Neural Computation

Week 2 - Linear Regression

Yunwen Lei

School of Computer Science, University of Birmingham

Outline

Linear Regression

Why studying linear regression?

- Least squares is at least 200 years old going back to Legendre and Gauss
- A fundamental method very easy to understand (theoretically sound)
- Often real processes can be approximated by linear models
- More complex models require understanding linear regression
- Closed form analytic solutions can be obtained
- Many key notions of machine learning can be introduced (regularization)

A Toy Example: Commute Time

Want to predict commute time to UoB

What variables would be useful

- Distance to UoB
- Day of the week

Data

dist(km)	day	commute time (min)
2.7	1	25
4.1	1	33
1.0	0	15
5.2	1	45
2.8	0	22

day = 1 if weekday, 0 if weekend

Problem Setup

Dataset: n input/output pairs

$$S = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$$

- $\mathbf{x}^{(i)} \in \mathbb{R}^d$ is the "input" for the i-th data point as a feature vector with d elements. E.g. (dist, day)
- $y^{(i)} \in \mathbb{R}$ is the "output" for the i-th data point. E.g. Commute Time

Regression Task: find a model, i.e., a function $f : \mathbb{R}^d \mapsto \mathbb{R}$ such that the predicted output $f(X)$ is close to the true output Y

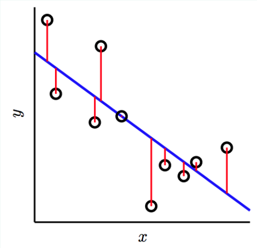
Linear Model: a linear regression model has the form

$$f(\mathbf{x}) = w_0 + w_1 x_1 + \ldots + w_d x_d$$

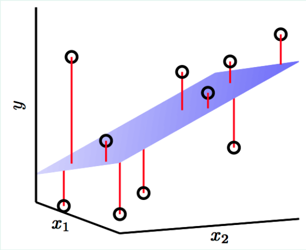
- bias (intercept): w_0
- weight parameter: $w_1, \ldots, w_d$
- feature: x_i is the i-th component of $\mathbf{x} \in \mathbb{R}^d$

Illustration

$\mathbf{x} = (x) \in \mathbb{R}$



$\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$



Linear regression: find linear function with small discrepancy!

- For a matrix $A \in \mathbb{R}^{m \times n}$, $A_{i,j}$ denotes the element in the i-th row and j-th column.

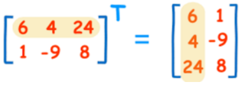
matrix multiplication: If $B \in \mathbb{R}^{n \times r}$, then

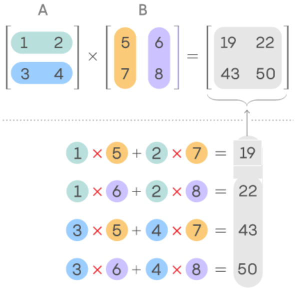
$$(AB)_{i,j} = \sum_{k=1}^{n} A_{i,k} B_{k,j}, \quad AB \in \mathbb{R}^{m \times r}$$

matrix transpose: $A^\top$ is defined by

$$A^\top_{i,j} = A_{j,i}, \quad A^\top \in \mathbb{R}^{n \times m}$$

Transposing a 2x3 matrix to create a 3x2 matrix





- For two vectors $\mathbf{u} = (u_1, \ldots, u_m)^\top, \mathbf{v} = (v_1, \ldots, v_m)^\top \in \mathbb{R}^m$

vector addition

$$\mathbf{u} + \mathbf{v} = \begin{pmatrix} u_1 + v_1 \\ \vdots \\ u_m + v_m \end{pmatrix}$$

dot product

$$\mathbf{u}^\top \mathbf{v} = (u_1, \ldots, u_m) \begin{pmatrix} v_1 \\ \vdots \\ v_m \end{pmatrix} = \sum_{i=1}^{m} u_i v_i$$

Hadamard product

$$\mathbf{u} \odot \mathbf{v} = \begin{pmatrix} u_1 v_1 \\ \vdots \\ u_m v_m \end{pmatrix}$$

