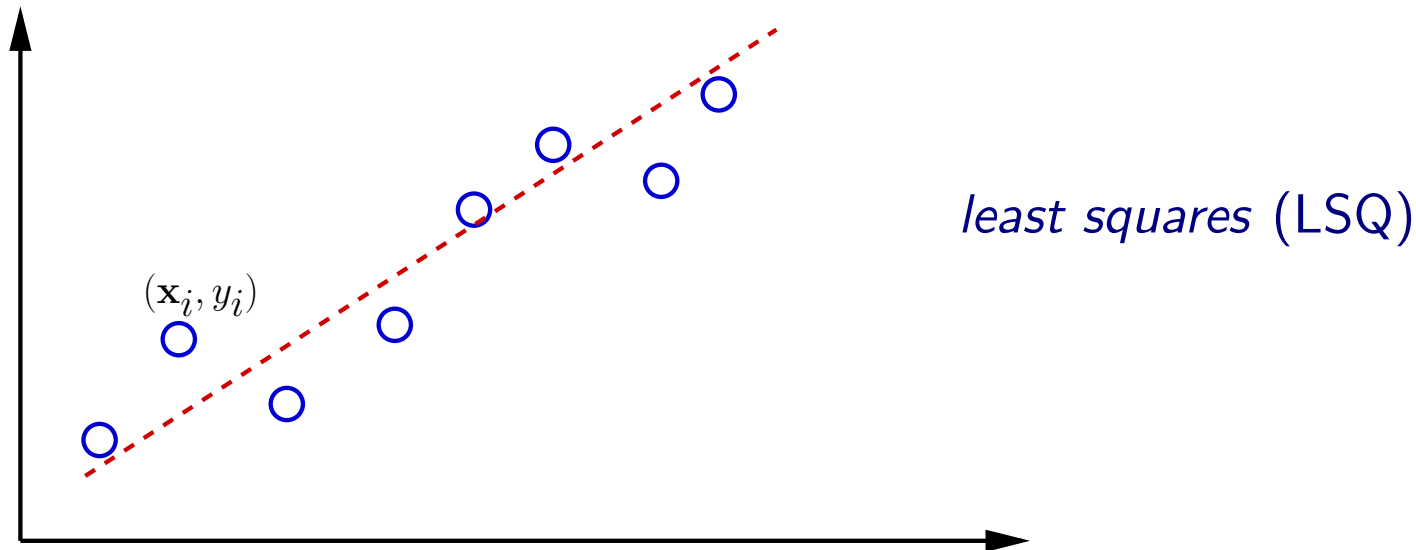
Linear fitting to data

- We want to fit a linear function to an observed set of points $X = [\mathbf{x}_1, \dots, \mathbf{x}_N]$ with associated labels $Y = [y_1, \dots, y_N]$.
 - Once we fit the function, we can use it to *predict* the y for new \mathbf{x} .
- Find the function that minimizes sum (or average) of square distances between actual y s in the training set and predicted ones.



