Machine Learning

INFQ612L, 440113450A

Spring Semester

Friday 17:00–18:40

IPS

WASEDA University

Prof. Shoji Makino

Machine Learning

Friday 17:00–18:40

1. 4/18
2. 4/25
3. 5/2
4. 5/9
5. 5/16
6. 5/23
7. 5/30
8. 6/6
9. 6/13
10. 6/20
11. 6/27

–. 7/4 No Lecture

–. 7/11 No Lecture

–. 7/18 No Lecture

At Zoom, set your name as:

Student ID, LAST_NAME, First_name

44251234, MAKINO, Shoji

At my class,

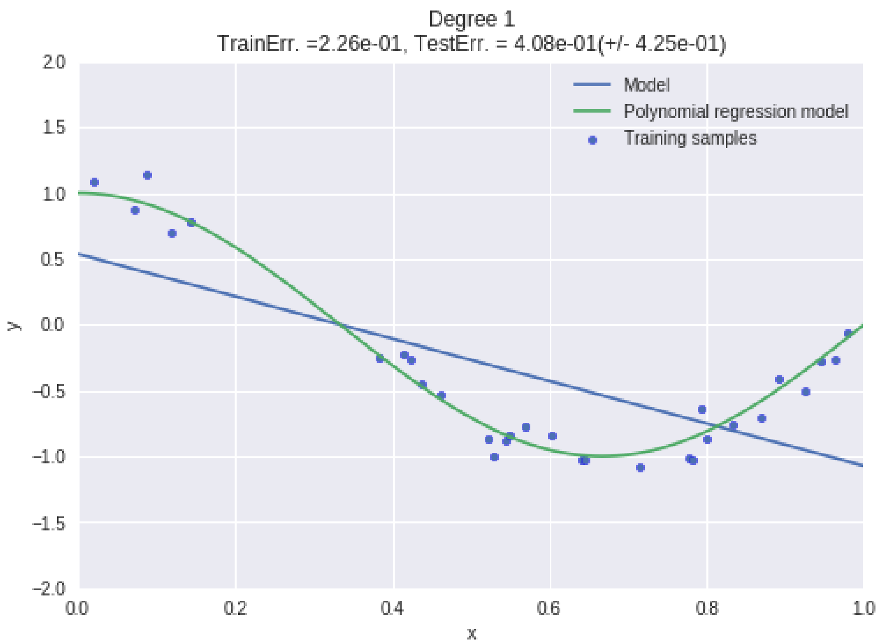
please turn on your camera

Machine Learning (6)(7)

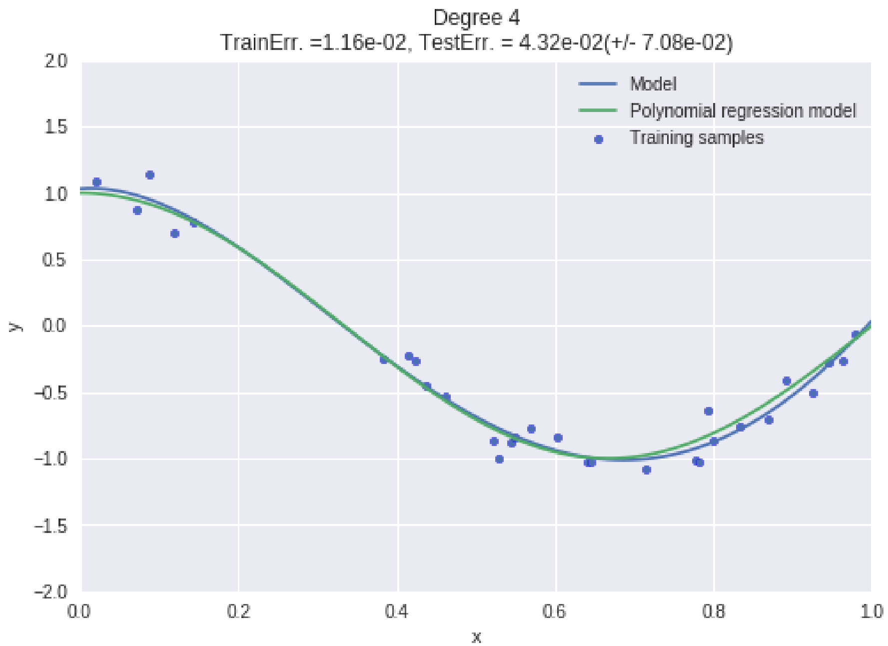
Model complexity
and
Generalization

Data Nonlinearity

- If the data distribution is nonlinear?
- We want to make predictions with higher representation ability than the linear model
(green line: correct model, blue line: learned model)



Single regression



Polynomial regression
(Single regression by polynomial features)

Polynomial Regression

- Linear regression model
  - 1D features $t = w_0 + w_1 x$
  - D-dimensional features

$$t = w_0 + w_1 x_1 + \ldots + w_D x_D = w_0 + \sum_{d=1}^{D} w_d x_d$$

- M-th order polynomial regression model
  - 1D features $t = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$
  - D-dimensional features

$$t = w_0 + \sum_{m=1}^{M} \sum_{i=d}^{D} w_{md} x_d^m$$

Regression by One-Dimensional Polynomial Features

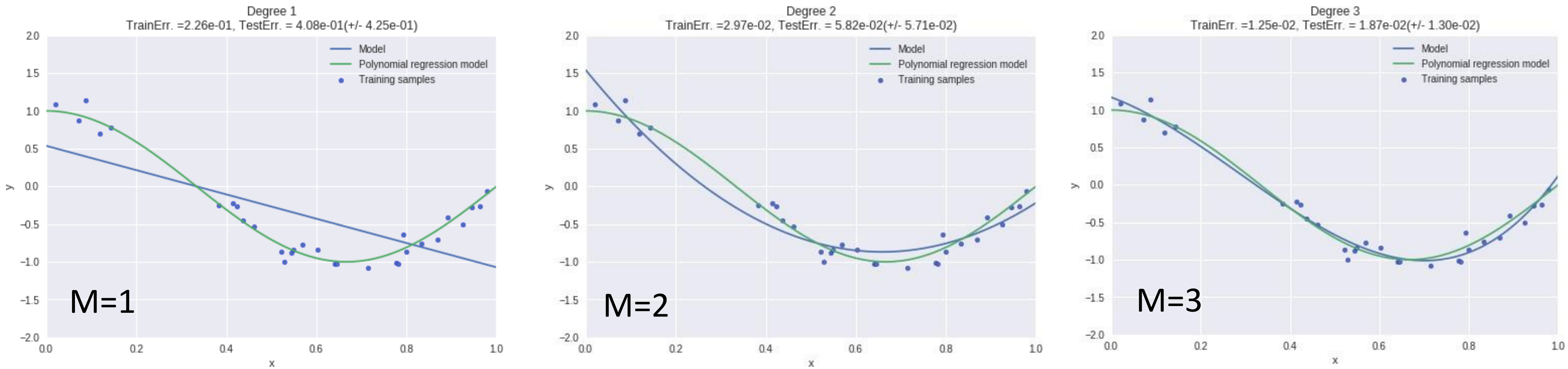
- Feature function
  - Polynomial features

$$\phi : \mathbb{R} \rightarrow \mathbb{R}^{M+1}$$

$$\phi(x) = (x^0, x^1, x^2, \ldots, x^M)$$

- Linear regression model with polynomial features

$$t = \boldsymbol{w}^T \boldsymbol{\phi}(x)$$



Multidimensional Polynomial Regression

- Feature vector ⇒ Feature vector (by ϕ)

$$\boldsymbol{x}^T = (1, 2, 5) \Rightarrow \phi(\boldsymbol{x})^T = \left(1, 2, 5, 1^2, 2^2, 5^2, 1^3, 2^3, 5^3\right)$$