Linear Model : Adding a feature for bias term

We can add 1 to get an expanded feature vector

dist(km) x_1	day x_2	commute time y
2.7	1	25
4.1	1	33
1.0	0	15
5.2	1	45
2.8	0	22

$\iff$

one x_0	dist(km) x_1	day x_2	commute time y
1	2.7	1	25
1	4.1	1	33
1	1.0	0	15
1	5.2	1	45
1	2.8	0	22

Linear model

We do not need to consider specially the bias for a linear model

$$f(\mathbf{x}) = w_0 + w_1 x_1 + \ldots + w_d x_d = \underbrace{(w_0, w_1, w_2, \ldots, w_d)}_{:=\mathbf{w}^\top} \underbrace{\begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}}_{:=\bar{\mathbf{x}}} = \mathbf{w}^\top \bar{\mathbf{x}}.$$

For brevity, we do not consider the bias and set $\mathbf{x}$ as the expanded feature vector, i.e., $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$!

Performance Measure

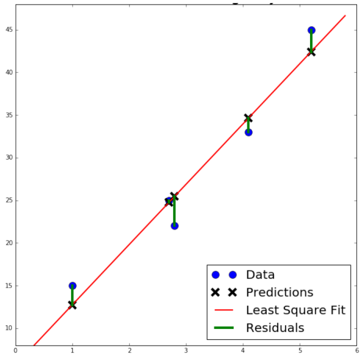
- Want a function $C(\mathbf{w})$ which quantifies the error in the predictions for a given parameter $\mathbf{w}$
- The "**residual**" on the i-th data point can be defined as

$$e^{(i)} := y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)}$$

- The following mean square error (MSR) C takes into account the errors for all n data points

$$C(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^{n} \left(y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)}\right)^2$$

- By squaring the residual, we
  - ignore the sign of the residuals
  - penalize large residuals more (if $e^{(i)} > 1$)



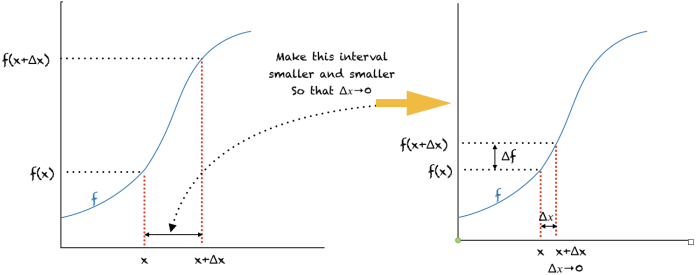
Find the parameter $\mathbf{w}$ which minimizes the loss $C(\mathbf{w})$!

Gradients

Derivative

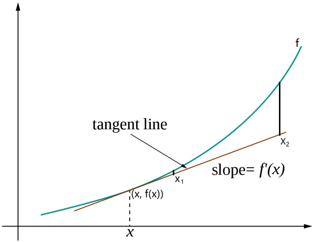
The derivative of a function $f : \mathbb{R} \mapsto \mathbb{R}$ is the rate of change of f

$$f'(x) = \frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}.$$



- The derivative of f at x is the slope of the tangent line to the graph of f at $(x, f(x))$
- The tangent line is the best linear approximation of the function near that input value

$$f(\bar{x}) \approx \underbrace{f(x) + f'(x)(\bar{x} - x)}_{\text{tangent line}}.$$



Derivative of Basic Functions

- Constant function: $a' = 0$.
- Linear function: $(ax)' = a$
- Quadratic function: $(ax^2)' = 2ax$
- Reciprocal function: $(1/x)' = -1/x^2$
- exponential function: $\exp(x)' = \exp(x)$
- logarithmic function: $(\log x)' = 1/x$

This can be proved by definitions. For example

$$
\begin{aligned}
(x^2)' &= \lim_{a \to 0} \frac{(a+x)^2 - x^2}{a} = \lim_{a \to 0} \frac{a^2 + x^2 + 2ax - x^2}{a} \\
&= \lim_{a \to 0} \frac{a^2 + 2ax}{a} = \lim_{a \to 0}(a + 2x) = 2x.
\end{aligned}
$$

Partial Derivatives

Partial derivative

The partial derivative of a **multivariate** function $f(x_1, \ldots, x_d)$ in the direction of variable x_i at $\mathbf{x} = (x_1, \ldots, x_d)$ is

$$\frac{\partial f(x_1, \ldots, x_d)}{\partial x_i} = \lim_{h \to 0} \frac{f(\ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots) - f(x_1, \ldots, x_i, \ldots, x_d)}{h}$$