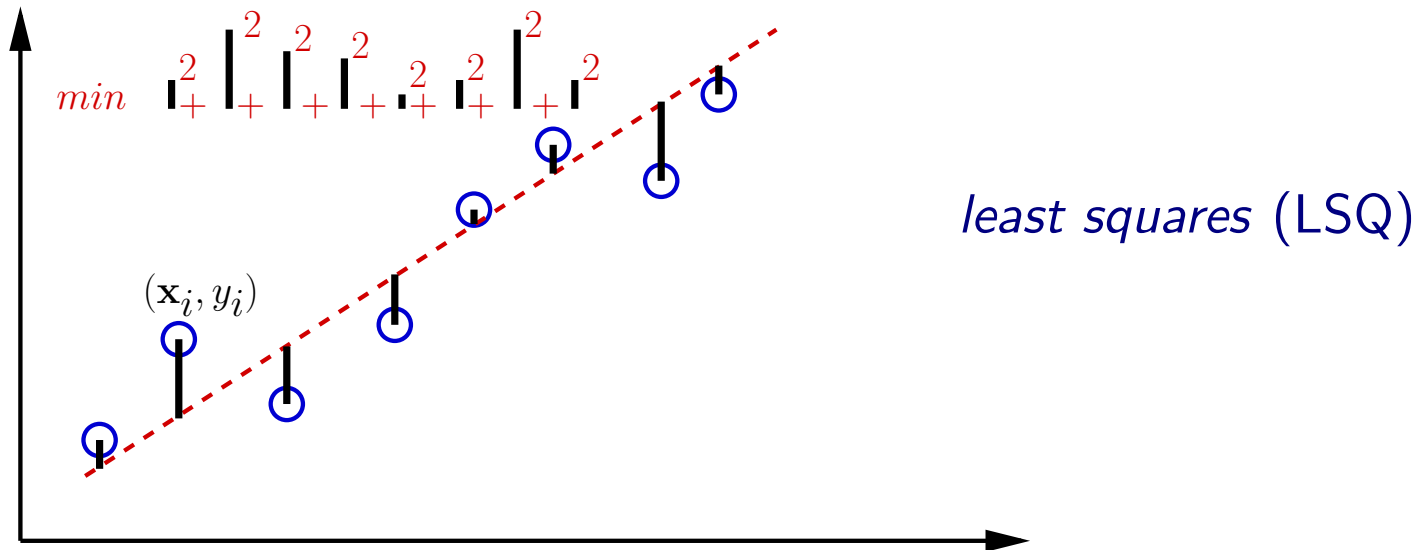
Linear fitting to data

- We want to fit a linear function to an observed set of points $X = [\mathbf{x}_1, \dots, \mathbf{x}_N]$ with associated labels $Y = [y_1, \dots, y_N]$.
 - Once we fit the function, we can use it to *predict* the y for new \mathbf{x} .
- Find the function that minimizes sum (or average) of square distances between actual y s in the training set and predicted ones.



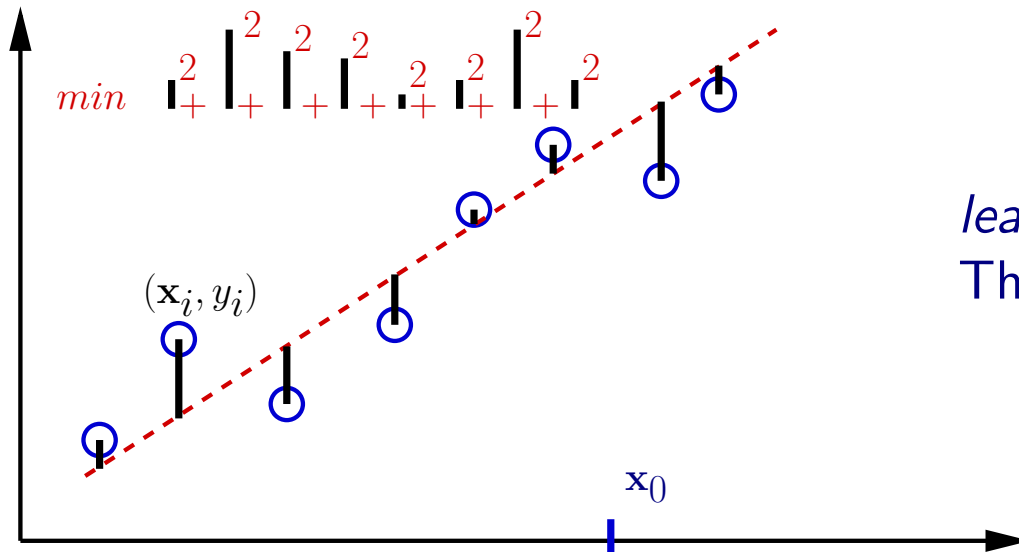
Linear fitting to data

- We want to fit a linear function to an observed set of points $X = [\mathbf{x}_1, \dots, \mathbf{x}_N]$ with associated labels $Y = [y_1, \dots, y_N]$.
 - Once we fit the function, we can use it to *predict* the y for new \mathbf{x} .
- Find the function that minimizes sum (or average) of square distances between actual y s in the training set and predicted ones.