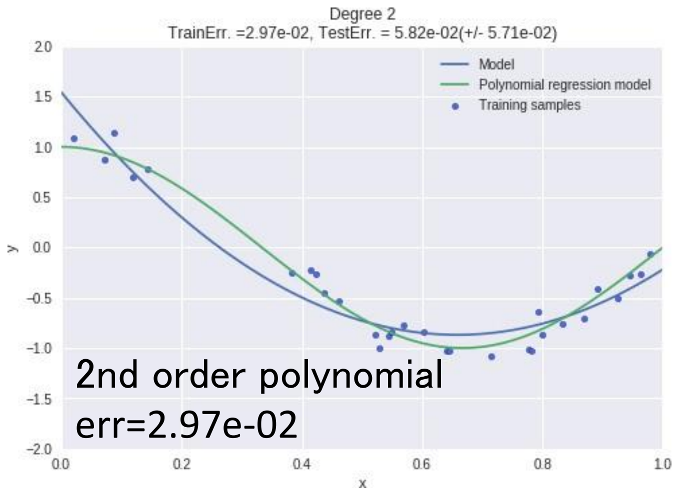
- Regression with feature vector ϕ instead of $\boldsymbol{x}$

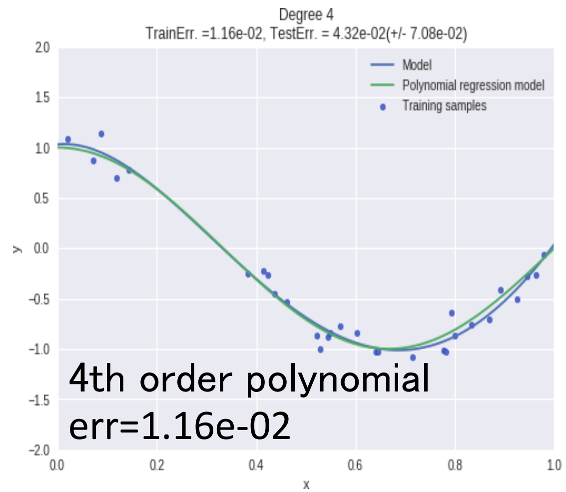
$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_N^T \end{pmatrix} \qquad \boldsymbol{\Phi} = \begin{pmatrix} \phi(\boldsymbol{x}_1)^T \\ \phi(\boldsymbol{x}_2)^T \\ \vdots \\ \phi(\boldsymbol{x}_N)^T \end{pmatrix}$$

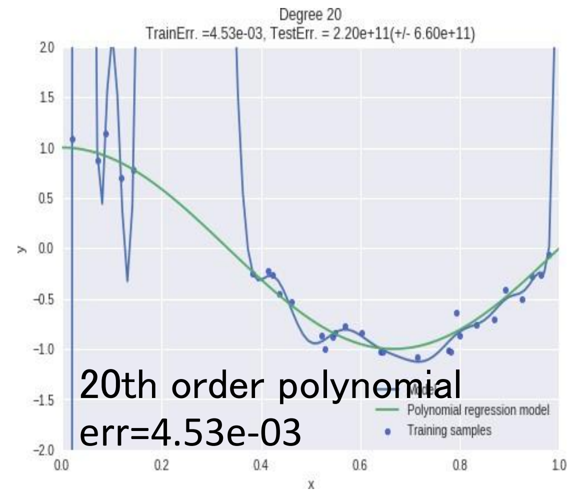
$$\boldsymbol{w} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{t} \qquad \boldsymbol{w} = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{t}$$

Model Complexity

- Introduction of polynomial features ⇒ Realization of rich representation regression
- How rich should it be?



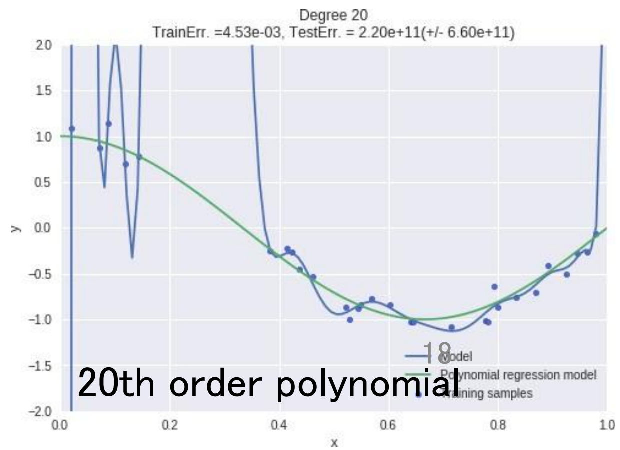




- Squared error of 20th order polynomial regression is smaller than that of 4th order polynomial regression
- But I don't think this is the best
- What should we do?

Occam's Razor

- You should not assume more than you need to explain something (14th century philosopher Occam)
  - Entities should not be multiplied beyond necessity
  - (In terms of ML …) Given some data, the more complex the model, the better it can be explained. However, such a model is an unnecessarily complex model that is not only difficult to calculate, but also overfits past data, making it impossible to explain future data.
- How to choose the right complexity?



20th order polynomial

Review Statistics

Probability Variables, Probability Distribution

- Probability variable: variable whose value is determined by a certain probability law
- Probability distribution: For each value of a probability variable, the likelihood of that value (probability)
- Examples of discrete probability variable x : coin back (= 0), front (= 1)
  - $P(x = 0) = 0.5$, $P(x = 1) = 0.5$
  - probability variable x follows the Bernoulli distribution with $p = 0.5$

$$f(x;p) = p^x(1-p)^{(1-x)}$$

- Example of a continuous probability variable: Distance from a fallen leaf x

$$P(1 \leq x \leq 2) = \int_1^2 N(x;0,1)$$

  - $N(x; 0,1)$ is a normal distribution with mean 0 and variance 1 (Probability density function)
  - Probability variable x follows normal distribution $N(x;0,1)$

Review
Statistics

Expected Value

- Expected value: value of a probability variable weighted and averaged by its probability *p(x)*

discrete probability variable *X*

$$E[X] = \sum_{i=1}^{\infty} x_i P(X = x_i)$$

continuous probability variable *x*

$$E[x] = \int_{-\infty}^{\infty} x p(x) dx$$

- If an infinite number of samples are obtained, the expected value can be calculated
- However, usually only a finite number of samples are available
- Use sample mean from finite samples instead of expected value

$$\bar{X}_N = \frac{1}{N} \sum_{i=1}^{N} x_i$$

- Sample mean of randomly sampled samples from a population approaches the expected value of the population as the number of samples increases (the law of large numbers)

$$\epsilon > 0, \ \lim_{N \to \infty} \Pr[|\bar{X}_N - E[X]| > \epsilon] = 0$$

What are we trying to know?