- Intuitively, $\frac{\partial f}{\partial x_i}$ the derivative of a univariate function

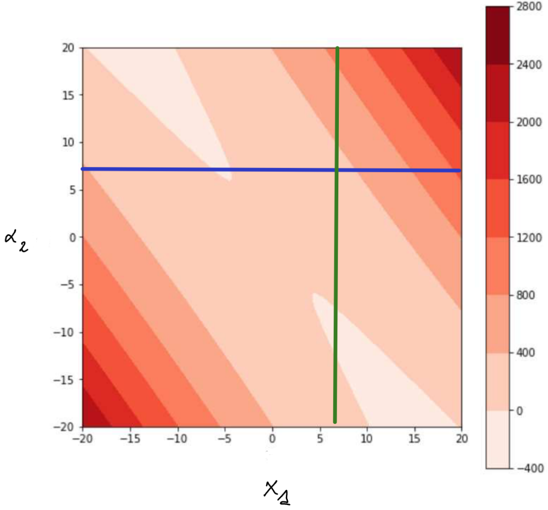
$$g(x_i) := f(x_1, \ldots, x_i, \ldots, x_d),$$

  where all variables except x_i are fixed as constants.
- Example: $f(x_1, x_2) = 2x_1^2 + x_2^2 + 3x_1x_2 + 4$. Then

$$\frac{\partial f}{\partial x_1} = \frac{\partial(2x_1^2 + 3x_1x_2)}{\partial x_1} = 4x_1 + 3x_2$$
$$\frac{\partial f}{\partial x_2} = \frac{\partial(x_2^2 + 3x_1x_2)}{\partial x_2} = 2x_2 + 3x_1.$$

Partial Derivatives: Geometric Interpretation

- $\frac{\partial f}{\partial x_1}$ is the rate of change of f along dimension x_1 (i.e., blue line)
- $\frac{\partial f}{\partial x_2}$ is the rate of change of f along dimension x_2 (i.e., green line)



Gradients

Let $f : \mathbb{R}^d \mapsto \mathbb{R}$. The gradient of f with respect to $\mathbf{x} \in \mathbb{R}^d$ is defined as

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_d}\right)^\top .$$

Example (Linear function). $f(\mathbf{x}) = \mathbf{a}^\top \mathbf{x}$, where $\mathbf{a} = (a_1, \ldots, a_d)^\top$. In this case, the gradient is

$$\nabla(\mathbf{a}^\top \mathbf{x}) = \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_d} \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial x_1} a_1 x_1 + \frac{\partial}{\partial x_1} \sum_{j \neq 1} a_j x_j \\ \frac{\partial}{\partial x_2} a_2 x_2 + \frac{\partial}{\partial x_2} \sum_{j \neq 2} a_j x_j \\ \vdots \\ \frac{\partial}{\partial x_d} a_d x_d + \frac{\partial}{\partial x_d} \sum_{j \neq d} a_j x_j \end{pmatrix} = \begin{pmatrix} a_1 \\ \vdots \\ a_d \end{pmatrix} = \mathbf{a}. \tag{1}$$

Exercise (Quadratic function). If $f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} = \sum_{i,j=1}^d a_{i,j} x_i x_j$, where

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,d} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,d} \\ \vdots & \vdots & \ldots & \vdots \\ a_{d,1} & a_{d,2} & \cdots & a_{d,d} \end{pmatrix}, \quad \text{prove } \nabla\left(\mathbf{x}^\top A \mathbf{x}\right) = A\mathbf{x} + A^\top \mathbf{x}. \tag{2}$$

Gradient is a key concept in optimisation!

