EECS 545: Machine Learning
Lecture 3. Linear Regression (part 2)

Honglak Lee

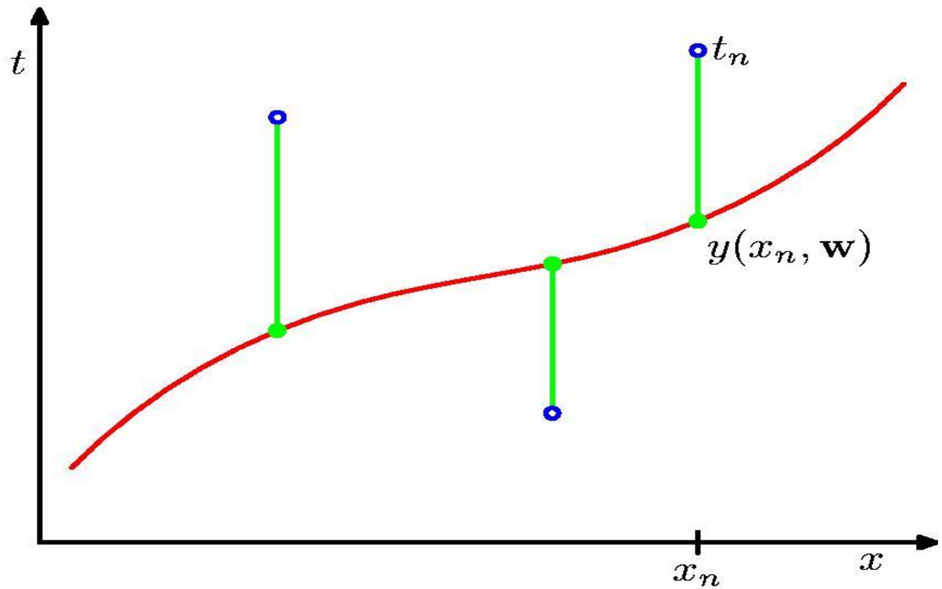
1/14/2026

UNIVERSITY OF MICHIGAN

Outline

- Linear regression review
- Regularized linear regression
- Review on probability
- Maximum likelihood interpretation of linear regression
- Locally-weighted linear regression

Regression, sum of square error



Linear

$$h(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left\{ h(\mathbf{x}^{(n)}, \mathbf{w}) - y^{(n)} \right\}^2$$

We want to find w that minimizes E(**w**) over the training data.

Least squares problem

- Objective function

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\sum_{j=0}^{M-1} w_j\phi_j(\mathbf{x}^{(n)}) - y^{(n)})^2$$

M: dimension of feature, N: number of data

$\phi_j(\mathbf{x}^{(n)})$: j-th feature of data

- Two ways to find w that minimizes E(w)

1. Gradient descent

2. Closed-form solution

Least squares problem

- Objective function

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}^{(n)}) - y^{(n)})^2$$

1. Gradient Descent

$$\frac{\partial E(w)}{\partial w_k} = \sum_{n=1}^{N}\left(\sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}^{(n)}) - y^{(n)}\right)\phi_k(\mathbf{x}^{(n)})$$

$$\nabla_{\mathbf{w}} E(\mathbf{w}) = \sum_{n=1}^{N}\left(\mathbf{w}^\top \phi(\mathbf{x}^{(n)}) - y^{(n)}\right)\phi(\mathbf{x}^{(n)})$$

$$\mathbf{w} := \mathbf{w} - \eta \nabla_{\mathbf{w}} E(\mathbf{w})$$

2. Closed-form solution

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\sum_{j=0}^{M-1} w_j \phi_j\left(\mathbf{x}^{(n)}\right) - y^{(n)}\right)^2$$

$$= \frac{1}{2}\mathbf{w}^\top \Phi^\top \Phi \mathbf{w} - \mathbf{w}^\top \Phi^\top \mathbf{y} + \frac{1}{2}\mathbf{y}^\top \mathbf{y}$$

Recap

w: M by 1 $\quad \mathbf{w} = [w_0, w_1, \cdots, w_{M-1}]^\top$

y : N by 1 $\quad \mathbf{y} = [y^{(1)}, y^{(2)}, \cdots, y^{(N)}]^\top$

Φ: N by M

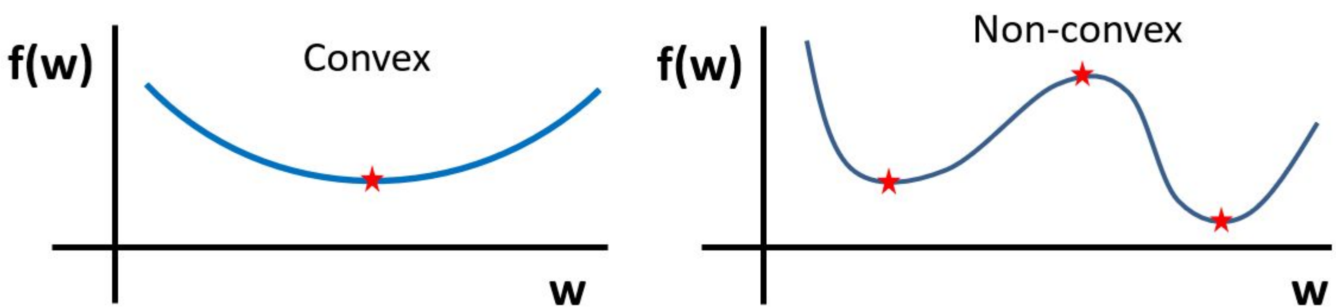
$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}^{(1)}) & \phi_1(\mathbf{x}^{(1)}) & \ldots & \phi_{M-1}(\mathbf{x}^{(1)}) \\ \phi_0(\mathbf{x}^{(2)}) & \phi_1(\mathbf{x}^{(2)}) & \ldots & \phi_{M-1}(\mathbf{x}^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}^{(N)}) & \phi_1(\mathbf{x}^{(N)}) & \ldots & \phi_{M-1}(\mathbf{x}^{(N)}) \end{pmatrix}$$

Useful trick: Matrix Calculus

- Compute gradient and set gradient to 0
  (condition for optimal solution)

Note: Least squared is a convex problem.

$$\nabla_{\mathbf{w}} E(\mathbf{w}) = 0$$



- Need to compute the first derivative in matrix form

Gradient via matrix calculus

- Compute gradient and set to zero

$$\begin{aligned}\nabla_{\mathbf{w}} E(\mathbf{w}) &= \nabla_{\mathbf{w}} \left(\frac{1}{2} \mathbf{w}^\top \Phi^\top \Phi \mathbf{w} - \mathbf{w}^\top \Phi^\top \mathbf{y} + \frac{1}{2} \mathbf{y}^\top \mathbf{y} \right) \\ &= \Phi^\top \Phi \mathbf{w} - \Phi^\top \mathbf{y} \\ &= \mathbf{0}\end{aligned}$$

- Solve the resulting equation (normal equation)

$$\Phi^\top \Phi \mathbf{w} = \Phi^\top \mathbf{y}$$

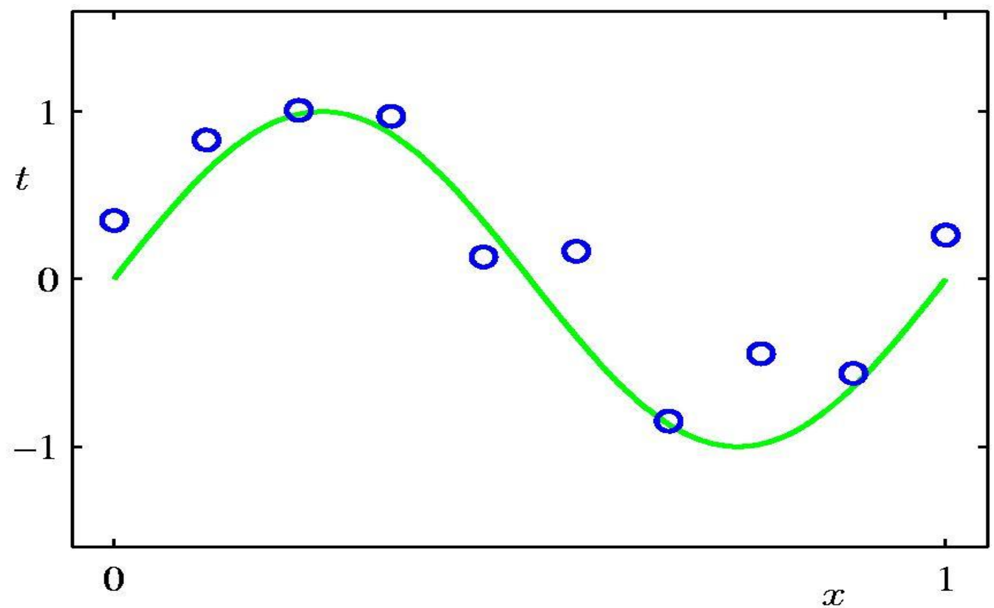
$$\mathbf{w}_{\mathrm{ML}} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y}$$