Linear fitting to data

- We want to fit a linear function to an observed set of points $X = [\mathbf{x}_1, \dots, \mathbf{x}_N]$ with associated labels $Y = [y_1, \dots, y_N]$.
 - Once we fit the function, we can use it to *predict* the y for new \mathbf{x} .
- Find the function that minimizes sum (or average) of square distances between actual y s in the training set and predicted ones.

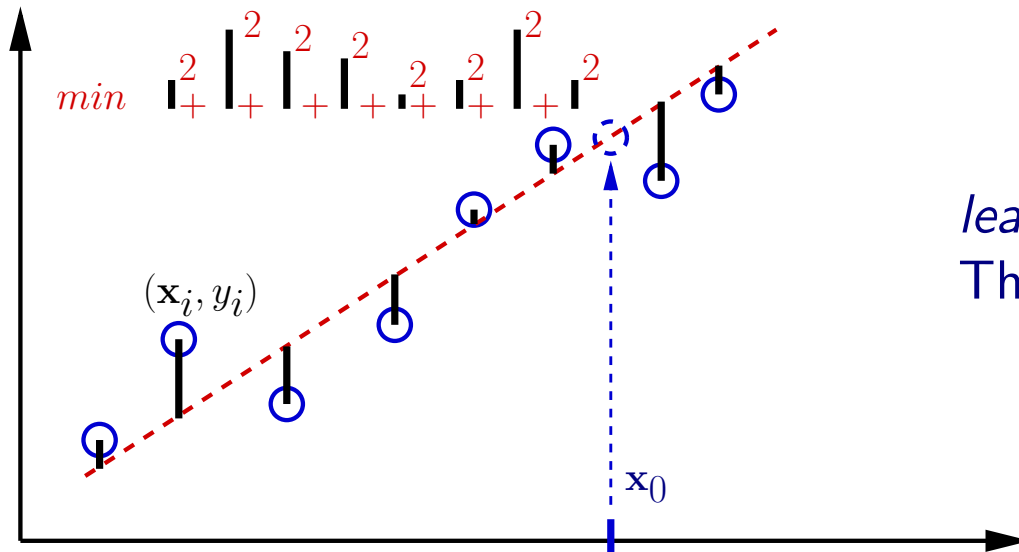


least squares (LSQ)

The fitted line is used as a predictor

Linear fitting to data

- We want to fit a linear function to an observed set of points $X = [\mathbf{x}_1, \dots, \mathbf{x}_N]$ with associated labels $Y = [y_1, \dots, y_N]$.
 - Once we fit the function, we can use it to *predict* the y for new \mathbf{x} .
- Find the function that minimizes sum (or average) of square distances between actual y s in the training set and predicted ones.



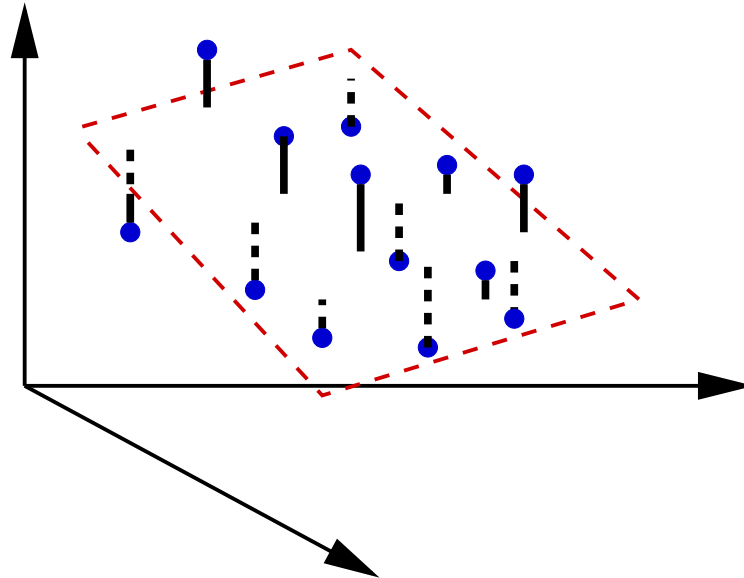
least squares (LSQ)

The fitted line is used as a predictor

Linear functions

- General form: $f(\mathbf{x}; \mathbf{w}) = w_0 + w_1x_1 + \dots + w_dx_d$
- 1D case ($\mathcal{X} = \mathbb{R}$): a line

- $\mathcal{X} = \mathbb{R}^2$: a plane



- *Hyperplane* in general, d -D case.