Unconstrained Minimization

Given an objective function $C : \mathbb{R}^d \mapsto \mathbb{R}$, we want to solve

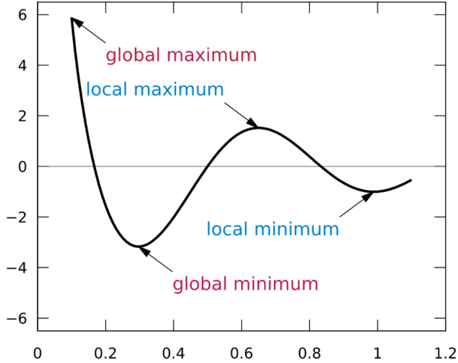
$$\min_{\mathbf{w}\in\mathbb{R}^d} \; C(\mathbf{w})$$

- $\mathbf{w}^*$ is a Global Minimum Point if

$$C(\mathbf{w}) \geq C(\mathbf{w}^*) \quad \forall \mathbf{w} \in \mathbb{R}^d$$

- $\mathbf{w}^*$ is a Local Minimum Point if there exists $\epsilon > 0$ such that

$$C(\mathbf{w}) \geq C(\mathbf{w}^*) \quad \text{for all } \mathbf{w} \text{ within distance } \epsilon \text{ of } \mathbf{w}^*.$$



First-order Necessary Optimality Condition

If $\mathbf{w}^*$ is a local minimum of a differentiable function C, then

$$\nabla C(\mathbf{w}^*) = 0. \tag{3}$$

We say $\mathbf{w}^*$ satisfying Eq. (3) a stationary point.

If $\nabla C(\mathbf{w}^*) \neq 0$, we can move along the direction $-\nabla C(\mathbf{w}^*)$ to get a smaller function value!

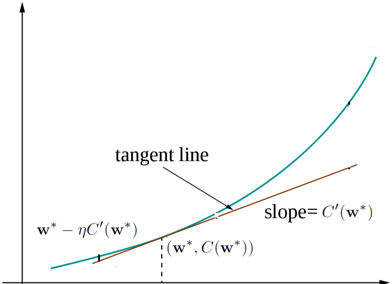
Understanding: (we assume $d = 1$ for brevity)

- If $C'(\mathbf{w}^*) \neq 0$, then $-C'(\mathbf{w}^*)$ is a descent direction
- We can move from $\mathbf{w}^*$ along the direction $\mathbf{v}^* := -C'(\mathbf{w}^*)$ with a step size η

$$C(\mathbf{w}^* + \eta\mathbf{v}^*) \approx \underbrace{C(\mathbf{w}^*) + C'(\mathbf{w}^*)(\eta\mathbf{v}^*)}_{\text{linear approximation}}$$

$$= C(\mathbf{w}^*) - \eta(C'(\mathbf{w}^*))^2 < C(\mathbf{w}^*)$$

for sufficiently small η, showing that $\mathbf{w}^*$ is not a local minimum!



Least Square Regression: One-dimensional Case

- Let $d = 1$. Then

$$C(w) = \frac{1}{2n}\sum_{i=1}^{n}(x^{(i)}w - y^{(i)})^2 = \frac{1}{2n}\sum_{i=1}^{n}\Big(\underbrace{(x^{(i)})^2w^2}_{\text{quadratic}} - \underbrace{2y^{(i)}x^{(i)}w}_{\text{linear}} + \underbrace{(y^{(i)})^2}_{\text{constant}}\Big)$$

- It then follows that

$$C'(w) = \frac{1}{2n}\sum_{i=1}^{n}\Big(2(x^{(i)})^2w - 2y^{(i)}x^{(i)}\Big) = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)})^2w - \frac{1}{n}\sum_{i=1}^{n}y^{(i)}x^{(i)}.$$

- According to the first-order optimality condition, we know the optimal w^* satisfies

$$C'(w^*) = 0 \Longrightarrow \frac{1}{n}\sum_{i=1}^{n}(x^{(i)})^2w^* = \frac{1}{n}\sum_{i=1}^{n}y^{(i)}x^{(i)}$$

- It then follows that

$$w^* = \frac{\sum_{i=1}^{n}y^{(i)}x^{(i)}}{\sum_{i=1}^{n}(x^{(i)})^2}.$$

How about the general case?