- In the world …
  - There is a distribution $p(\boldsymbol{x}, t)$ that produces the data
    - Distribution of wine ingredients $\boldsymbol{x}$ and quality of the wine t
  - Model exists $f : \mathbb{R}^D \to \mathbb{R}$
    - Map of wine ingredients $\boldsymbol{x}$ to wine quality t
  - But these are all unknown

- What we can observe is …
  - Wine ingredients $\boldsymbol{x}_i$ and quality samples t_i of the wine

    $$(\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), \ldots, (\boldsymbol{x}_N, t_N),$$
  - Assumption: quality has (unknown) noise $t_i = f(\boldsymbol{x}_i) + \epsilon$

- Under this condition, we want to estimate a model $\hat{f}$ that is close to f

Training Samples and Test Samples

- N samples on hand $(\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), \ldots, (\boldsymbol{x}_N, t_N),$
- Very accurate but meaningless learning
  - Build a hash function that returns t_i given x_i
  - Square error is always zero
  - However, we cannot respond to inquiries other than the samples we have

- Divide samples into training samples and test samples

$$\underbrace{(\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), .}_{\text{Training samples } X_{\mathrm{tr}}} \underbrace{.., (\boldsymbol{x}_N, t_N)}_{\text{Test samples } X_{\mathrm{ts}}},$$

– Learn with training samples and check performance with test samples

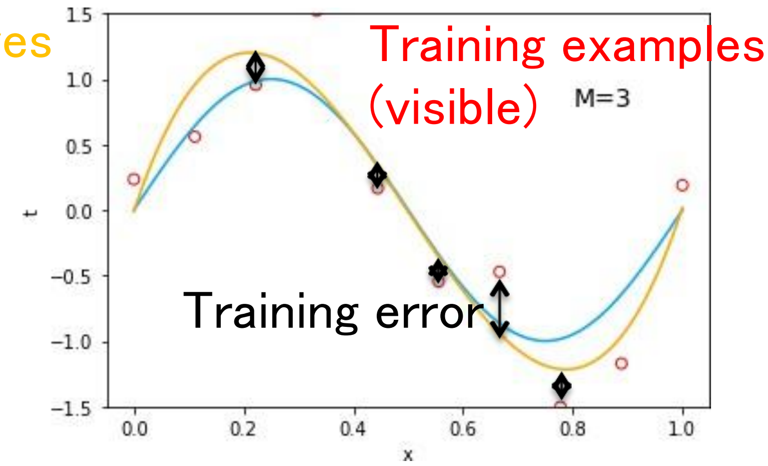
– Training samples = Exercise with answer (100 points can be obtained by memorizing)

– Test samples = Final exam (Check understanding with questions different from exercises)

Training Error

Learned regression curves (visible)

True regression curve (invisible)



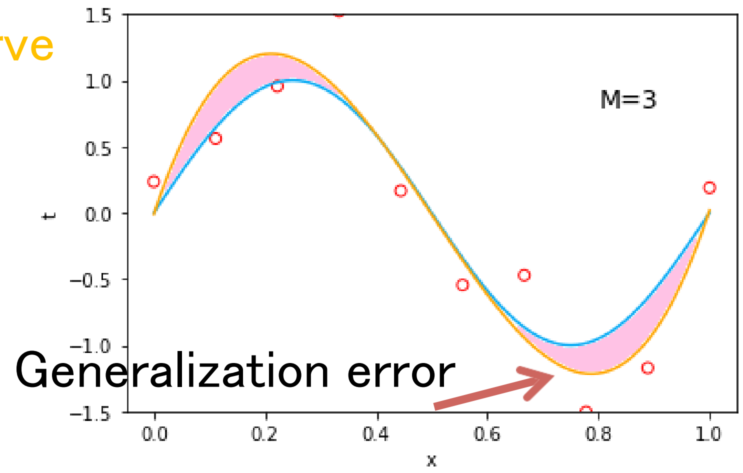
- Model: $t = f(\boldsymbol{x})$ We want to know
- Training examples: $X_{tr} = \{(\boldsymbol{x}_i, t_i)\}$ visible
- Learned regression models: $\hat{t}_i = \hat{f}(\boldsymbol{x}_i)$ visible
- Training error (= what was previously called squared error)
  - Incorrect answer rate in the training samples of the regression model obtained in the training samples (incorrect answer rate of the Exercise with answer)
  - Note that divide by the number of training samples

$$\mathrm{TrainingErr} = \frac{1}{|X_{\mathrm{tr}}|} \sum_{(\boldsymbol{x}_i, t_i) \in X_{\mathrm{tr}}} (\underbrace{t_i}_{\text{Training sample target value}} - \underbrace{\hat{f}(\boldsymbol{x}_i)}_{\text{Prediction}})^2$$

Generalization Error

Learned regression curve (visible)

True regression curve (invisible)

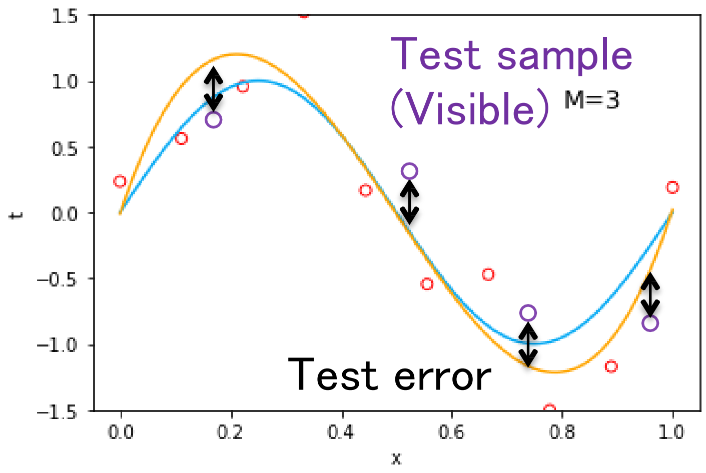


- Generalization error :
  - $p(\boldsymbol{x}, t)$ data generation distribution, invisible
  - True model (target): $t = f(\boldsymbol{x})$ invisible
  - Learned regression model: $\hat{t}_i = \hat{f}(\boldsymbol{x}_i)$ visible

$$\mathrm{GeneralizationErr} = \iint (\underbrace{t}_{\text{Correct (unknown)}} - \underbrace{\hat{f}(\boldsymbol{x})}_{\text{Prediction}})^2 \underbrace{p(\boldsymbol{x}, t)}_{\text{Data generation distribution (unknown)}} d\boldsymbol{x}dt$$