This is the *Moore-Penrose pseudo-inverse*: $\Phi^\dagger = (\Phi^\top \Phi)^{-1} \Phi^\top$

applied to: $\Phi \mathbf{w} \approx \mathbf{y}$

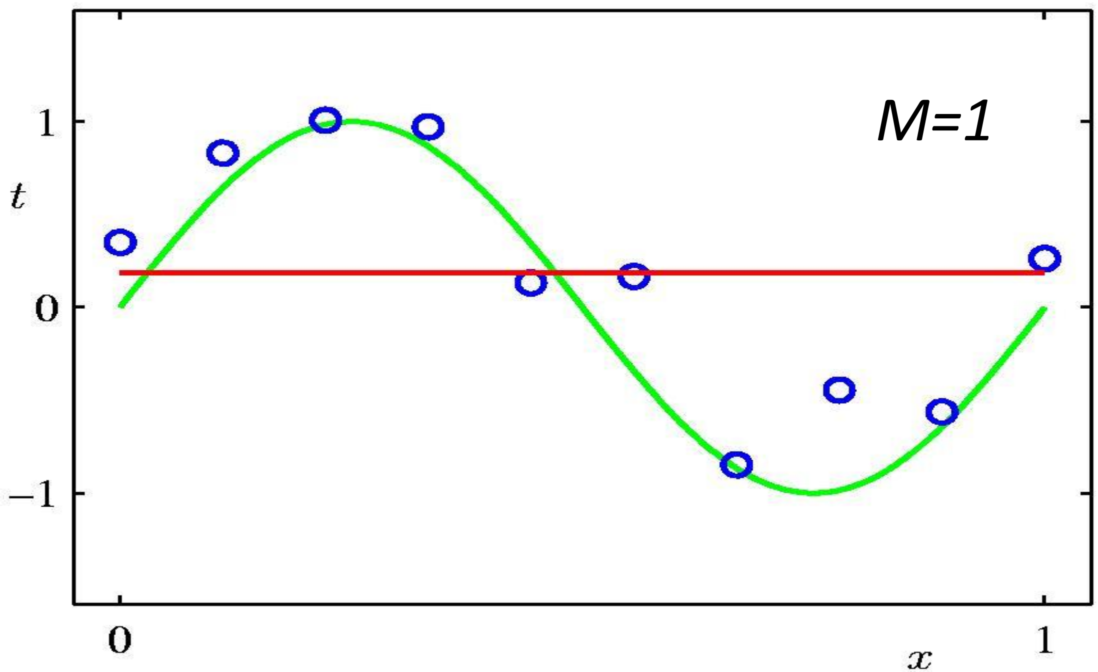
Back to curve-fitting examples

Polynomial Curve Fitting

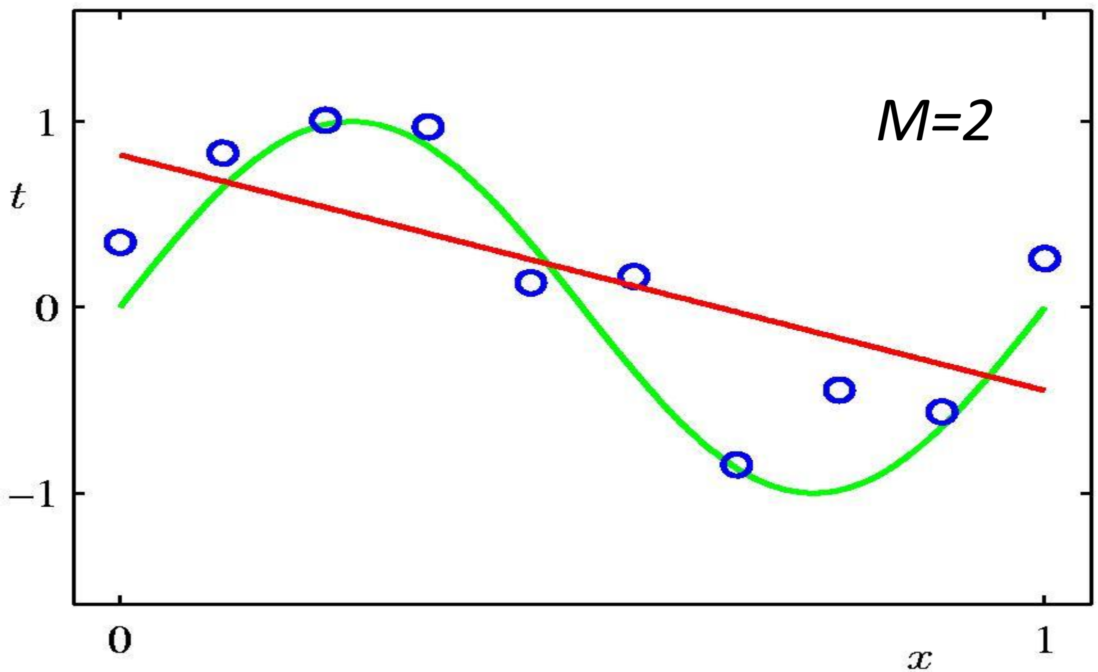


$$h(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_{M-1} x^{M-1} = \sum_{j=0}^{M-1} w_j x^j$$

