11. Regularization, Validation

STA3142 Statistical Machine Learning

Kibok Lee

Assistant Professor of

Applied Statistics / Statistics and Data Science

Apr {2, 4}, 2024

연세대학교
YONSEI UNIVERSITY

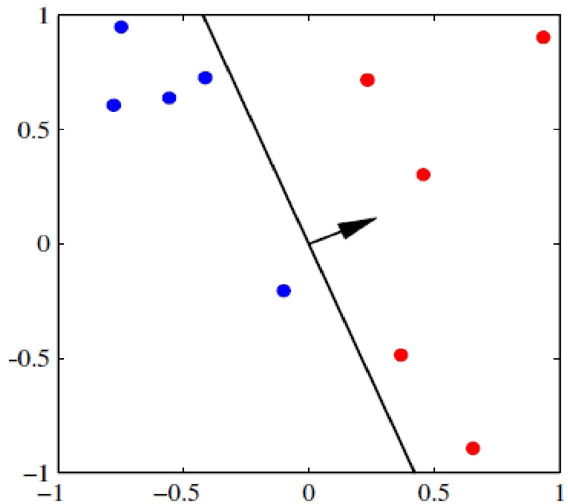
Assignment 2

- Due **Friday 4/12, 11:59pm**
- Topics
  - (Math/Programming) Logistic Regression
  - (Math/Programming) Softmax Regression
  - (Math) Gaussian Discriminant Analysis
  - (Programming) Naïve Bayes for Spam Classification

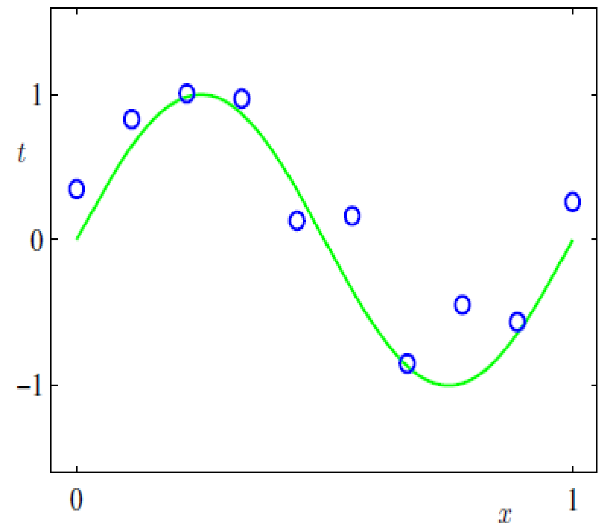
- Please read the instruction carefully!
  - Submit one pdf and one zip file separately
  - Write your code only in the designated spaces
  - Do not import additional libraries
  - …
- If you feel difficult, consider to take **option 2**.

Recap: Supervised Learning

- Learning a function $h: \mathcal{X} \rightarrow \mathcal{Y}$
- Labels could be discrete or continuous
  - Discrete labels: **classification**
  - Continuous labels: **regression**



classification



regression

Outline

- Regularization
  - Probabilistic Interpretation: MAP Estimation
  - Example: Curve Fitting
- Bias-Variance Tradeoff
- Validation for Model Selection

MLE vs. MAP

- **Maximum Likelihood Estimation (MLE)**
  - Find w that maximizes the probability of observed data.
  - **Learning objective**: Log-likelihood
$$\log p(D|w)$$
    - e.g., Linear regression
    - e.g., Logistic regression
- **Maximum A Posteriori (MAP) Estimation**
  - Find the most probable w given the observed data.
  - Bayes' rule: $p(w|D) \propto p(D|w)p(w)$
  - **Learning objective**: Log-likelihood + log-prior
$$\log p(D|w)p(w)$$
    - e.g., Regularized linear regression
    - e.g., Regularized logistic regression

Maximum Likelihood Estimation

- Find $\mathbf{w}$ that maximizes the probability of observed data.
- **Learning objective**: Log-likelihood
  - Under the **i.i.d.** (independent and identically distributed) assumption:

$$\log \mathrm{P}(\mathbf{D} \mid \mathbf{w}) = \log \prod_{n=1}^{N} P(y^{(n)}|\phi(\mathbf{x}^{(n)}), \mathbf{w})$$

$$= \sum_{n=1}^{N} \log P(y^{(n)}|\phi(\mathbf{x}^{(n)}), \mathbf{w})$$

- Issue: Risk of overfitting

Maximum A Posteriori Estimation

- Find the most probable $\mathbf{w}$ given the observed data.
- **Learning objective**:
  - Under the **i.i.d.** assumption:

$$\log \mathrm{P}(\mathbf{D} \mid \mathbf{w})\mathrm{P}(\mathbf{w}) = \log \prod_{n=1}^{N} P(y^{(n)}|\phi(\mathbf{x}^{(n)}), \mathbf{w}) + \log \mathrm{P}(\mathbf{w})$$

$$= \sum_{n=1}^{N} \log P(y^{(n)}|\phi(\mathbf{x}^{(n)}), \mathbf{w}) + \log \mathrm{P}(\mathbf{w})$$