- We really want this expected value
- However, we cannot evaluate it because we cannot get $p(\boldsymbol{x}, t), f(\boldsymbol{x})$

Test Error



- Model: $t = f(\boldsymbol{x})$ We want to know
- Test samples: $X_{ts} = \{(\boldsymbol{x}_i, t_i)\}$ visible
- Learned regression model: $\hat{t}_i = \hat{f}(\boldsymbol{x}_i)$ visible
- Test error:
  - Incorrect answer rate in test samples of regression model learned in training samples (incorrect answer rate of Final exam)
  - Note that it is divided by the number of test samples

$$\mathrm{TestErr} = \underbrace{\frac{1}{|X_{\mathrm{ts}}|} \sum_{(\boldsymbol{x}_i, t_i) \in X_{\mathrm{ts}}}}_{\text{Finite sample average}} (\underbrace{t_i}_{\text{Test sample target value}} - \underbrace{\hat{f}(\boldsymbol{x}_i)}_{\text{Prediction}})^2$$

Finite sample average   Test sample target value   Prediction

Evaluating Errors

We have | We don't have

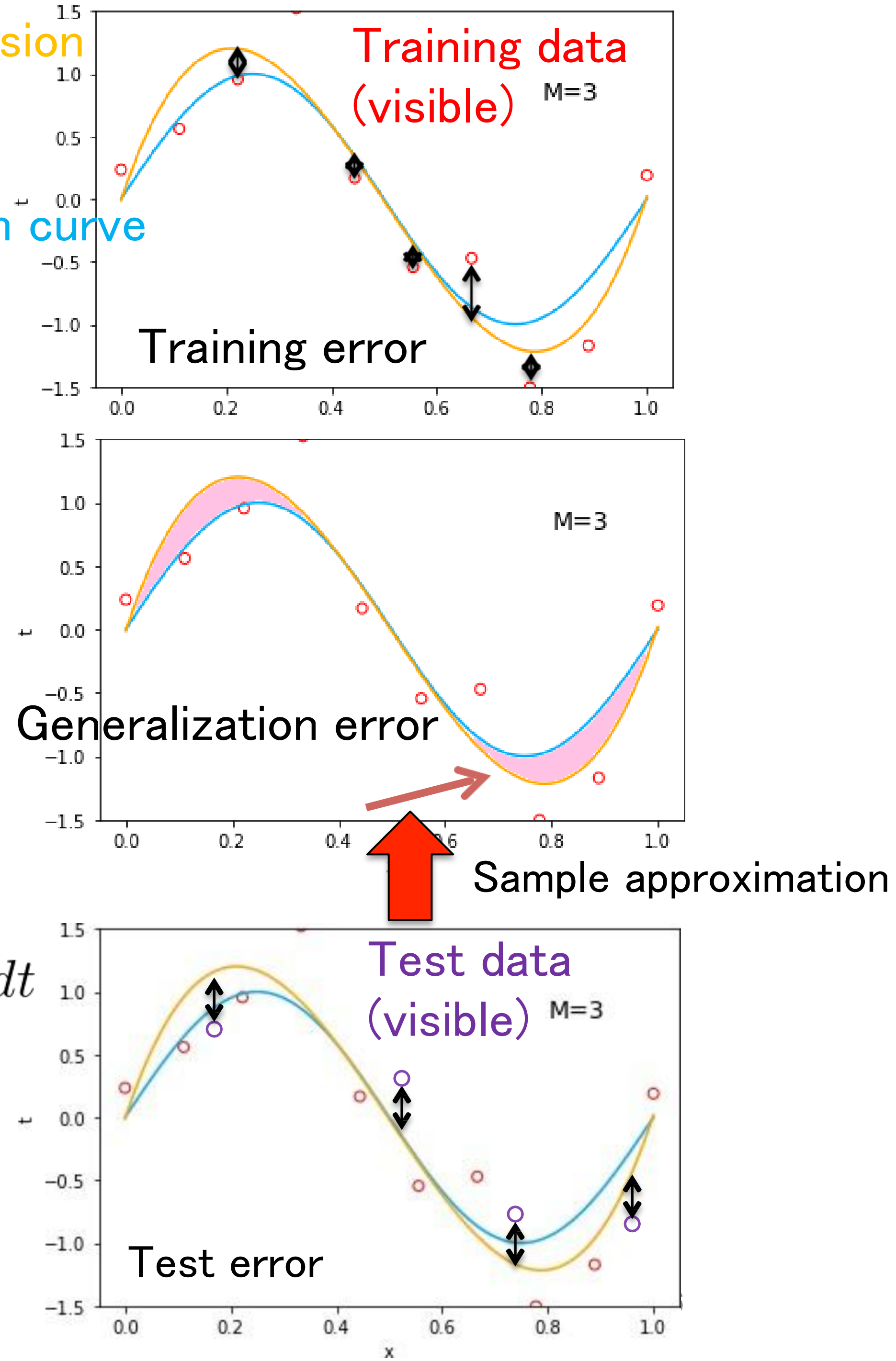
Data	Data
Target value	Target value

Training error | Test error

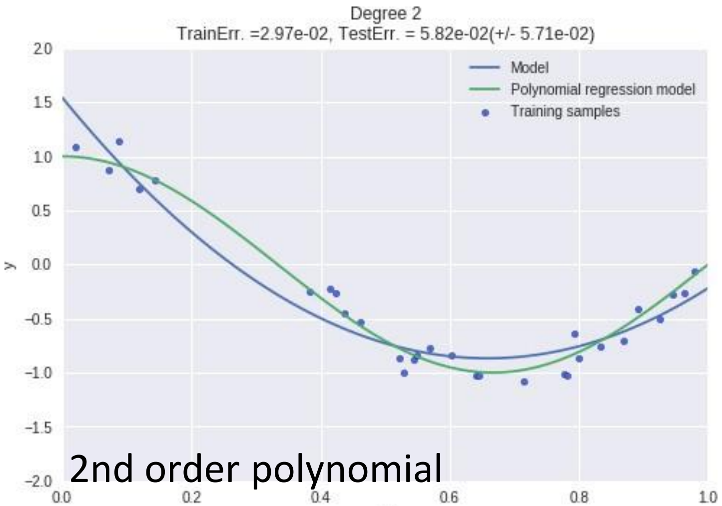
$$\mathrm{GeneralizationErr} = \iint (t - \hat{f}(\boldsymbol{x}))^2 p(\boldsymbol{x}, t) d\boldsymbol{x} dt$$