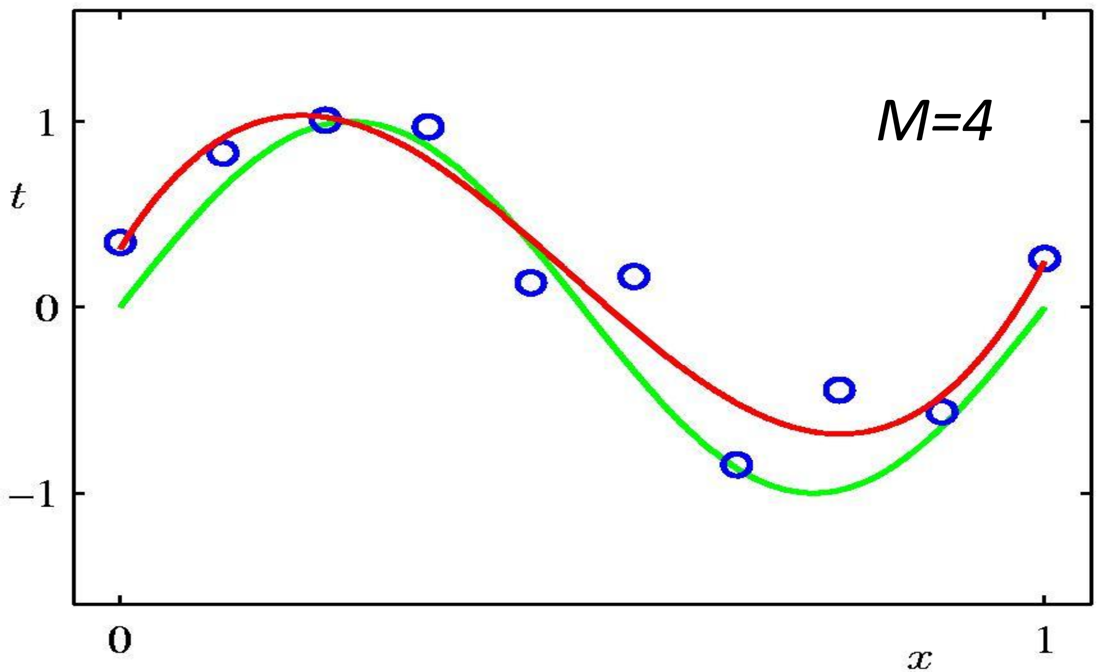
0^{th} Order Polynomial



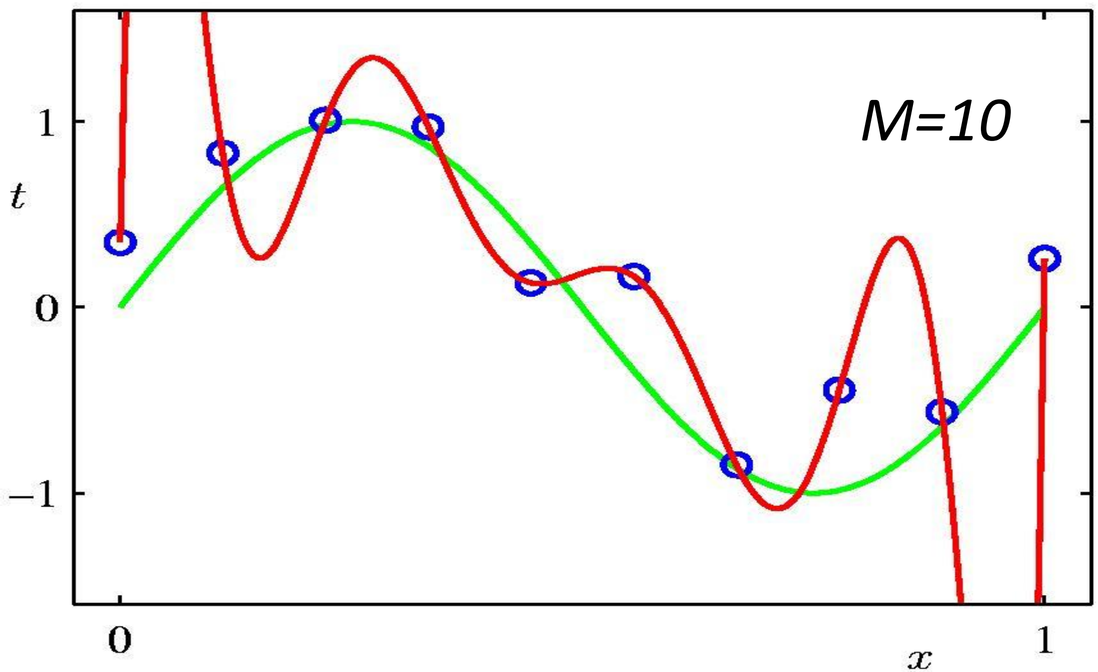
1^{st} Order Polynomial



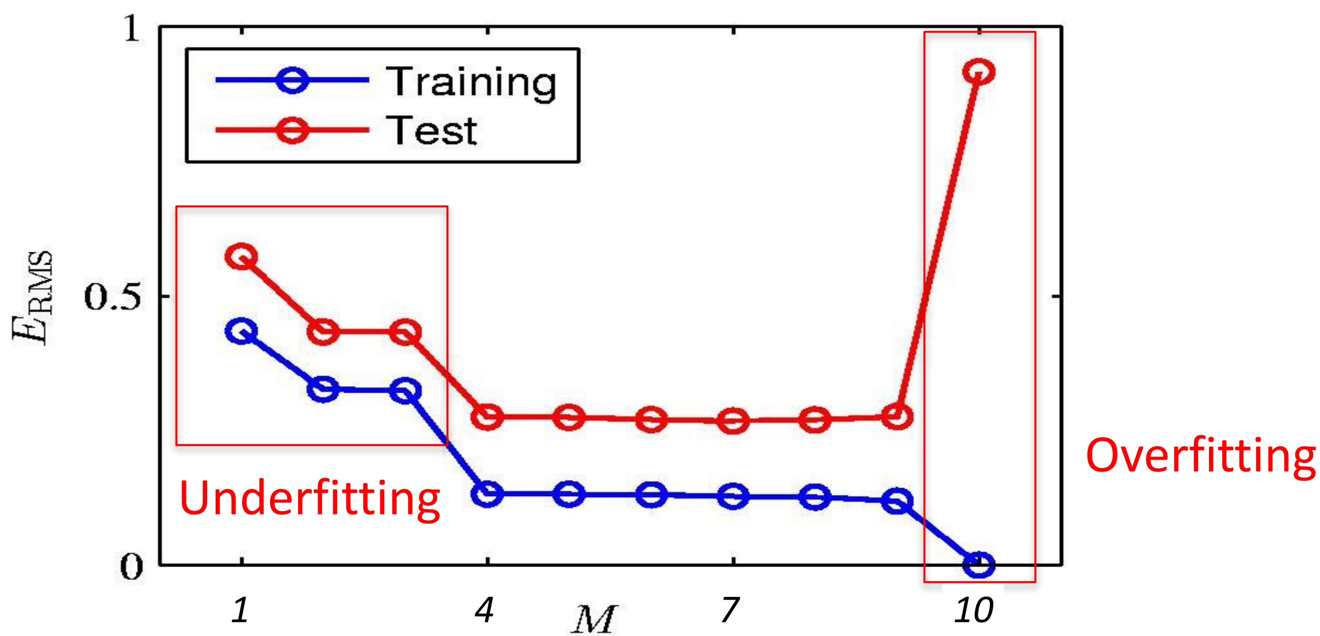
3^{rd} Order Polynomial



9th Order Polynomial



Over-fitting

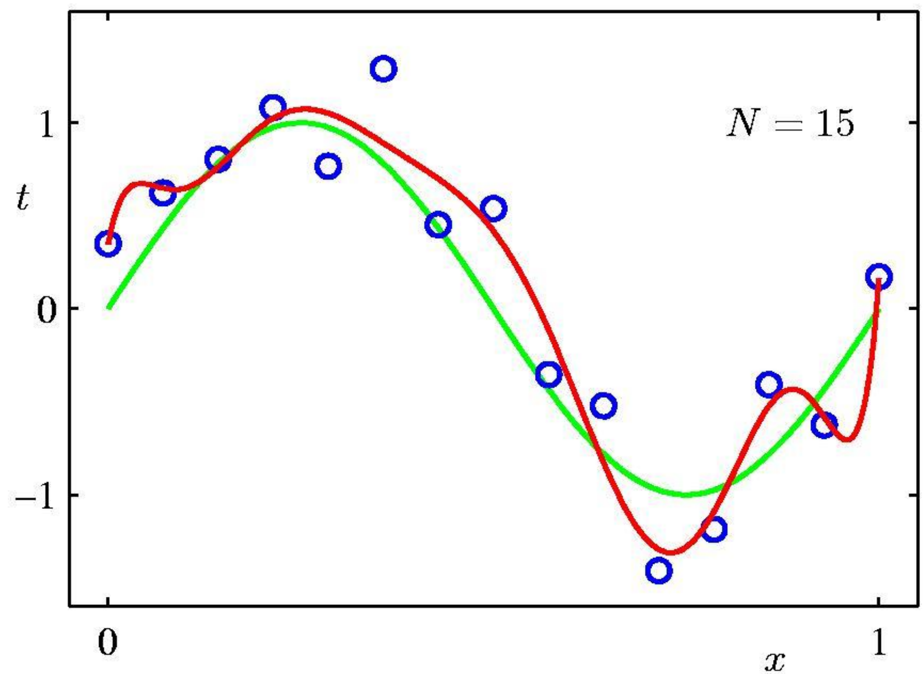


Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

Q: How do we resolve the over-fitting problem?

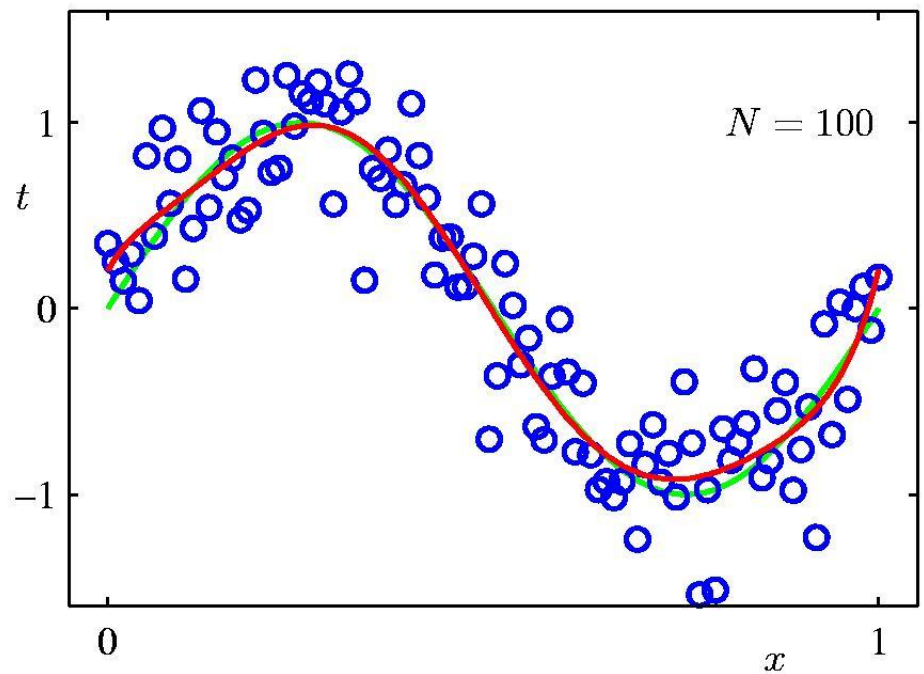
Data Set Size: $N = 15$

9th Order Polynomial



Data Set Size: $N = 100$

9th Order Polynomial



Increasing data set size can help

Q. How do we choose the degree of polynomial?

Rule of thumb

- If you have a small number of data, then use low order polynomial (small number of features).
  - Otherwise, your model will overfit
- As you obtain more data, you can gradually increase the order of the polynomial (more features).
  - However, your model is still limited by the finite amount of the data available (i.e., the optimal model for finite data cannot be infinite dimensional polynomial).
- Controlling model complexity: **regularization**

Regularized Linear Regression

Back to Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Underfitting

Good

Overfitting;

Coefficients are large!

$$h(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_{M-1} x^{M-1}$$

Regularized Least Squares (1)

- Consider the error function:

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

Data term + Regularization

λ is called the regularization coefficient.