Notation

We will mostly stick to these throughout the course:

\mathbf{x}_i the i -th data point in \mathcal{X} .

Often $\mathcal{X} \equiv \mathbb{R}^d$, so that $[x_1^{(i)}, \dots, x_d^{(i)}]^T$

y_i the label of the i -th data point; $y_i \in \mathcal{Y}$

\mathbf{x}_0, y_0 a single test point and its (unknown) label

\mathbf{X} the $N \times d$ data matrix where i -th row is \mathbf{x}_i^T

\mathbf{y} the label vector $\mathbf{y} = [y_1, \dots, y_N]^T$

More to come. . .

Loss function

- Suppose target labels are in \mathcal{Y}
 - Binary classification: $\mathcal{Y} = \{-1, +1\}$
 - Regression: $\mathcal{Y} \equiv \mathbb{R}$.
- A *loss function* $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ maps decisions to costs:
 - $L(y, \hat{y})$ defines the penalty paid for predicting \hat{y} when the true value is y .
- Standard choice for classification: 0/1 loss $L_{0/1}(\hat{y}, y) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{otherwise} \end{cases}$
- Standard choice for regression: squared loss $L(\hat{y}, y) = (\hat{y} - y)^2$

Empirical loss

- We consider *parametric* function $f(\mathbf{x}; \mathbf{w})$.
- The *empirical loss* of function $y = f(\mathbf{x}; \mathbf{w})$ on the training set is defined as

$$L_N(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L(f(\mathbf{x}_i; \mathbf{w}), y_i)$$

- LSQ minimizes the empirical loss for squared loss L .
- We care about accuracy of *predicting* labels for new examples. Why/when does empirical loss minimization help us achieve that?

Loss: empirical and expected

- A fundamental assumption in statistical learning: pairs example \mathbf{x} /label y are drawn (sampled) from an joint probability distribution $p(\mathbf{x}, y)$.
- It's the same (unknown!) distribution for all pairs (\mathbf{x}, y) in both training and test data.
- Empirical loss $L_N(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L(f(\mathbf{x}_i; \mathbf{w}), y_i)$
- The ultimate goal is to minimize the *expected loss*, also known as *risk*:

$$R(\mathbf{w}) = E_{(\mathbf{x}_0, y_0) \sim p(\mathbf{x}, y)} [L(f(\mathbf{x}_0; \mathbf{w}), y_0)]$$

Loss: empirical and expected

- A fundamental assumption in statistical learning: pairs example \mathbf{x} /label y are drawn (sampled) from an joint probability distribution $p(\mathbf{x}, y)$.
- It's the same (unknown!) distribution for all pairs (\mathbf{x}, y) in both training and test data.
- Empirical loss $L_N(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L(f(\mathbf{x}_i; \mathbf{w}), y_i)$
- The ultimate goal is to minimize the *expected loss*, also known as *risk*:

$$R(\mathbf{w}) = E_{(\mathbf{x}_0, y_0) \sim p(\mathbf{x}, y)} [L(f(\mathbf{x}_0; \mathbf{w}), y_0)]$$

Expectation of a function $g(Z)$ of a random variable $Z \sim p(z)$:

$$E_{Z \sim p(Z)} [g(Z)] = \int_{\Omega} g(Z) p(Z) dz$$

Loss: empirical and expected

- Empirical loss: $L_N(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L(f(\mathbf{x}_i, \mathbf{w}), y_i)$
- Risk: $R(\mathbf{w}) = E_{(\mathbf{x}_0, y_0) \sim p(\mathbf{x}, y)} [L(f(\mathbf{x}_0, \mathbf{w}), y_0)]$
- To the extent that the training set is a representative of the underlying distribution $p(\mathbf{x}, y)$, the empirical loss serves as a proxy for the risk (expected loss).