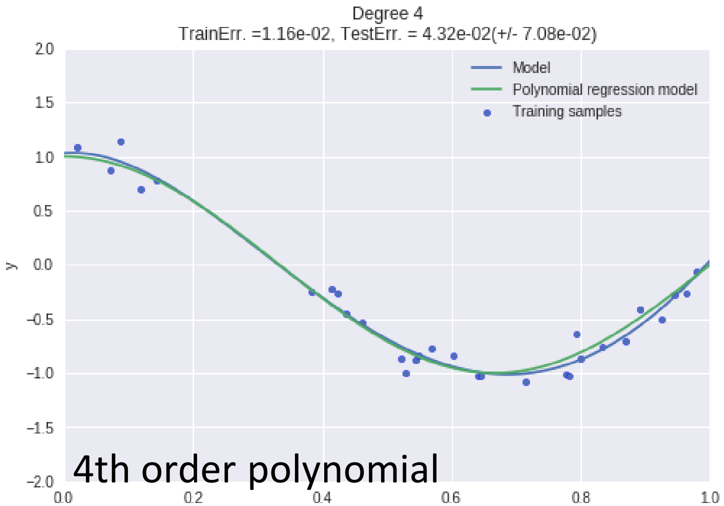
Finite sample approximation of generalization error = Test error

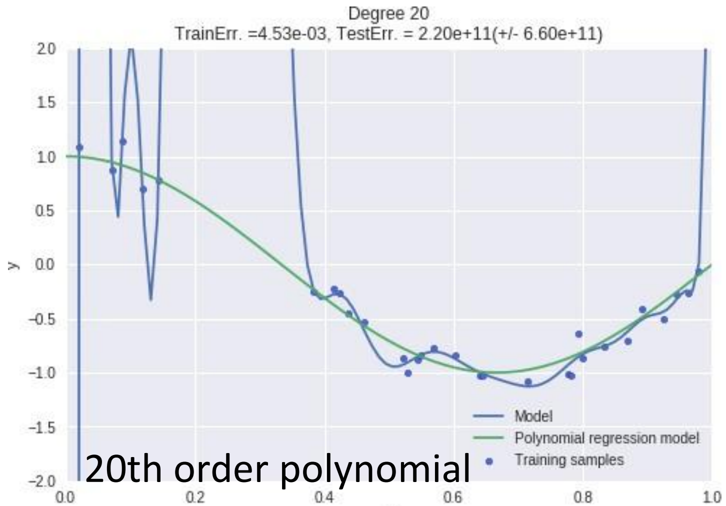
$$\mathrm{TestErr} = \frac{1}{|X_{\mathrm{ts}}|} \sum_{(\boldsymbol{x}_i, t_i) \in X_{\mathrm{ts}}} (t_i - \hat{f}(\boldsymbol{x}_i))^2$$



Evaluation based on Test Error



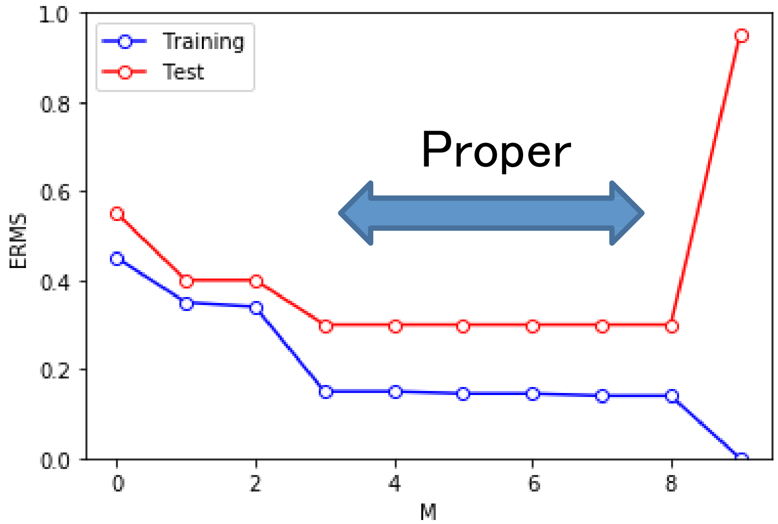


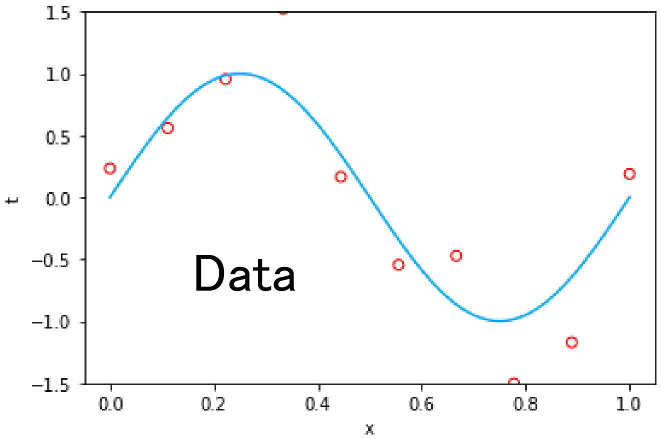


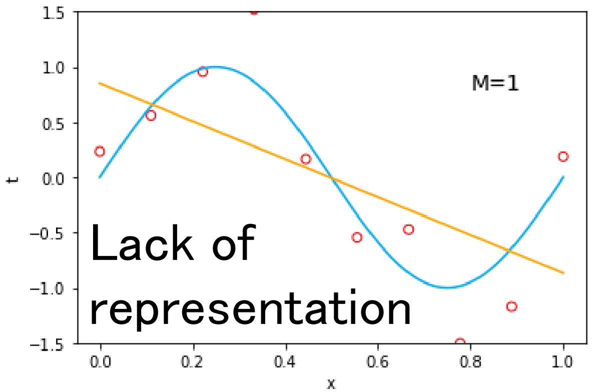
2nd order polynomial		4th order polynomial		20th order polynomial
Training error =2.97e-02	>	Training error = 1.16e-02	>	Training error = 4.53e-03
Test error =5.82e-02	>	Test error = 4.32e-02	<	Test error = 2.20e+11

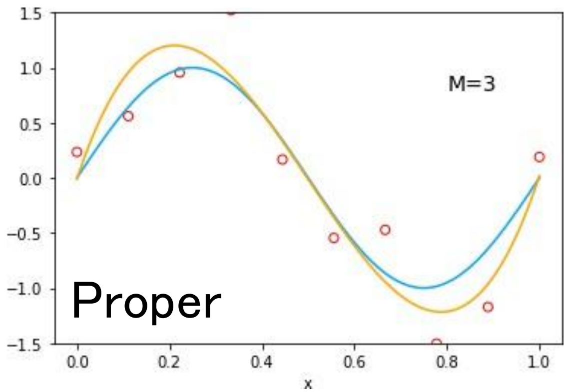
- Training error
  - decreases when the polynomial dimension increases
  - can be zero when the polynomial dimension steadily increased
- Test error
  - is large when the polynomial dimension is low
  - decreases to a certain extent, when increasing the polynomial dimension
  - increases when the polynomial dimension is raised more than necessary (overfitting)