- Assuming a prior distribution: $P(\mathbf{w})$
- **Point estimate** using Bayes' rule:

$$\underset{\mathbf{w}}{\operatorname{argmax}} P(\mathbf{w}|D) = \underset{\mathbf{w}}{\operatorname{argmax}} P(D|\mathbf{w})P(\mathbf{w})$$

Maximum A Posteriori Estimation

- Find the most probable $\mathbf{w}$ given the observed data.
- **Learning objective**:
  - Under the **i.i.d.** assumption:

$$\log \mathrm{P}(\mathbf{D} \mid \mathbf{w})\mathrm{P}(\mathbf{w}) = \log \prod_{n=1}^{N} P(y^{(n)} | \phi(\mathbf{x}^{(n)}), \mathbf{w}) + \log \mathrm{P}(\mathbf{w})$$

$$= \sum_{n=1}^{N} \log P(y^{(n)} | \phi(\mathbf{x}^{(n)}), \mathbf{w}) + \log \mathrm{P}(\mathbf{w})$$

- Popular prior distributions:
  - Isotropic Gaussian prior: L2 regularizer
    - $P(\mathbf{w}) = N(0, \lambda^{-1}\mathbf{I}) \Leftrightarrow \log P(\mathbf{w}) = -\frac{\lambda}{2}\|\mathbf{w}\|^2 + const$
  - Isotropic Laplace prior: L1 regularizer
    - $P(\mathbf{w}) \propto \exp(-\lambda\|\mathbf{w}\|_1) \Leftrightarrow \log P(\mathbf{w}) = -\lambda\|\mathbf{w}\|_1 + const$

MAP with Isotropic Gaussian Prior

- Isotropic Gaussian distribution for $\mathbf{w}$

$$
\begin{aligned}
P(\mathbf{w}) &= \mathcal{N}(0, \lambda^{-1}\mathbf{I}) \\
&= const * \exp\left(-\frac{1}{2}\mathbf{w}^T(\lambda^{-1}I)^{-1}\mathbf{w}\right) \\
&= const * \exp\left(-\frac{\lambda}{2}\mathbf{w}^T\mathbf{w}\right)
\end{aligned}
$$

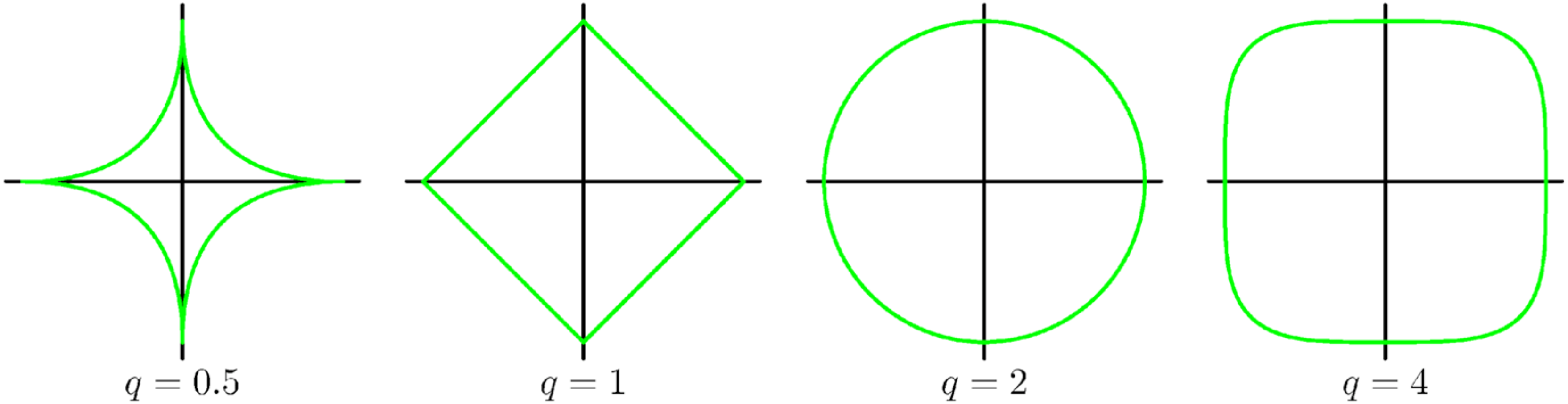
- Taking a log:

$$
\log P(\mathbf{w}) = const - \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} = const - \frac{\lambda}{2}\|\mathbf{w}\|^2
$$

- Isotropic Gaussian prior for $\mathbf{w}$ is equivalent to L2 regularizer.