Learning via empirical loss minimization

Recall two of the main steps in learning:

- **Modeling:** select a restricted class \mathcal{F} of *hypotheses* $f : \mathcal{X} \rightarrow \mathcal{Y}$
 - Linear functions, parametrized by \mathbf{w} : $\hat{y} = f(\mathbf{x}; \mathbf{w}) = w_0 + \sum_{i=1}^d w_i x_i$
- **Estimation:** select a hypothesis $f^* \in \mathcal{F}$ based on training set (X, Y) .
 - Find the hypothesis that minimizes empirical loss.
 - Regression by least squares fitting: $f^* \equiv f(\mathbf{x}; \mathbf{w}^*)$ where

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^N \left(y_i - w_0 - \sum_{j=1}^d w_j x_j^{(i)} \right)^2$$

- How exactly do we set $\mathbf{w}^* = [w_0^*, w_1^*, \dots, w_d^*]^T$?

Least squares: estimation

- We need to minimize

$$L_N(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - f(\mathbf{x}_i; \mathbf{w}))^2$$

Least squares: estimation

- We need to minimize

$$\begin{aligned}L_N(\mathbf{w}) &= \frac{1}{N} \sum_{i=1}^N (y_i - f(\mathbf{x}_i; \mathbf{w}))^2 \\ &= \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x_1^{(i)} - \dots - w_d x_d^{(i)} \right)^2\end{aligned}$$

Least squares: estimation

- We need to minimize

$$\begin{aligned}L_N(\mathbf{w}) &= \frac{1}{N} \sum_{i=1}^N (y_i - f(\mathbf{x}_i; \mathbf{w}))^2 \\ &= \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x_1^{(i)} - \dots - w_d x_d^{(i)} \right)^2\end{aligned}$$

let's look at 1D for the moment

$$L_N(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right)^2$$

Least squares: estimation

- We need to minimize

$$\begin{aligned}L_N(\mathbf{w}) &= \frac{1}{N} \sum_{i=1}^N (y_i - f(\mathbf{x}_i; \mathbf{w}))^2 \\ &= \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x_1^{(i)} - \dots - w_d x_d^{(i)} \right)^2\end{aligned}$$

let's look at 1D for the moment

$$L_N(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right)^2$$

- Necessary condition to minimize L_N : derivatives w.r.t. w_0 and w_1 must be zero.

Least squares: estimation

$$L_N(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right)^2$$

$$\frac{\partial}{\partial w_0} L_N(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N \frac{\partial}{\partial w_0} \left(y_i - w_0 - w_1 x^{(i)} \right)^2$$

Least squares: estimation

$$L_N(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right)^2$$

$$\begin{aligned} \frac{\partial}{\partial w_0} L_N(w_0, w_1) &= \frac{1}{N} \sum_{i=1}^N \frac{\partial}{\partial w_0} \left(y_i - w_0 - w_1 x^{(i)} \right)^2 \\ &= \frac{1}{N} \sum_{i=1}^N 2 \left(y_i - w_0 - w_1 x^{(i)} \right) \end{aligned}$$

Least squares: estimation

$$L_N(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right)^2$$

$$\begin{aligned} \frac{\partial}{\partial w_0} L_N(w_0, w_1) &= \frac{1}{N} \sum_{i=1}^N \frac{\partial}{\partial w_0} \left(y_i - w_0 - w_1 x^{(i)} \right)^2 \\ &= \frac{1}{N} \sum_{i=1}^N 2 \left(y_i - w_0 - w_1 x^{(i)} \right) \cdot (-1) \end{aligned}$$