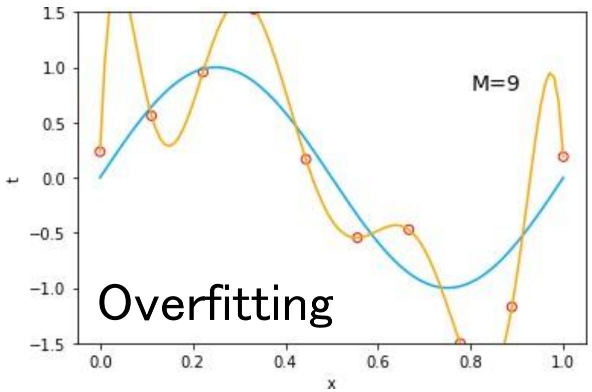
Overfitting











Large training error ← Small training error

Large generalization error ← → Large generalization error

Small generalization error

We need to properly control the complexity of the model

How to Control Model Complexity

- Model selection
  - Prepare models with various complexity
  - Train all of those models and select the one that gives the smallest test error
  - As a result, a model of moderate complexity is selected

- Regularization
  - Have a sufficiently complex model
  - Embed a mechanism to reduce model complexity in the error function
  - Train while varying the complexity of the model and select the complexity that gives the smallest test error
  - As a result, a model of moderate complexity is trained

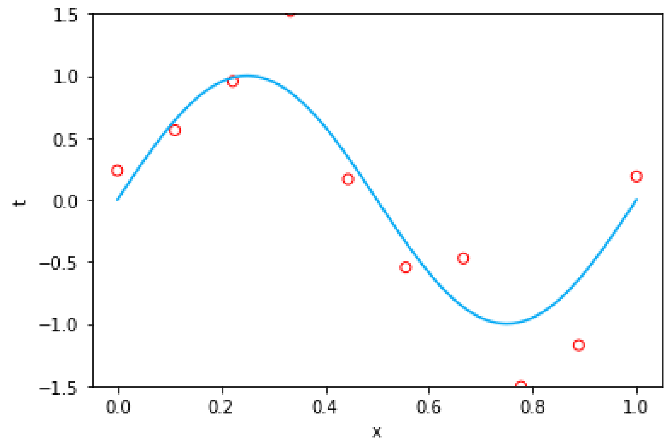
- In any case, it is important to evaluate by Test error

Evaluation of Test Error by K-Fold Cross Validation

Learning (training error minimization) Test error evaluation Test error evaluation Learning (training error minimization)

- Divide the sample into *k* pieces
- For i = 1,⋯, k
  - Evaluate the test error using the i-th division as a test sample and the remaining $K-1$ as a training sample
- Average the test errors of all folds and output this as an estimate of generalization error
- Is it not enough to simply divide into two, learn with one (training sample), and evaluate the test error with one (test sample)?
  - Learning results depend on training samples
  - If the samples with a large "accidental" bias are concentrated on the training samples, the learning model will also be biased
  - Training and test error evaluation in various divisions minimizes bias

Model Selection using Cross-Validation



- Model selection
  - We have data
  - M = 1,2,･･･, 9, Regression with which polynomial?

- Model selection by cross-validation
  1. For M = 1,･･･, 9
     - Modeling with M-th order polynomial regression
     - Evaluate test error with k-fold cross-validation
  2. Adopt M which gives the minimum test error
     - Overall MK learning / test error evaluation required

Regularization

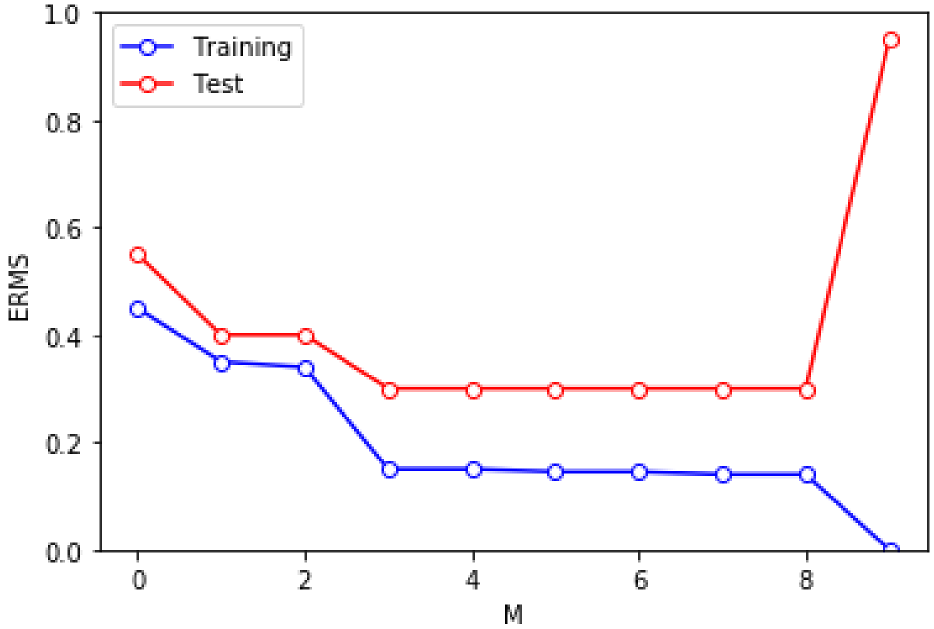
- High complexity model
  - Training error is low
  - Test error is high

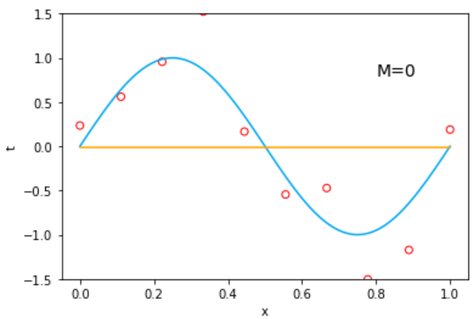
  → Overfitting

- Training error can be minimized directly, but test error cannot be minimized directly
  - If we use a test sample during learning, that sample cannot be used for test error evaluation
- Instead, regularization
  - Use by blunting a blade that is too sharp
  - Control complexity with knowledge of the model
  - Controlling complexity can control test error (expected)

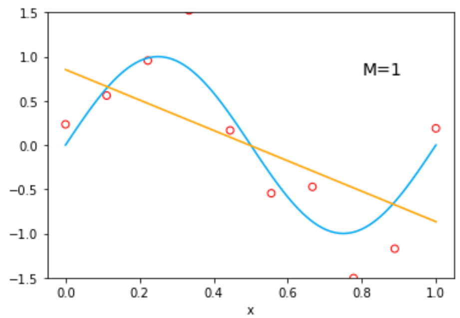
What kind of model is a complicated model?