- With the sum-of-squares error function and a quadratic regularizer, we get

Penalize large coefficient values

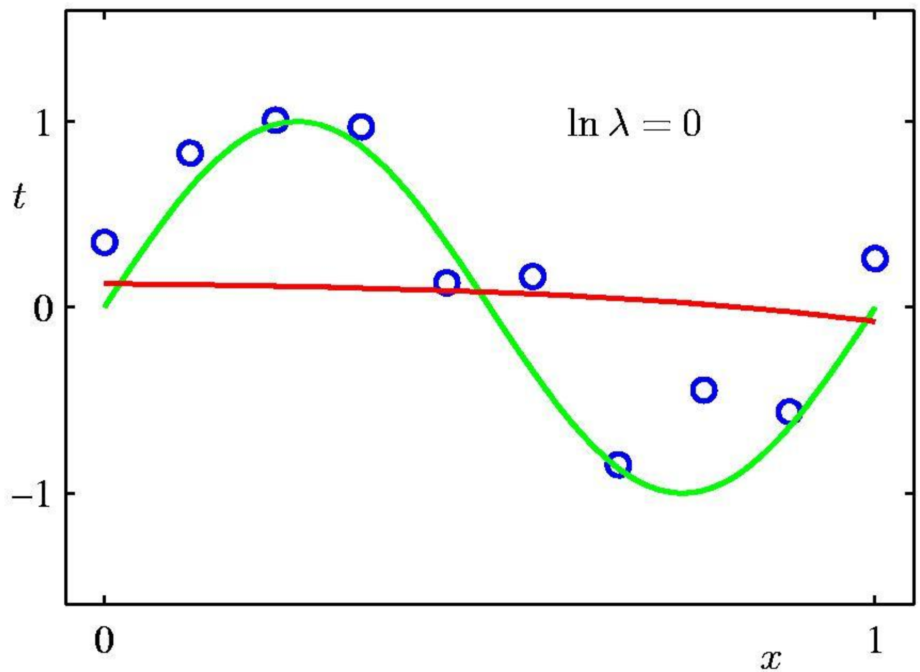
$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^\top\phi(\mathbf{x}^{(n)}) - y^{(n)}\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

New objective function

Definition (L2): $\|\mathbf{w}\|_2^2 = \sum_{j=0}^{M-1} w_j^2$

- Effect of λ

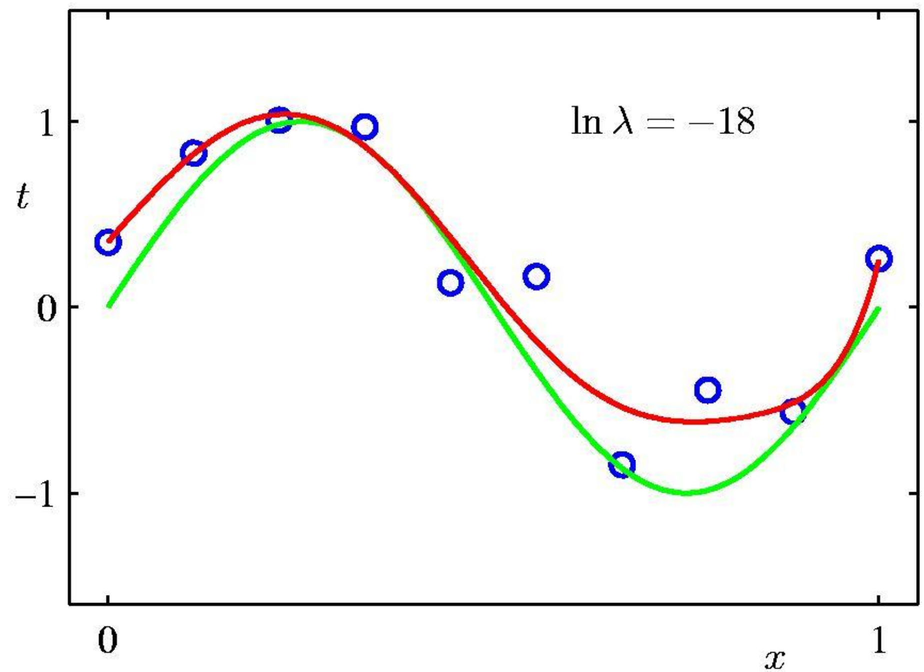
L2 Regularization: $\lambda = 1$ (too large)



$M = 9$

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^\top \phi(\mathbf{x}^{(n)}) - y^{(n)}\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

L2 Regularization: $\ln \lambda = -18$

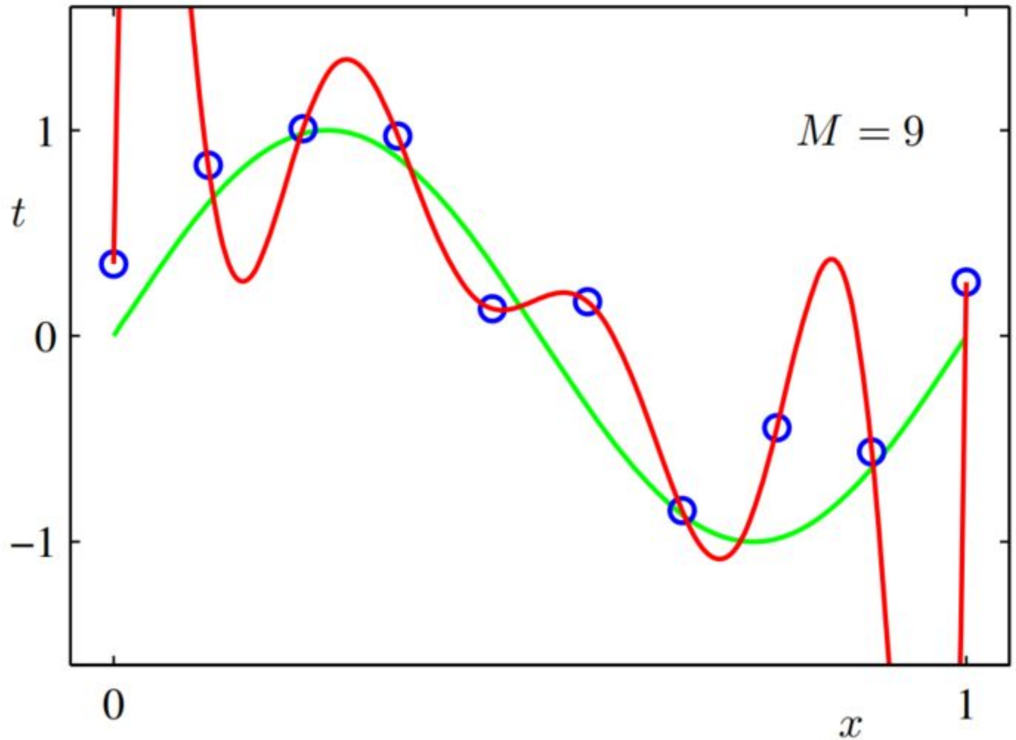


$M = 9$

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^\top \phi(\mathbf{x}^{(n)}) - y^{(n)}\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