Maximum A Posteriori Estimation

- Popular prior distributions:
  - Isotropic Gaussian prior: L2 regularizer
    - $P(\mathbf{w}) = N(0, \lambda^{-1}\mathbf{I}) \Leftrightarrow \log P(\mathbf{w}) = -\frac{\lambda}{2}\|\mathbf{w}\|^2 + const$
  - Isotropic Laplace prior: L1 regularizer
    - $P(\mathbf{w}) \propto \exp(-\lambda\|\mathbf{w}\|_1) \Leftrightarrow \log P(\mathbf{w}) = -\lambda\|\mathbf{w}\|_1 + const$

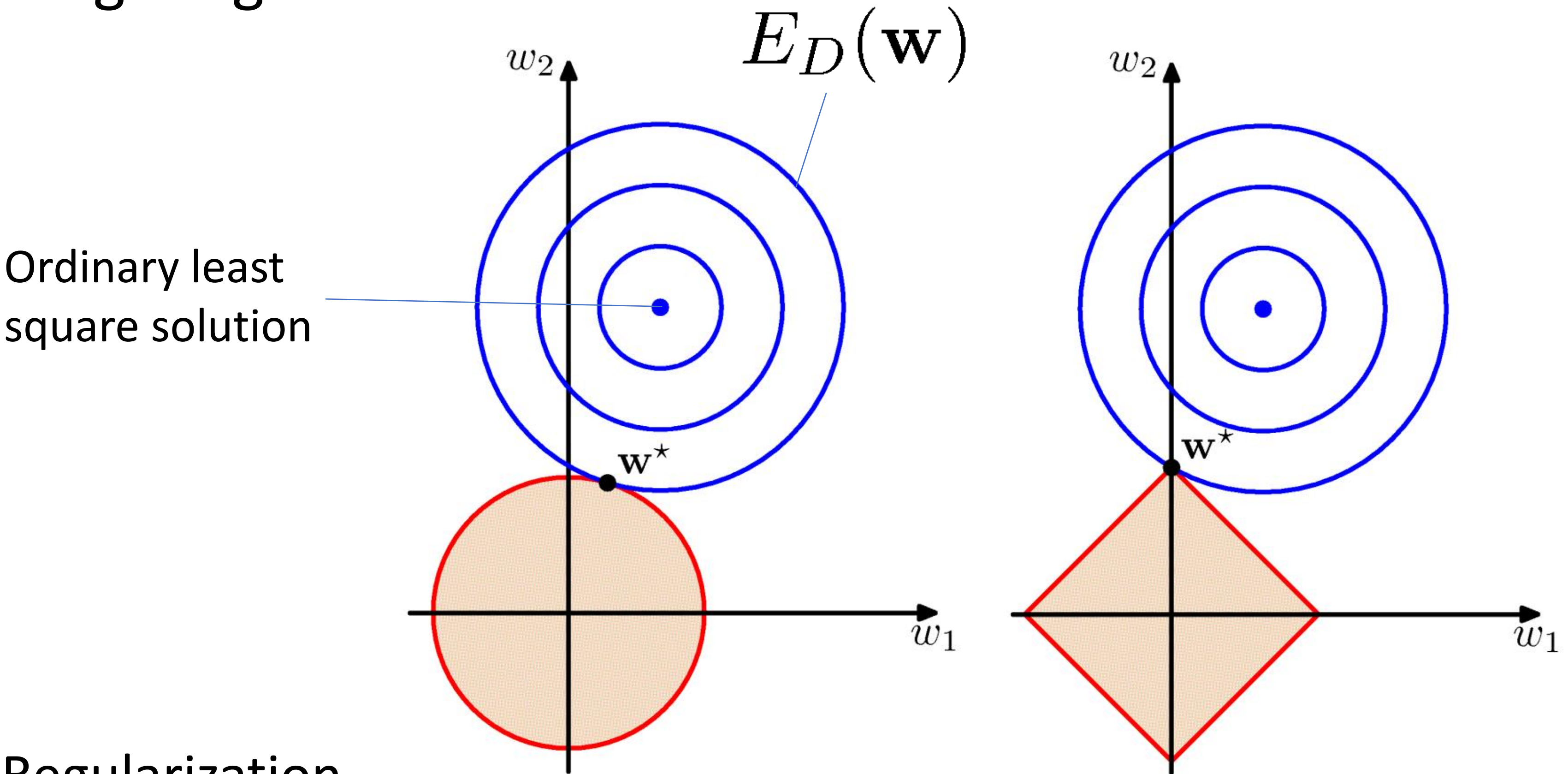


Lasso/L1 regularization

Quadratic/Ridge/L2 regularization

Recap: L1 vs. L2 Regularization

- Lasso tends to generate sparser solutions than ridge regularization.



Regularization as constraints: $\sum_j w_j^2 \leq C$ $\quad\quad$ $\sum_j |w_j| \leq C$

Recap: Regularized Least Squares

- Consider the error function:

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

Data term + Regularization term

λ is called the regularization coefficient.

- With the sum-of-squares error function and a quadratic (a.k.a. ridge or L2) regularizer, we get