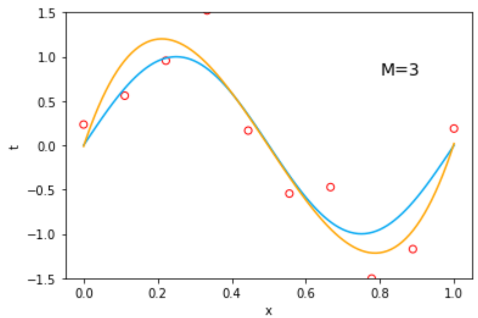
	M=0	M=1	M=3	M=9
w_0^*	0.13	0.64	-0.1	0.07
w_1^*		-1.04	10.9	356.27
w_2^*			-31.05	-812.08
w_3^*			20.53	7247.59
w_4^*				-33584.99
w_5^*				90058.44
w_6^*				-145162.27
w_7^*				138775.85
w_8^*				-72490.8
w_9^*				15932.65



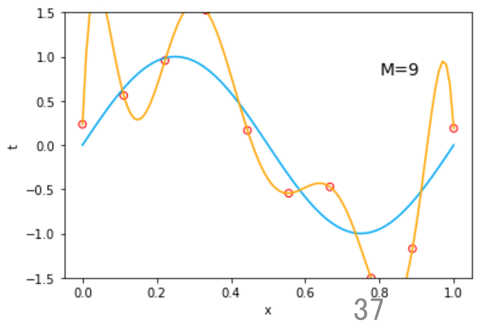


Training error is high
Test error is high





Training error is low
Test error is low



Training error is low
Test error is high

Optimization with Penalty

We want to minimize both $f(\boldsymbol{x}), g(\boldsymbol{x})$

- Both are contradictory, e.g.
- $f(\boldsymbol{x})$ Rent (minimized)
- $g(\boldsymbol{x})$ Distance from the station (minimized)

Trade-off

- Optimization with penalty

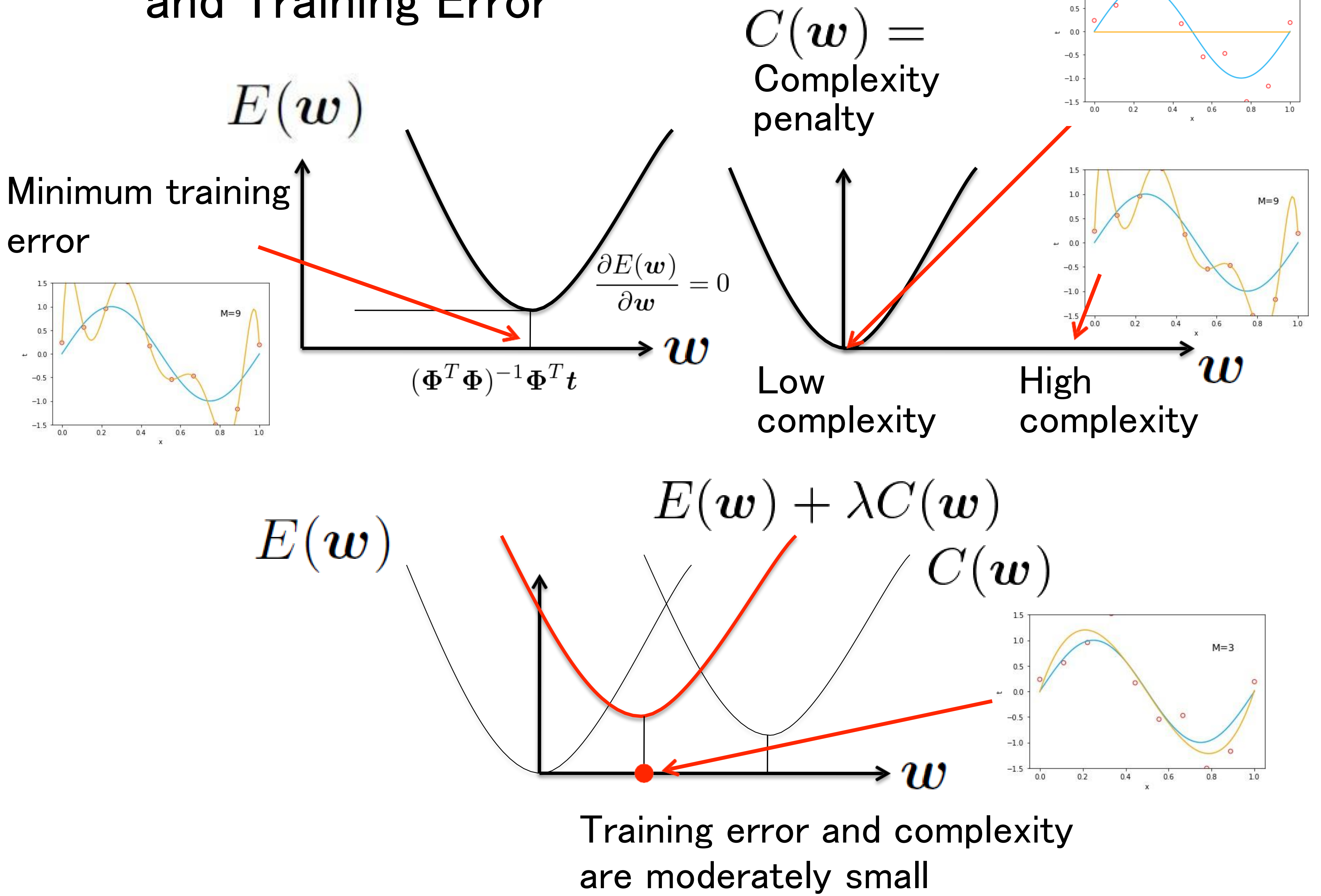
$$\text{minimize } h(\boldsymbol{x}) = f(\boldsymbol{x}) + \underbrace{\lambda}_{\text{Trade-off parameter}} g(\boldsymbol{x})$$