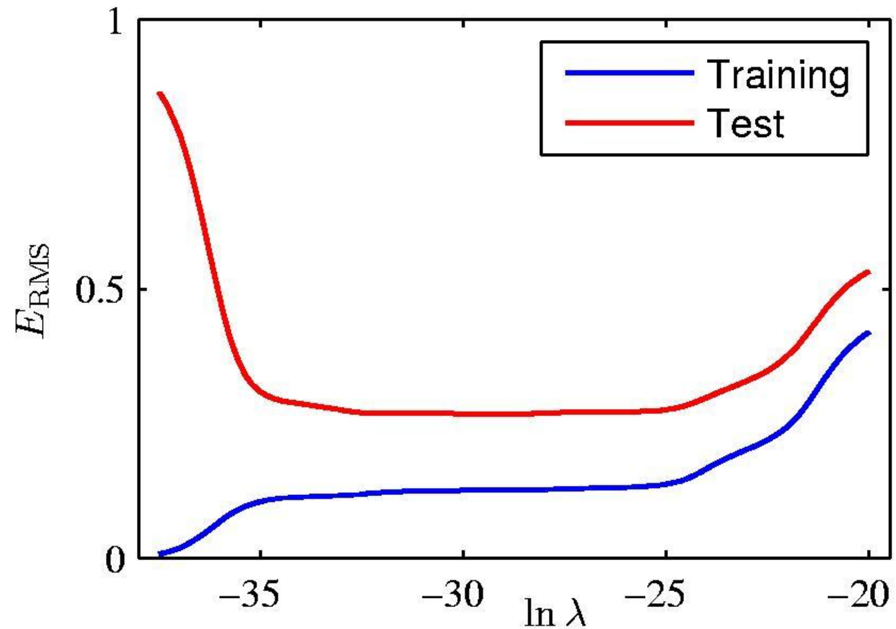
“No” L2 Regularization: $\lambda = 0$

(or when L2 regularization is too small)



$M = 9$ $$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^\top\phi(\mathbf{x}^{(n)}) - y^{(n)}\right)^2 + \cancel{\frac{\lambda}{2}\|\mathbf{w}\|_2^2}$$

L2 Regularization: E_{RMS} vs. $\ln \lambda$



Larger regularization

$$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^{\star})/N}$$

NOTE: For simplicity of presentation, we divided the data into training set and test set. However, it's **not** legitimate to find the optimal hyperparameter based on the test set. We will talk about legitimate ways of doing this when we cover model selection and cross-validation.

Polynomial Coefficients

(i.e., λ = 0)

	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Overfitting;
Coefficients are large!

Good

Underfitting

Regularized Least Squares (1)

- Consider the error function:

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

Data term + Regularization

λ is called the regularization coefficient.

- With the sum-of-squares error function and a quadratic regularizer, we get

Penalize large coefficient values

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^\top \phi(\mathbf{x}^{(n)}) - y^{(n)}\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

- Closed-form solution:

$$\mathbf{w}_{\text{reg}} = (\lambda \mathbf{I} + \Phi^\top \Phi)^{-1}\Phi^\top \mathbf{y}$$

Derivation

Objective function

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^\top \phi(\mathbf{x}^{(n)}) - y^{(n)}\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

$$= \frac{1}{2}\mathbf{w}^\top \Phi^\top \Phi \mathbf{w} - \mathbf{w}^\top \Phi^\top \mathbf{y} + \frac{1}{2}\mathbf{y}^\top \mathbf{y} + \frac{\lambda}{2}\mathbf{w}^\top \mathbf{w}$$

Compute the gradient and set it zero:

$$\nabla_{\mathbf{w}}\widetilde{E}(\mathbf{w}) = \nabla_{\mathbf{w}}\left[\frac{1}{2}\mathbf{w}^\top \Phi^\top \Phi \mathbf{w} - \mathbf{w}^\top \Phi^\top \mathbf{y} + \frac{1}{2}\mathbf{y}^\top \mathbf{y} + \frac{\lambda}{2}\mathbf{w}^\top \mathbf{w}\right]$$

$$= \Phi^\top \Phi \mathbf{w} - \Phi^\top \mathbf{y} + \lambda \mathbf{w}$$

$$= (\lambda \mathbf{I} + \Phi^\top \Phi)\mathbf{w} - \Phi^\top \mathbf{y}$$

$$= 0$$

$$\mathbf{w}_{\mathrm{ML}} = (\Phi^\top \Phi)^{-1}\Phi^\top \mathbf{y}$$

v.s. Ordinary Least Square

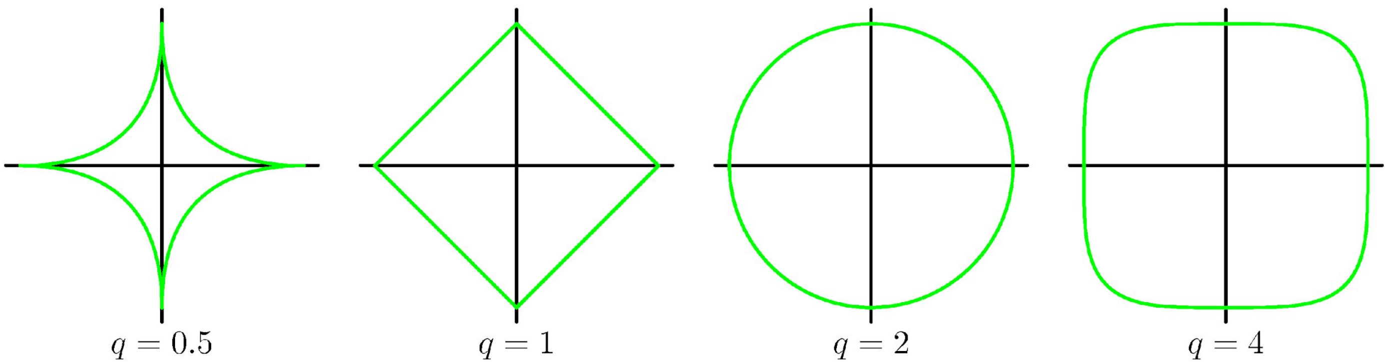
Therefore, we get: $\mathbf{w}_{\mathrm{reg}} = (\lambda \mathbf{I} + \Phi^\top \Phi)^{-1}\Phi^\top \mathbf{y}$

Regularized Least Squares (2)

- With a more general regularizer, we have

$$\frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^\top \phi(\mathbf{x}^{(n)}) - y^{(n)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

Note: In this lecture, we focus on q=2 (L2 regularization), but other values of q>0 can be used.



Lasso
"L1 regularization"

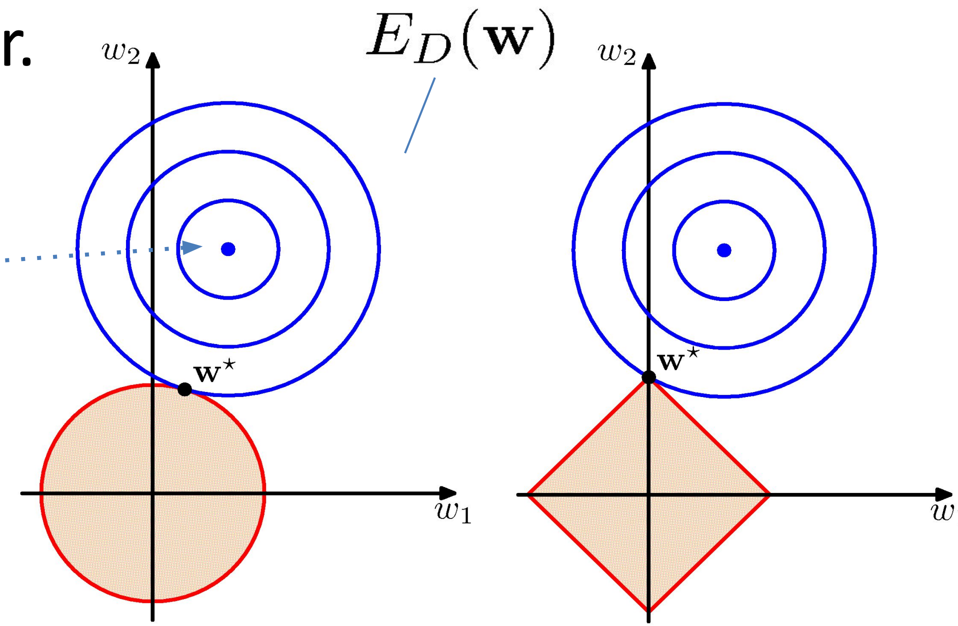
Quadratic
"L2 regularization"

plotting curves of (w_1,w_2) where $\sum_{j=1}^{M}|w_j|^q$ is a constant. (M=2)

Regularized Least Squares (3)

- Lasso tends to generate sparser solutions than a quadratic regularizer.

Ordinary least square solution



Regularization as constraints:

$$\sum_j w_j^2 \leq C \qquad \sum_j |w_j| \leq C$$

Assuming a simple scenario of isotropic data covariance, the optimal solution to L2/L1 regularization is closest point to the original solution (center of the concentric circles) that touches the boundary of the L2/L1 constraint.

Summary: Regularized Linear Regression

- Simple modification of linear regression
- Regularization controls the tradeoff between "fitting error" and "complexity"
  - Small regularization results in complex models (but with risk of overfitting)
  - Large regularization results in simple models (but with risk of underfitting)

- It is important to find an optimal regularization that balances between the two.