Matrix Form for Least Square Regression

Recall $C(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}(\mathbf{x}^{(i)\top}\mathbf{w} - y^{(i)})^2$. Introduce $((\mathbf{x}^{(i)})^\top = (x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}))$

$$X = \begin{pmatrix} (\mathbf{x}^{(1)})^\top \\ \vdots \\ (\mathbf{x}^{(n)})^\top \end{pmatrix} \in \mathbb{R}^{n\times d}, \mathbf{y} = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{pmatrix} \in \mathbb{R}^n \Longrightarrow X\mathbf{w} - \mathbf{y} = \begin{pmatrix} (\mathbf{x}^{(1)})^\top\mathbf{w} - y^{(1)} \\ \vdots \\ (\mathbf{x}^{(n)})^\top\mathbf{w} - y^{(n)} \end{pmatrix}$$

It follows that

$$\begin{aligned}(X\mathbf{w} - \mathbf{y})^\top(X\mathbf{w} - \mathbf{y}) &= \left((\mathbf{x}^{(1)})^\top\mathbf{w} - y^{(1)}, \cdots, (\mathbf{x}^{(n)})^\top\mathbf{w} - y^{(n)}\right) \begin{pmatrix} (\mathbf{x}^{(1)})^\top\mathbf{w} - y^{(1)} \\ \vdots \\ (\mathbf{x}^{(n)})^\top\mathbf{w} - y^{(n)} \end{pmatrix} \\ &= \sum_{i=1}^{n}(\mathbf{x}^{(i)\top}\mathbf{w} - y^{(i)})^2 = 2nC(\mathbf{w})\end{aligned}$$

and (note $(X\mathbf{w})^\top = \mathbf{w}^\top X^\top$)

$$\begin{aligned}C(\mathbf{w}) &= \frac{1}{2n}(X\mathbf{w} - \mathbf{y})^\top(X\mathbf{w} - \mathbf{y}) = \frac{1}{2n}(\mathbf{w}^\top X^\top - \mathbf{y}^\top)(X\mathbf{w} - \mathbf{y}) \\ &= \frac{1}{2n}\left(\mathbf{w}^\top X^\top X\mathbf{w} - 2\mathbf{w}^\top X^\top\mathbf{y} + \mathbf{y}^\top\mathbf{y}\right).\end{aligned}$$

Least Square Regression with Closed-form Solution

- Remember the objective function

$$C(\mathbf{w}) = \frac{1}{2n}\Big(\underbrace{\mathbf{w}^\top X^\top X\mathbf{w}}_{\text{quadratic}} - 2\underbrace{\mathbf{w}^\top X^\top \mathbf{y}}_{\text{linear}} + \underbrace{\mathbf{y}^\top \mathbf{y}}_{\text{constant}}\Big)$$

- We compute the gradient by Eq. (1), (2)

$$\nabla C(\mathbf{w}) = \frac{1}{2n}\Big(2X^\top X\mathbf{w} - 2X^\top \mathbf{y}\Big) \tag{4}$$

- By the first-order necessary condition, we know that optimal $\mathbf{w}^*$ satisfies

$$\nabla C(\mathbf{w}^*) = \frac{1}{n}\Big(X^\top X\mathbf{w}^* - X^\top \mathbf{y}\Big) = 0 \Longrightarrow (X^\top X)\mathbf{w}^* = X^\top \mathbf{y} \quad (\text{normal equation})$$

- If $X^\top X$ is invertible, we get $\mathbf{w}^* = (X^\top X)^{-1}X^\top \mathbf{y}$!
- We can get the prediction

$$\hat{\mathbf{y}} = X\mathbf{w}^* = \begin{pmatrix} (\mathbf{x}^{(1)})^\top \mathbf{w}^* \\ \vdots \\ (\mathbf{x}^{(n)})^\top \mathbf{w}^* \end{pmatrix} \Longrightarrow \hat{\mathbf{y}} = X(X^\top X)^{-1}X^\top \mathbf{y}$$

Back to Toy Example

one	dist(km)	weekday?	commute time (min)
x_0	x_1	x_2	y
1	2.7	1	25
1	4.1	1	33
1	1.0	0	15
1	5.2	1	45
1	2.8	0	22

We have $n = 5, d = 3$ and so we get

$$\mathbf{y} = \begin{pmatrix} 25 \\ 33 \\ 15 \\ 45 \\ 22 \end{pmatrix}, \quad X = \begin{pmatrix} 1 & 2.7 & 1 \\ 1 & 4.1 & 1 \\ 1 & 1.0 & 0 \\ 1 & 5.2 & 1 \\ 1 & 2.8 & 0 \end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