- In case of regression …
  - Minimize training error
  - Minimize complexity

Trade-off

- Expected to reduce generalization error

Minimization of Complexity Penalty and Training Error



Norm

- Vector

$$\boldsymbol{x}^T = (x_1, x_2, \ldots, x_D)$$

- p-norm

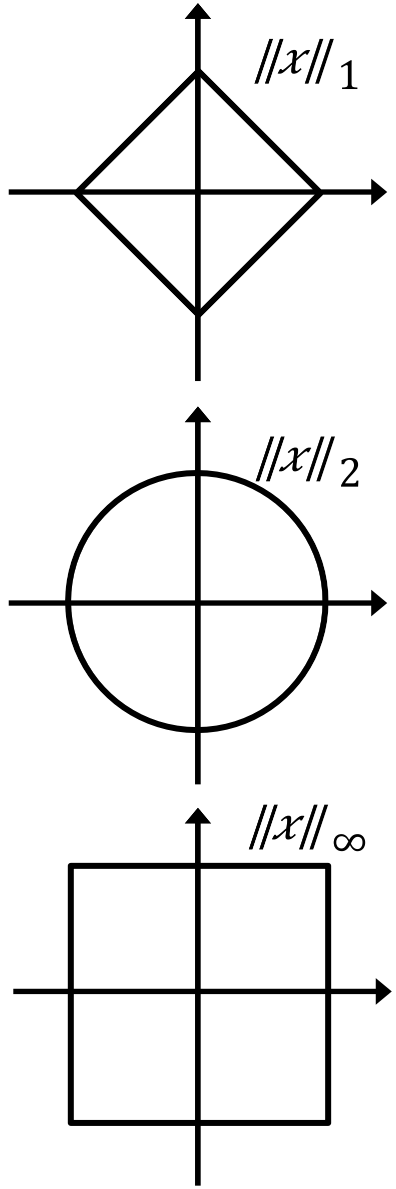
$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{D} |x_i|^p\right)^{1/p}$$

- Euclidean norm (2-norm, distance)

$$\|\boldsymbol{x}\|_2 = \left(\sum_{i=1}^{D} |x_i|^2\right)^{1/2}$$

- Max Norm

$$\|\boldsymbol{x}\|_\infty = \max(|x_1|, |x_2|, \ldots, |x_D|)$$



Unit circle defined by each norm

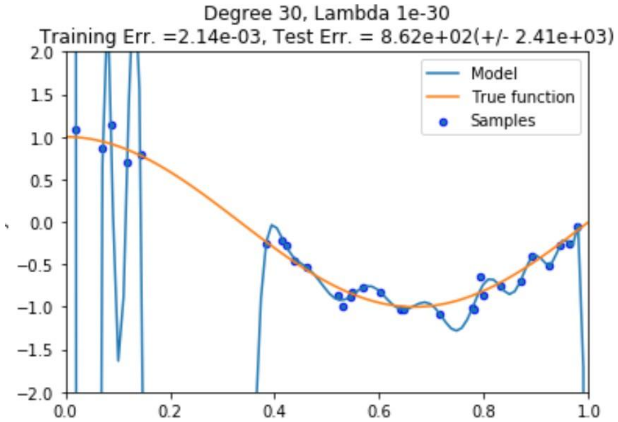
Ridge Regression = Squared Error Term + L2 Regularization Term

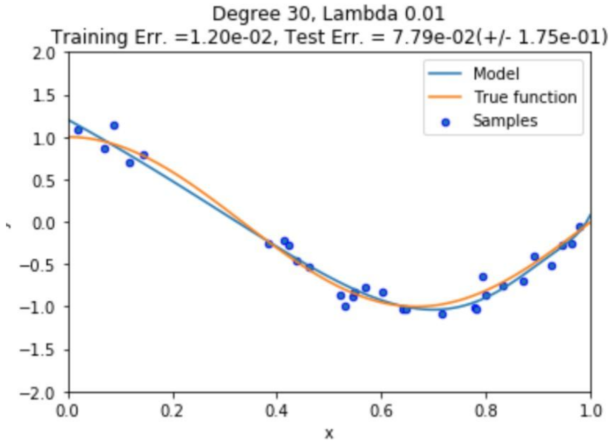
- Error function so far $E(\boldsymbol{w}) = \sum_{i=1}^{N} (t_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2$
- Error function with L2 regularization

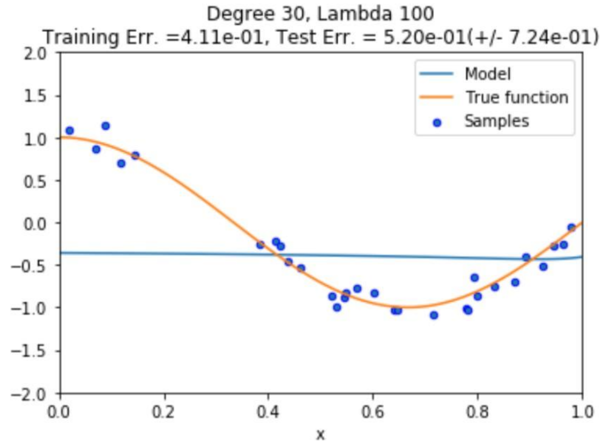
$$E(\boldsymbol{w}) = \sum_{i=1}^{N} \underbrace{(t_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2}_{\text{Squared error}} + \underbrace{\lambda}_{\text{Regularization parameter}} \underbrace{\boldsymbol{w}^T \boldsymbol{w}}_{\text{L2 regularization}}$$

- L2 regularization term
  - Becomes large when each element of *w* takes a large value
  - – Reduce complexity
  - Sum of convex functions is convex function → only one local optimal solution
  - Differentiable → Analytical solution can be found

Effect of L2 Regularization







No regularization
$\lambda=0$

$\lambda=0.01$

Strong regularization
$\lambda=100$

Regularization parameter	Training error	Model complexity	Generalization error
$\lambda = 0$	Small	Easy to get complicated	Large
$\lambda = \alpha$ Intermediate	Moderate	Moderate	Moderate (expected)
$\lambda = \infty$	Not minimized	Too simple	After all big

Optimization in Machine Learning

- Analytical solution
  - It can be used when "the objective function is differentiable" and "w can be found with a gradient of 0"
  - An accurate solution can be found (once the calculation is completed)
  - Inverse matrix calculation of D*D matrix is required (D = number of dimension)
  - If the data is very large (number of dimension D, number of samples N), it may not be possible to calculate (memory constraints)

- Approximate solution (gradient descent GD, stochastic gradient descent SGD)
  - Repeated descent in the direction of the slope
  - Gradually improve the solution, so a better solution can be obtained according to the time spent
  - No inverse matrix calculation required, light calculation per step
  - Less influenced by memory constraints (especially SGD) even when the data is large

Analytical Solution of Ridge Regression

- In principle, it is same as a regression

$$\frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} = 0 \quad \Rightarrow \quad \boldsymbol{w} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{t}$$

cf. Single regression $\boldsymbol{w} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{t}$

- Cross-validation is required for tuning regularization parameter λ
- It also has the effect of stabilizing the inverse matrix calculation

Complexity Control Summary

- Introduction of nonlinearity → Polynomial features (actually, nothing is different from linear regression including calculation method)
- Models that are too complex have low training error but high generalization error (overfitting) and are meaningless
- How to choose a model with right complexity?
  - Prepare models with various complexity, estimate each test error by k-fold cross-validation, and select a model with smallest test error
  - Introduce the regularization term into the error function, estimate the test error while changing the regularization parameter, and select the learning result by the regularization parameter λ with smallest test error

Machine Learning (6)(7)

Model complexity
and
Generalization

Machine Learning

INFQ612L, 440113450A

Spring Semester

Friday 17:00–18:40

IPS

WASEDA University

Prof. Shoji Makino