Maximum Likelihood interpretation of least squares regression

Review on probability

Probability: Terminology

- **Experiment**: Procedure that yields an outcome
  - E.g., Tossing a coin three times:
    - Outcome: HHH in one trial, HTH in another trial, etc.
- **Sample space**: Set of all possible outcomes in the experiment, denoted as Ω (or S)
  - E.g., for the above example:
    - Ω = {HHH, HHT, HTH, HTT, THH, THT, THT, TTH, TTT}
- **Event**: subset of the sample space Ω (i.e., an event is a set consisting of individual outcomes)
  - Event space: Collection of all events , called $\mathscr{F}$ (aka σ-algebra)
  - E.g., Event that # of heads is an even number.
    - E = {HHT, HTH, THH, TTT}
- **Probability measure**: function (mapping) from events to probability levels. I.e., $P : \mathscr{F} \longrightarrow [0, 1]$ (see next slide)
  - Probability that # of heads is an even number: 4/8 = 1/2.
- **Probability space**: $(\Omega, \mathscr{F}, P)$

Law of Total Probability

$$P(A) \geq 0, \forall A \in \mathcal{F}$$

$$P(\Omega) = 1$$

- Law of total probability

$$P(A) = P(A \cap B) + P(A \cap B^C)$$

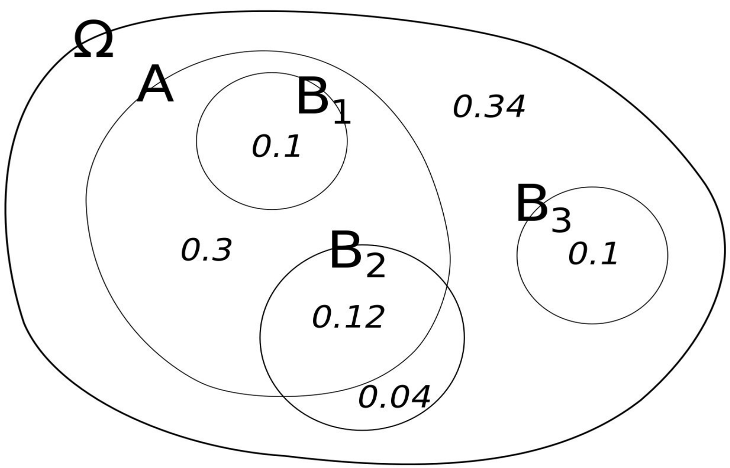
$$P(A) = \sum_i P(A \cap B_i) \quad \text{Discrete } B_i$$

$$P(A) = \int P(A \cap B_i) dB_i \quad \text{Continuous } B_i$$

Conditional Probability

For events $A, B \in \mathcal{F}$ with $P(B) > 0$, we may write the **conditional probability** of A given B:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$



From Wikipedia

$$P(A|B_1) = 1$$

$$P(A|B_2) = 0.12/(0.12 + 0.04) = 0.75$$

$$P(A|B_3) = 0$$ (disjoint)

$$P(A) = 0.30 + 0.10 + 0.12 = 0.52$$

(the unconditional probability)

Bayes' Rule

Using the chain rule we may see:

$$P(A|B)P(B) = P(A \cap B) = P(B|A)P(A)$$

Rearranging this yields **Bayes' rule:**

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

Often this is written as:

$$P(B_i|A) = \frac{P(A|B_i)P(B_i)}{\sum_i P(A|B_i)P(B_i)}$$

Where B_i are a partition of Ω (note the bottom is just the law of total probability).

Likelihood Functions

Why is Bayes' so useful in learning? Allows us to compute the posterior of **w** given data *D*:

$$p(\mathbf{w}|D) = \frac{p(D|\mathbf{w})p(\mathbf{w})}{p(D)}$$

Prior

Posterior

Bayes' rule in words: posterior ∝ likelihood × prior

$$p(\mathbf{w}|D) \propto p(D|\mathbf{w})p(\mathbf{w})$$

The likelihood function, *p*(**w** | *D*), is evaluated for observed data *D* as a function of **w**. It expresses how parameter settings **w**.

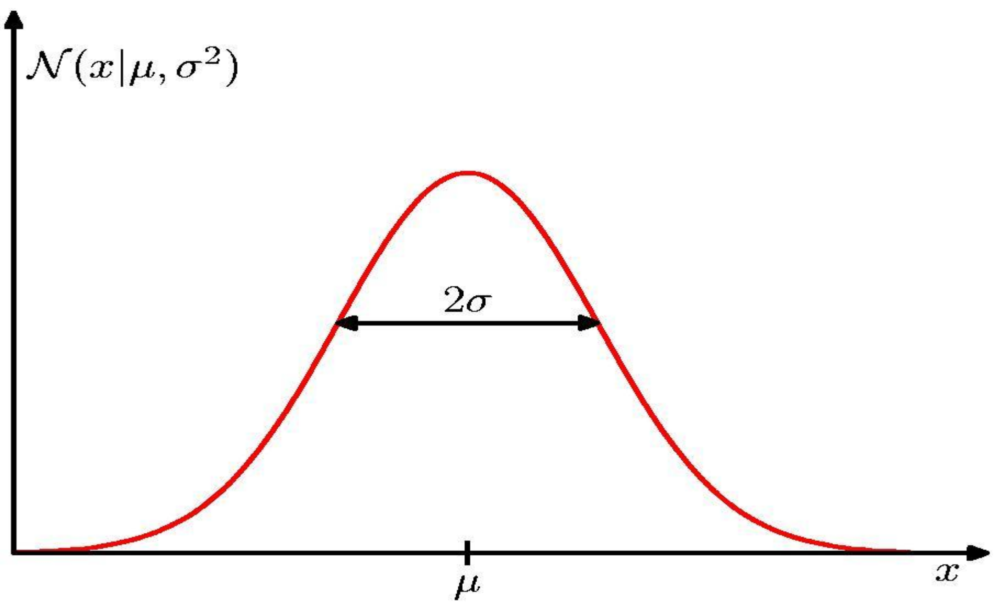
Maximum Likelihood Estimation (MLE)

- Maximum likelihood:
  - choose parameter setting **w** that maximizes likelihood function *p*(*D* | **w**).
  - choose the value of **w** that maximizes the probability of observed data.

- Cf. MAP (Maximum a posteriori) estimation
  - Equivalent to maximizing $p(\mathbf{w}|D) \propto p(D|\mathbf{w})p(\mathbf{w})$
  - Can compute this using Bayes rule!
  - This will be covered in later lectures

The Gaussian Distribution

$$\mathcal{N}\left(x|\mu,\sigma^2\right) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

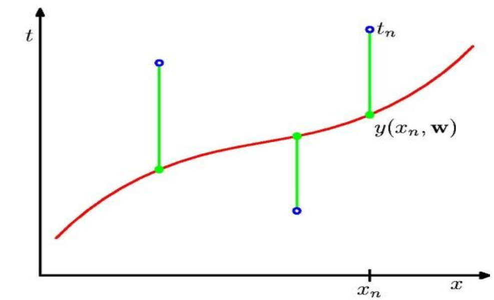


$$\mathcal{N}(x|\mu,\sigma^2) > 0$$

$$\int_{-\infty}^{\infty} \mathcal{N}\left(x|\mu,\sigma^2\right) \, \mathrm{d}x = 1$$

Maximum Likelihood interpretation of least squares regression

MLE for Linear Regression



- Assume a stochastic model:

$$y^{(n)} = \mathbf{w}^\top \phi(\mathbf{x}^{(n)}) + \epsilon \qquad \text{where } \epsilon \sim \mathcal{N}(0, \beta^{-1})$$

- This gives a likelihood function:

$$p(y^{(n)} \mid \phi(\mathbf{x}^{(n)}), \mathbf{w}, \beta) = \mathcal{N}(y^{(n)} \mid \mathbf{w}^\top \phi(\mathbf{x}^{(n)}), \beta^{-1})$$

- With input matrix $\mathbf{\Phi}$ and output matrix $\mathbf{y}$, the data likelihood is:

$$p(\mathbf{y} \mid \mathbf{\Phi}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(y^{(n)} \mid \mathbf{w}^\top \phi(\mathbf{x}^{(n)}), \beta^{-1})$$

Log-likelihood

- Given data likelihood (prev. slide)

$$p(\mathbf{y} \mid \mathbf{\Phi}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(y^{(n)} \mid \mathbf{w}^\top \phi(\mathbf{x}^{(n)}), \beta^{-1})$$

- Log likelihood:

$$\log p(\mathbf{y}|\mathbf{\Phi}, \mathbf{w}, \beta) = \frac{N}{2} \log \beta - \frac{N}{2} \log 2\pi - \beta E_D(\mathbf{w})$$

where $$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left(\mathbf{w}^\top \phi(\mathbf{x}^{(n)}) - y^{(n)} \right)^2$$

- Derivation?