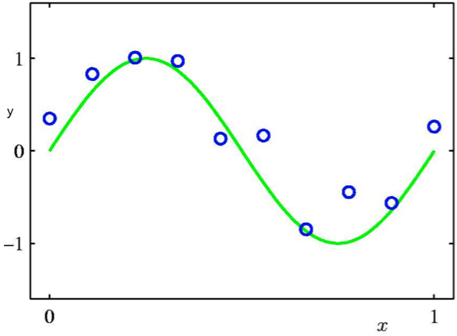
By the closed-form solution, we get

$$\mathbf{w}^* = \begin{pmatrix} 6.09 \\ 6.53 \\ 2.11 \end{pmatrix} \quad \hat{\mathbf{y}} = \begin{pmatrix} 25.84 \\ 34.99 \\ 12.62 \\ 42.17 \\ 24.38 \end{pmatrix} \Longrightarrow \hat{\mathbf{e}} = \begin{pmatrix} -0.84 \\ -1.99 \\ 2.38 \\ 2.82 \\ -2.38 \end{pmatrix}$$

Polynomial Regression

Polynomial regression

- Suppose we want to model the following data



- The input-output relationship is nonlinear!
- One option: fit a low-degree polynomial; this is known as polynomial regression

$$f(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M.$$

Do we need to derive a whole new algorithm?

Feature Mappings

- We get polynomial regression for free!
- Define the feature map

$$\phi(x) = \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \end{pmatrix} \begin{array}{l} \longrightarrow \text{1st feature} \\ \longrightarrow \text{2nd feature} \\ \longrightarrow \text{3rd feature} \\ \longrightarrow \text{4st feature} \end{array}$$

- Polynomial regression model now becomes a linear model w.r.t. the new feature

$$f(x) = \mathbf{w}^\top \phi(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

We transform a univariate nonlinear problem to a multivariate linear problem!

- All of the derivations and algorithms so far in this lecture remain exactly the same!

$$X = \begin{pmatrix} (\mathbf{x}^{(1)})^\top \\ (\mathbf{x}^{(2)})^\top \\ \vdots \\ (\mathbf{x}^{(n)})^\top \end{pmatrix} \mapsto \begin{pmatrix} (\phi(x^{(1)}))^\top \\ (\phi(x^{(2)}))^\top \\ \vdots \\ (\phi(x^{(n)}))^\top \end{pmatrix} = \begin{pmatrix} 1 & x^{(1)} & (x^{(1)})^2 & (x^{(1)})^3 \\ 1 & x^{(2)} & (x^{(2)})^2 & (x^{(2)})^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x^{(n)} & (x^{(n)})^2 & (x^{(n)})^3 \end{pmatrix} := \bar{X}.$$

Feature Mappings

$$\begin{pmatrix} f(x^{(1)}) \\ \vdots \\ f(x^{(n)}) \end{pmatrix} = \begin{pmatrix} (\phi(x^{(1)}))^\top \mathbf{w} \\ \vdots \\ (\phi(x^{(n)}))^\top \mathbf{w} \end{pmatrix} = \bar{X}\mathbf{w} \Longrightarrow \min_{\mathbf{w}} \|\bar{X}\mathbf{w} - \mathbf{y}\|_2^2$$

- The optimal weights become

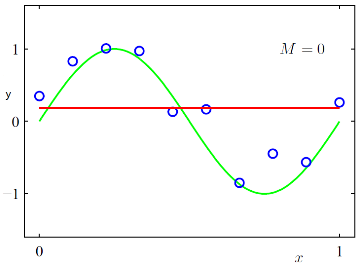
$$\mathbf{w}^* = (\bar{X}^\top \bar{X})^{-1} \bar{X}^\top \mathbf{y}$$

- The prediction on training examples become

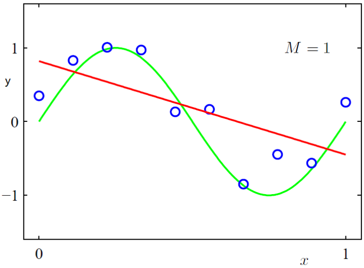
$$\hat{\mathbf{y}} = \bar{X}\mathbf{w}^* = \bar{X}(\bar{X}^\top \bar{X})^{-1} \bar{X}^\top \mathbf{y}.$$