Least squares: estimation

$$L_N(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right)^2$$

$$\begin{aligned} \frac{\partial}{\partial w_0} L_N(w_0, w_1) &= \frac{1}{N} \sum_{i=1}^N \frac{\partial}{\partial w_0} \left(y_i - w_0 - w_1 x^{(i)} \right)^2 \\ &= \frac{1}{N} \sum_{i=1}^N 2 \left(y_i - w_0 - w_1 x^{(i)} \right) \cdot (-1) \\ &= -\frac{2}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right) \end{aligned}$$

Least squares: estimation

$$L_N(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right)^2$$

$$\begin{aligned} \frac{\partial}{\partial w_0} L_N(w_0, w_1) &= \frac{1}{N} \sum_{i=1}^N \frac{\partial}{\partial w_0} \left(y_i - w_0 - w_1 x^{(i)} \right)^2 \\ &= \frac{1}{N} \sum_{i=1}^N 2 \left(y_i - w_0 - w_1 x^{(i)} \right) \cdot (-1) \\ &= -\frac{2}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right) = 0. \end{aligned}$$

Least squares: estimation

$$L_N(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right)^2$$

$$\begin{aligned} \frac{\partial}{\partial w_0} L_N(w_0, w_1) &= \frac{1}{N} \sum_{i=1}^N \frac{\partial}{\partial w_0} \left(y_i - w_0 - w_1 x^{(i)} \right)^2 \\ &= \frac{1}{N} \sum_{i=1}^N 2 \left(y_i - w_0 - w_1 x^{(i)} \right) \cdot (-1) \\ &= -\frac{2}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right) = 0. \end{aligned}$$

- $y_i - w_0 - w_1 x^{(i)}$ is the *prediction error* on the i -th example.
- \Rightarrow Necessary condition for optimal \mathbf{w} is that the errors have zero mean.
 - Otherwise we could reduce L_N even further!

Least squares: estimation

$$L_N(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right)^2$$

Similarly,

$$\frac{\partial}{\partial w_1} L_N(w_0, w_1) = -\frac{2}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right) x^{(i)}$$

(2)

Least squares: estimation

$$L_N(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right)^2$$

Similarly,

$$\frac{\partial}{\partial w_1} L_N(w_0, w_1) = -\frac{2}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right) x^{(i)} = 0.$$

(2)

Least squares: estimation

$$L_N(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right)^2$$

Similarly,

$$\frac{\partial}{\partial w_1} L_N(w_0, w_1) = -\frac{2}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right) x^{(i)} = 0.$$

- Second necessary condition: errors are *uncorrelated* with the data! (And with *any linear function* of the data)

(2)

Least squares: estimation

$$L_N(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right)^2$$

Similarly,

$$\frac{\partial}{\partial w_1} L_N(w_0, w_1) = -\frac{2}{N} \sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right) x^{(i)} = 0.$$

- Second necessary condition: errors are *uncorrelated* with the data! (And with *any linear function* of the data)
- Two unknowns w_0, w_1 and two equations, **linear** in $w_0, w_1 \Rightarrow$ always a solution.

$$\sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right) x^{(i)} = 0, \quad (1)$$

$$\sum_{i=1}^N \left(y_i - w_0 - w_1 x^{(i)} \right) = 0 \quad (2)$$

Linear regression and overfitting

- What happens when we only have a single data point?

Linear regression and overfitting

- What happens when we only have a single data point?
 - Ill-posed problem: an infinite number of lines pass through the point and produce “perfect” fit.
- Two points? . . .

Linear regression and overfitting

- What happens when we only have a single data point?
 - Ill-posed problem: an infinite number of lines pass through the point and produce “perfect” fit.
- Two points? . . .
- This is a general phenomenon: the amount of data needed to obtain a meaningful estimate of a model is related to the number of parameters in the model (its *complexity*).

General case (d -dim, matrix form)

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}, \quad \mathbf{y} = [y_1, \dots, y_N]^T, \quad \mathbf{w} = [w_0, w_1, \dots, w_d]^T.$$

In this notation, predictions are $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$, the errors are $\mathbf{y} - \mathbf{X}\mathbf{w}$, and

$$L_N(\mathbf{w}) = \frac{1}{N} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$

General case (d -dim, matrix form)

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}, \quad \mathbf{y} = [y_1, \dots, y_N]^T, \quad \mathbf{w} = [w_0, w_1, \dots, w_d]^T.$$

In this notation, predictions are $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$, the errors are $\mathbf{y} - \mathbf{X}\mathbf{w}$, and

$$L_N(\mathbf{w}) = \frac{1}{N} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$

For any matrices \mathbf{A}, \mathbf{B} , $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$, $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$, $(\mathbf{A}^T)^T = \mathbf{A}$.