Derivation of log-likelihood of p

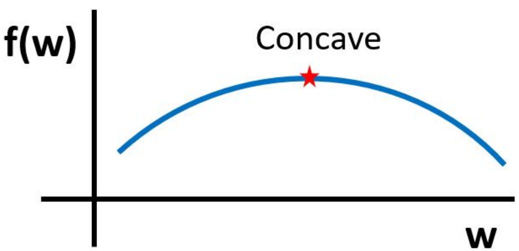
From $$p(y^{(n)} \mid \phi(\mathbf{x}^{(n)}), \mathbf{w}, \beta) = \mathcal{N}(y^{(n)} \mid \mathbf{w}^\top \phi(\mathbf{x}^{(n)}), \beta^{-1})$$
$$= \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2} \left\| y^{(n)} - \mathbf{w}^\top \phi(\mathbf{x}^{(n)}) \right\|^2\right)$$

Derive: $$\log p(y^{(1)}, y^{(2)}, \ldots, y^{(N)} \mid \mathbf{\Phi}, \mathbf{w}, \beta)$$
$$= \log \prod_{n=1}^{N} \mathcal{N}\left(y^{(n)} \mid \mathbf{w}^\top \phi(\mathbf{x}^{(n)}), \beta^{-1}\right)$$
$$= \sum_{n=1}^{N} \log\left(\sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2} \left\| y^{(n)} - \mathbf{w}^\top \phi(\mathbf{x}^{(n)}) \right\|^2\right)\right)$$
$$= \sum_{n=1}^{N} \left(\frac{1}{2}\log\beta - \frac{1}{2}\log 2\pi - \frac{\beta}{2} \left\| y^{(n)} - \mathbf{w}^\top \phi(\mathbf{x}^{(n)}) \right\|^2\right)$$
$$= \frac{N}{2}\log\beta - \frac{N}{2}\log 2\pi - \sum_{n=1}^{N} \frac{\beta}{2} \left\| y^{(n)} - \mathbf{w}^\top \phi(\mathbf{x}^{(n)}) \right\|^2$$

Maximum likelihood estimation (MLE)

- Let's maximize the log-likelihood!
- Set the gradient of log-likelihood = 0 (Why?)

$$\nabla_{\mathbf{w}} \log p(y|\mathbf{\Phi}, \mathbf{w}, \beta) = \nabla_{\mathbf{w}} \left(\underbrace{\frac{N}{2}\log\beta - \frac{N}{2}\log 2\pi}_{\text{Constant}} - \sum_{n=1}^{N} \frac{\beta}{2} \left\| y^{(n)} - \mathbf{w}^\top \phi(\mathbf{x}^{(n)}) \right\|^2 \right)$$



$$= \beta \sum_{n=1}^{N} \left(y^{(n)} - \underbrace{\mathbf{w}^\top \phi(\mathbf{x}^{(n)})}_{\text{Scalar}} \phi(\mathbf{x}^{(n)}) \right)$$

$$= \beta \left(\sum_{n=1}^{N} y^{(n)} \phi(\mathbf{x}^{(n)}) - \phi(\mathbf{x}^{(n)}) \phi(\mathbf{x}^{(n)})^\top \mathbf{w} \right) = 0$$

- In matrix form, $\beta(\Phi^\top \mathbf{y} - \Phi^\top \Phi \mathbf{w}) = 0$

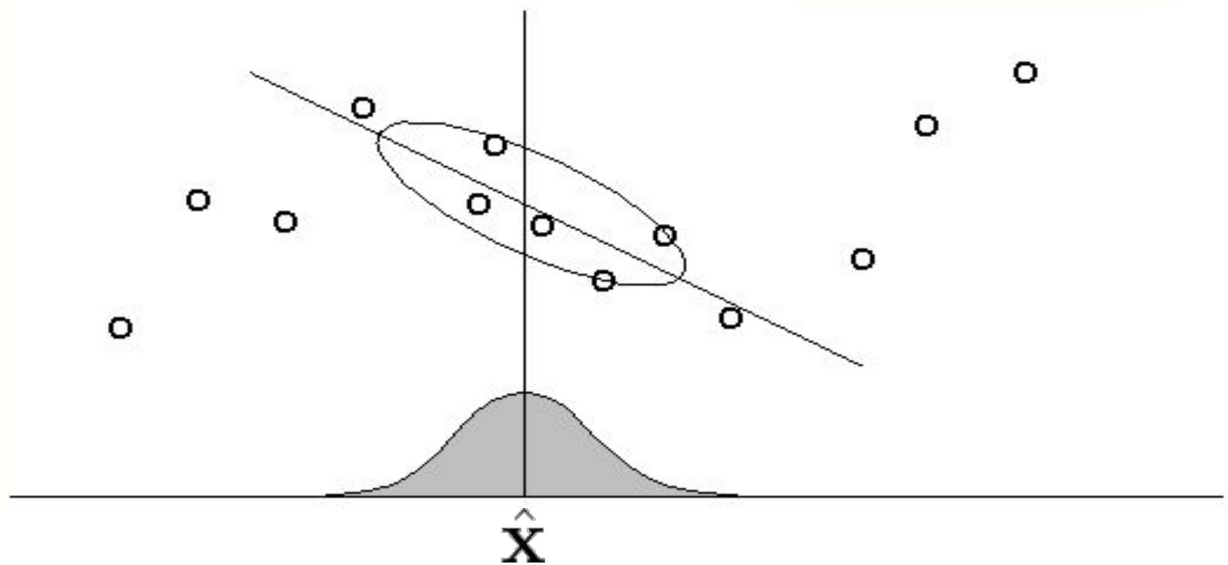
$$\mathbf{w}_{\text{ML}} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y}$$

- MLE solution is equivalent to OLS solution!

Locally-weighted Linear Regression

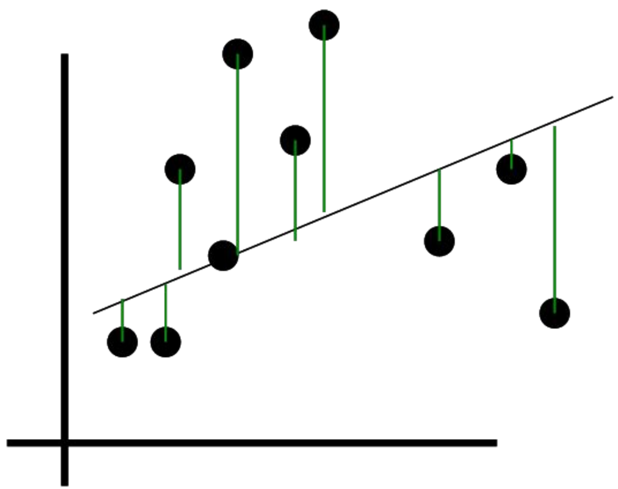
Locally weighted linear regression

- Main idea: When predicting $f(\hat{\mathrm{x}})$, give high weights for "neighbors" of $\hat{\mathrm{x}}$.



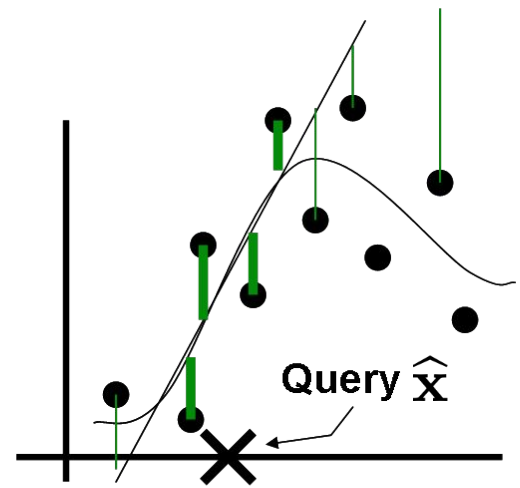
In locally weighted regression, points are weighted by proximity to the current $\hat{\mathrm{x}}$ in question using a kernel. A regression is then computed using the weighted points.