Fitting Polynomials

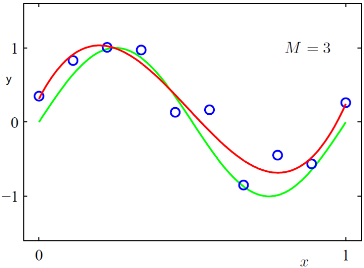
$$y = w_0$$



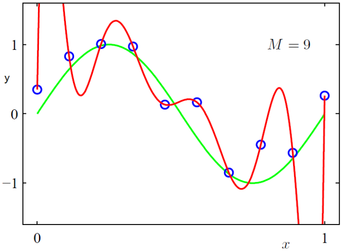
$$y = w_0 + w_1 x$$



$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$



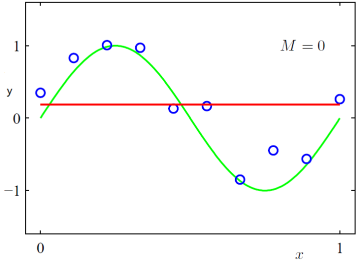
$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \ldots + w_9 x^9$$



Underfitting versus Overfitting

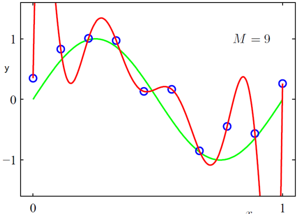
Underfitting: the model is too simple — does not fit the data

$$y = w_0$$



Overfitting: the model is too complex — fits perfectly, does not generalize to new data!

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \ldots + w_9 x^9$$



Regularization

- Generalization=model's ability to predict the unseen data
- Our model with $M = 9$ overfits the data
- One way to handle this is to encourage the weights to be small (this way no feature will have too much influence on prediction). This is called regularization

Regularized Least Square Regression

Given dataset $S = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$ and a regularization parameter $\lambda > 0$, find a model to minimize

$$C(\mathbf{w}) = \underbrace{\frac{1}{2n} \sum_{i=1}^{n} \left(y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)}\right)^2}_{\text{fitting to data}} + \underbrace{\frac{\lambda}{2} \|\mathbf{w}\|_2^2}_{\text{regularizer}}$$

p-norm

For a vector $\mathbf{v} = (v_1, \ldots, v_d) \in \mathbb{R}^d$, the p-norm is (2-norm is the Euclidean norm)

$$\|\mathbf{v}\|_p = \left(|v_1|^p + |v_2|^p + \cdots + |v_d|^p\right)^{1/p}.$$

The regularized least square regression also has a closed-form solution.

- One can show that $\nabla\|\mathbf{w}\|_2^2 = 2\mathbf{w}$ and therefore

$$\nabla C(\mathbf{w}) = \frac{1}{2n}\Big(2X^\top X\mathbf{w} - 2X^\top\mathbf{y}\Big) + \lambda\mathbf{w}.$$

- According to the first-order optimality condition, we know $\nabla C(\mathbf{w}^*) = 0$, i.e.,

$$\nabla C(\mathbf{w}^*) = \frac{1}{n}\Big(X^\top X\mathbf{w}^* - X^\top\mathbf{y}\Big) + \lambda\mathbf{w}^* = 0 \Longrightarrow \Big(\frac{1}{n}(X^\top X) + \lambda\mathbb{I}\Big)\mathbf{w}^* = \frac{1}{n}X^\top\mathbf{y}.$$

- It then follows that ($\mathbb{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix)

$$\mathbf{w}^* = \Big(\frac{1}{n}(X^\top X) + \lambda\mathbb{I}_n\Big)^{-1}\Big(\frac{1}{n}X^\top\mathbf{y}\Big).$$

- A nice property is that we do not require to assume that the matrix $X^\top X$ is invertible. We can always find the above solution for the regularized least regression problem.
- If $\lambda = 0$, then this becomes the solution of the least squares regression problem. If $\lambda = \infty$, we get $\mathbf{w}^* = 0$, which is a trivial solution. We need to choose an appropriate λ.

Summary

- Linear regression (least square regression)
  - model linear relationship between input and output (task)
  - mean square error as objective function (performance)
  - closed-form solution
- Derivative and gradient
- Polynomial regression
  - Underfitting and overfitting
  - Regularization

Next Lecture

Gradient Descent