General case (d -dim, matrix form)

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}, \quad \mathbf{y} = [y_1, \dots, y_N]^T, \quad \mathbf{w} = [w_0, w_1, \dots, w_d]^T.$$

In this notation, predictions are $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$, the errors are $\mathbf{y} - \mathbf{X}\mathbf{w}$, and

$$L_N(\mathbf{w}) = \frac{1}{N} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = \frac{1}{N} (\mathbf{y}^T - \mathbf{w}^T \mathbf{X}^T) (\mathbf{y} - \mathbf{X}\mathbf{w}).$$

For any matrices \mathbf{A}, \mathbf{B} , $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$, $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$, $(\mathbf{A}^T)^T = \mathbf{A}$.

General case (d -dim, matrix form)

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}, \quad \mathbf{y} = [y_1, \dots, y_N]^T, \quad \mathbf{w} = [w_0, w_1, \dots, w_d]^T.$$

In this notation, predictions are $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$, the errors are $\mathbf{y} - \mathbf{X}\mathbf{w}$, and

$$L_N(\mathbf{w}) = \frac{1}{N} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = \frac{1}{N} (\mathbf{y}^T - \mathbf{w}^T \mathbf{X}^T) (\mathbf{y} - \mathbf{X}\mathbf{w}).$$

For any matrices \mathbf{A}, \mathbf{B} , $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$, $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$, $(\mathbf{A}^T)^T = \mathbf{A}$.

$$\frac{\partial}{\partial \mathbf{w}} L_N(\mathbf{w}) = \dots = -\frac{2}{n} (\mathbf{X}^T \mathbf{y} - \mathbf{X}^T \mathbf{X} \mathbf{w})$$

General case (d -dim, matrix form)

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}, \quad \mathbf{y} = [y_1, \dots, y_N]^T, \quad \mathbf{w} = [w_0, w_1, \dots, w_d]^T.$$

In this notation, predictions are $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$, the errors are $\mathbf{y} - \mathbf{X}\mathbf{w}$, and

$$L_N(\mathbf{w}) = \frac{1}{N} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = \frac{1}{N} (\mathbf{y}^T - \mathbf{w}^T \mathbf{X}^T) (\mathbf{y} - \mathbf{X}\mathbf{w}).$$

For any matrices \mathbf{A}, \mathbf{B} , $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$, $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$, $(\mathbf{A}^T)^T = \mathbf{A}$.

$$\frac{\partial}{\partial \mathbf{w}} L_N(\mathbf{w}) = \dots = -\frac{2}{n} (\mathbf{X}^T \mathbf{y} - \mathbf{X}^T \mathbf{X} \mathbf{w}) = 0$$

General case (d -dim, matrix form)

$$\frac{\partial}{\partial \mathbf{w}} L_N(\mathbf{w}) = -\frac{2}{n} (\mathbf{X}^T \mathbf{y} - \mathbf{X}^T \mathbf{X} \mathbf{w})$$

General case (d -dim, matrix form)

$$\begin{aligned}\frac{\partial}{\partial \mathbf{w}} L_N(\mathbf{w}) &= -\frac{2}{n} (\mathbf{X}^T \mathbf{y} - \mathbf{X}^T \mathbf{X} \mathbf{w}) = 0 \\ \Rightarrow \mathbf{w}^* &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}\end{aligned}$$

- $\mathbf{X}^\dagger \triangleq (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ is called the *Moore-Penrose pseudoinverse* of \mathbf{X} .
- Linear regression in Matlab:

```
% X(i,:) is i-th example, y(i) is i-th label  
wLSQ = pinv([ones(size(X,1),1) X])*y;
```

General case (d -dim, matrix form)

$$\begin{aligned}\frac{\partial}{\partial \mathbf{w}} L_N(\mathbf{w}) &= -\frac{2}{n} (\mathbf{X}^T \mathbf{y} - \mathbf{X}^T \mathbf{X} \mathbf{w}) = 0 \\ \Rightarrow \mathbf{w}^* &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}\end{aligned}$$

- $\mathbf{X}^\dagger \triangleq (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ is called the *Moore-Penrose pseudoinverse* of \mathbf{X} .