Regular linear regression vs. locally weighted linear regression



Regular linear regression

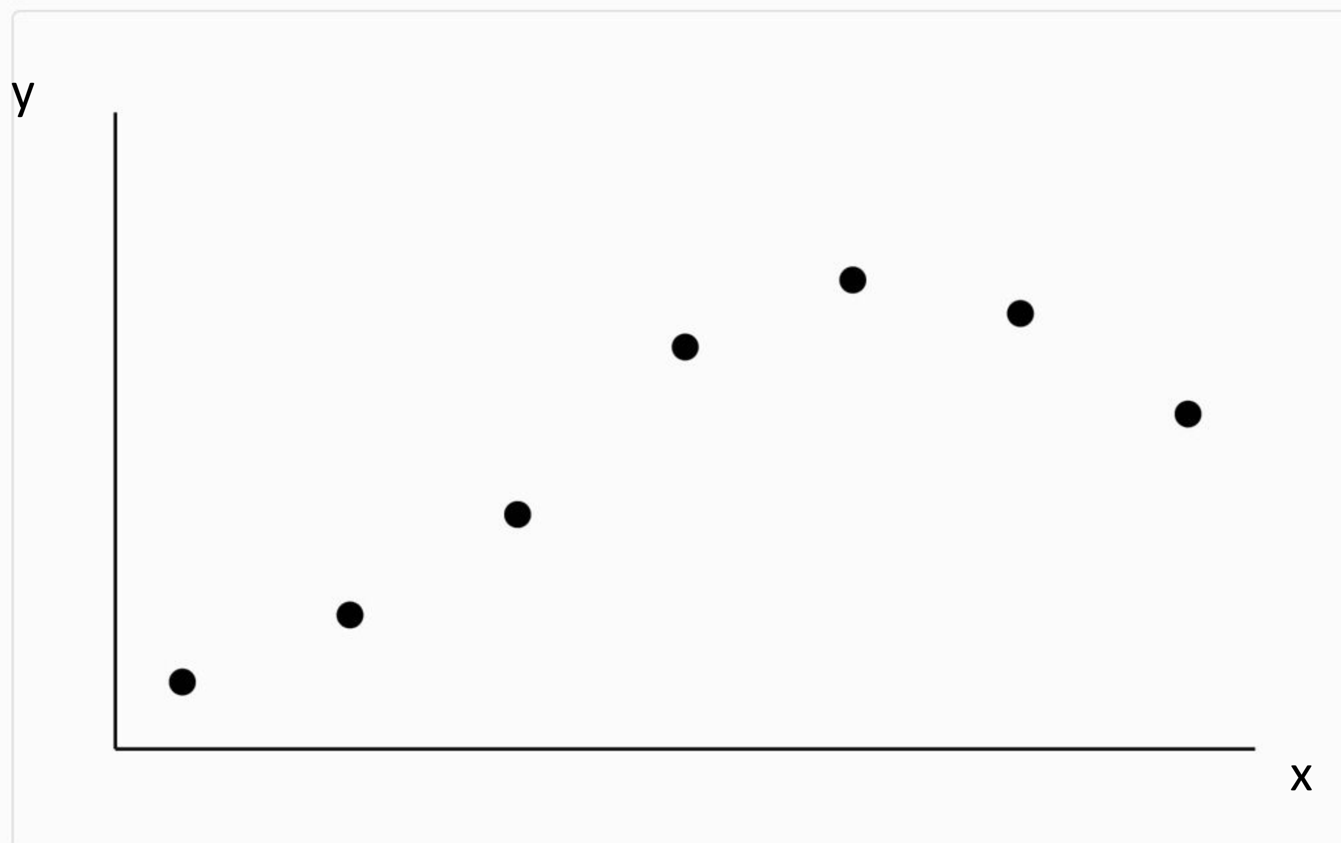
$$\sum_{n=1}^{N} \left(\mathbf{w}^\top \phi(\mathbf{x}^{(n)}) - y^{(n)} \right)^2$$



Locally weighted linear regression

$$\sum_{n=1}^{N} r^{(n)}(\widehat{\mathbf{x}}) \left(\mathbf{w}^\top \phi(\mathbf{x}^{(n)}) - y^{(n)} \right)^2$$

locally weighted linear regression



Step 1: Start with training data points

Linear regression vs. Locally-weighted Linear Regression

- A query point $\widehat{\mathbf{x}}$, training set $\left\{\left(\mathbf{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}$
- Linear regression
  1. Fit $\mathbf{w}$ to minimize $\sum_{n=1}^{N}\left(\mathbf{w}^\top\phi(\mathbf{x}^{(n)}) - y^{(n)}\right)^2$
  2. Predict $\mathbf{w}^\top\phi(\widehat{\mathbf{x}})$
- Locally-weighted linear regression
  1. Fit $\mathbf{w}$ to minimize $\sum_{n=1}^{N} r^{(n)}(\widehat{\mathbf{x}})\left(\mathbf{w}^\top\phi(\mathbf{x}^{(n)}) - y^{(n)}\right)^2$
  2. Predict $\mathbf{w}^\top\phi(\widehat{\mathbf{x}})$

weights are dependent on the query $\widehat{\mathbf{x}}$
(i.e., need to solve the optimization for each query value)

Linear regression vs. Locally-weighted Linear Regression

- Locally-weighted linear regression
  1. Fit $\mathbf{w}$ to minimize $\sum_{n=1}^{N} r^{(n)}(\widehat{\mathbf{x}}) \left(\mathbf{w}^\top \phi(\mathbf{x}^{(n)}) - y^{(n)}\right)^2$
  2. Predict $\mathbf{w}^\top \phi(\widehat{\mathbf{x}})$

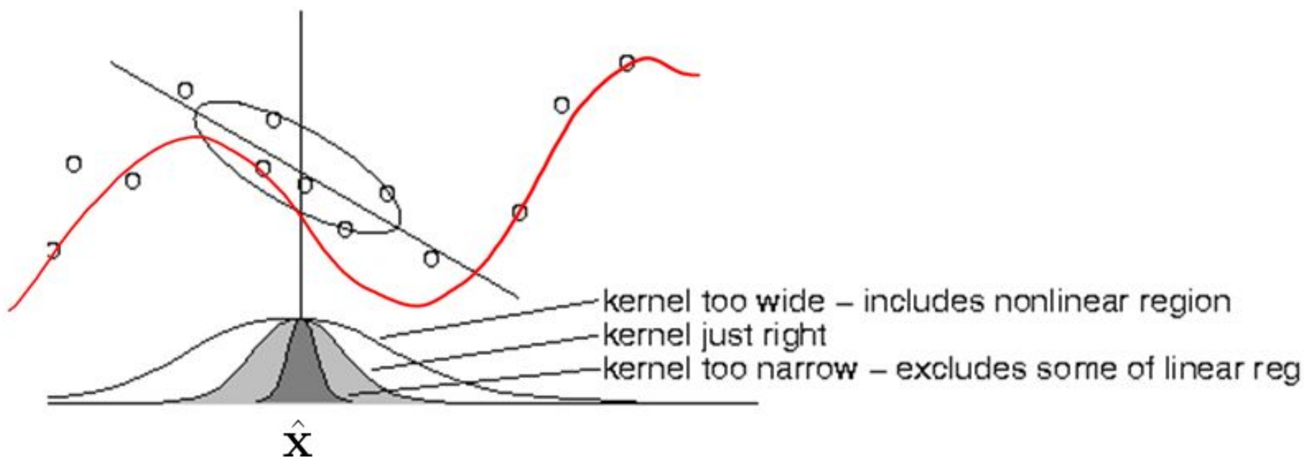
- Remarks:

  Gaussian kernel with kernel width τ

  1. Standard choice: $r^{(n)}(\widehat{\mathbf{x}}) = \exp\left(-\frac{\left\|\phi(\mathbf{x}^{(n)}) - \phi\left(\widehat{\mathbf{x}}\right)\right\|^2}{2\tau^2}\right)$
  2. Note that $r^{(n)}(\widehat{\mathbf{x}})$ depends on $\widehat{\mathbf{x}}$ (query point), and you solve linear regression for each query point $\widehat{\mathbf{x}}$
  3. The problem can be formulated as a modified version of least squares problem (HW#1)

Locally weighted linear regression

- Choice of kernel width τ matters
  - Requires hyper-parameter tuning



The estimator is minimized when kernel includes as many training points as can be accommodated by the model. Too large a kernel includes points that degrade the fit; too small a kernel neglects points that increase confidence in the fit.

Summary

- L_2 Regularized linear regression $\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^\top \phi(\mathbf{x}^{(n)}) - y^{(n)}\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$
  - Adding L_2 regularizer
  - Can be solved via closed form (simple modification of the original linear regression)
  - penalizes complex solutions (with high weights)

- Maximum likelihood interpretation of linear regression
  - Linear regression can be interpreted as performing MLE assuming the Gaussian noise distribution for targets

- Locally-weighted linear regression

Any feedback (about lecture, slide, homework, project, etc.)?

(via **anonymous** google form: https://forms.gle/fpYmiBtG9Me5qbP37)

Change Log of lecture slides:
https://docs.google.com/document/d/e/2PACX-1vS6WIDIvW16-DZF8Ja7SpbMYK732Cy62xgNfr5kS-dt34fhMr8RD-6xvApS41Vmqi2y5wToIk_WI3SB/pub