- Linear regression in Matlab:

```
% X(i,:) is i-th example, y(i) is i-th label  
wLSQ = pinv([ones(size(X,1),1) X])*y;
```

- Prediction:

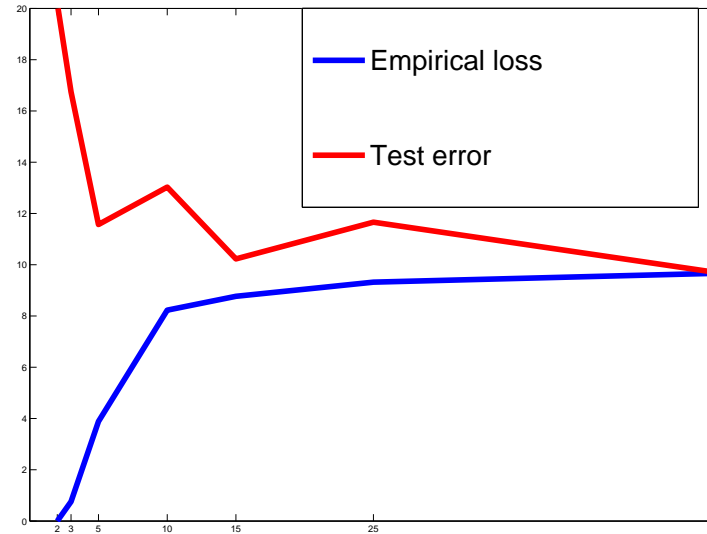
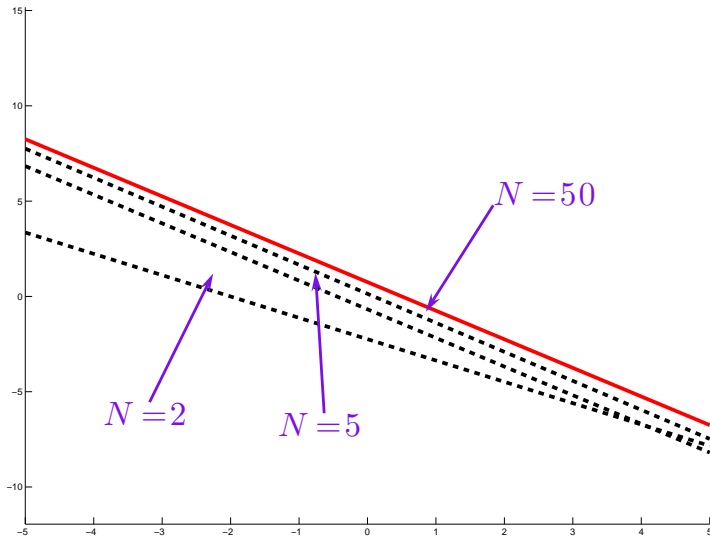
$$\hat{y} = \mathbf{w}^{*T} \begin{bmatrix} 1 \\ \mathbf{x}_0 \end{bmatrix} = \mathbf{y}^T \mathbf{X}^{\dagger T} \begin{bmatrix} 1 \\ \mathbf{x}_0 \end{bmatrix}$$

Linear regression - generalization

- Matlab demo: `1inRegSim1D.m`

Linear regression - generalization

- Matlab demo: 1inRegSim1D.m



- A paradox?
 - The more training data we have, the “worse” is the fit;
 - But at the same time our prediction ability seems to improve.

Next time

We will find the statistical interpretation of the LSQ procedure for regression, and discover an explanation to this behavior.