Penalize large $\mathbf{w}$ values

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

- Closed-form solution:

$$\mathbf{w}_{ML} = (\lambda\mathbf{I} + \Phi^T\Phi)^{-1}\Phi^T\mathbf{y}$$

Derivation

Objective function

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

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

Compute gradient and set it zero:

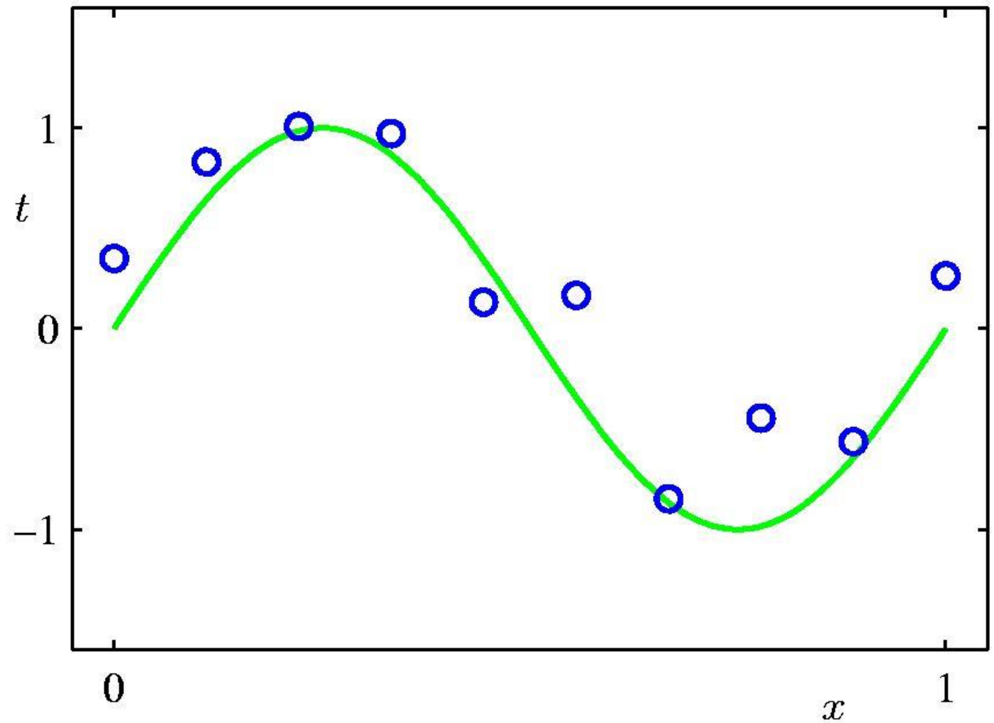
$$\begin{aligned}\nabla_{\mathbf{w}}E(\mathbf{w}) &= \nabla_{\mathbf{w}}[\frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}^T\Phi^T\mathbf{y} + \frac{1}{2}\mathbf{y}^T\mathbf{y} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}] \\ &= \Phi^T\Phi\mathbf{w} - \Phi^T\mathbf{y} + \lambda\mathbf{w} \\ &= (\lambda\mathbf{I} + \Phi^T\Phi)\mathbf{w} - \Phi^T\mathbf{y} \\ &= 0\end{aligned}$$

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

C.f. Ordinary Least Squares

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

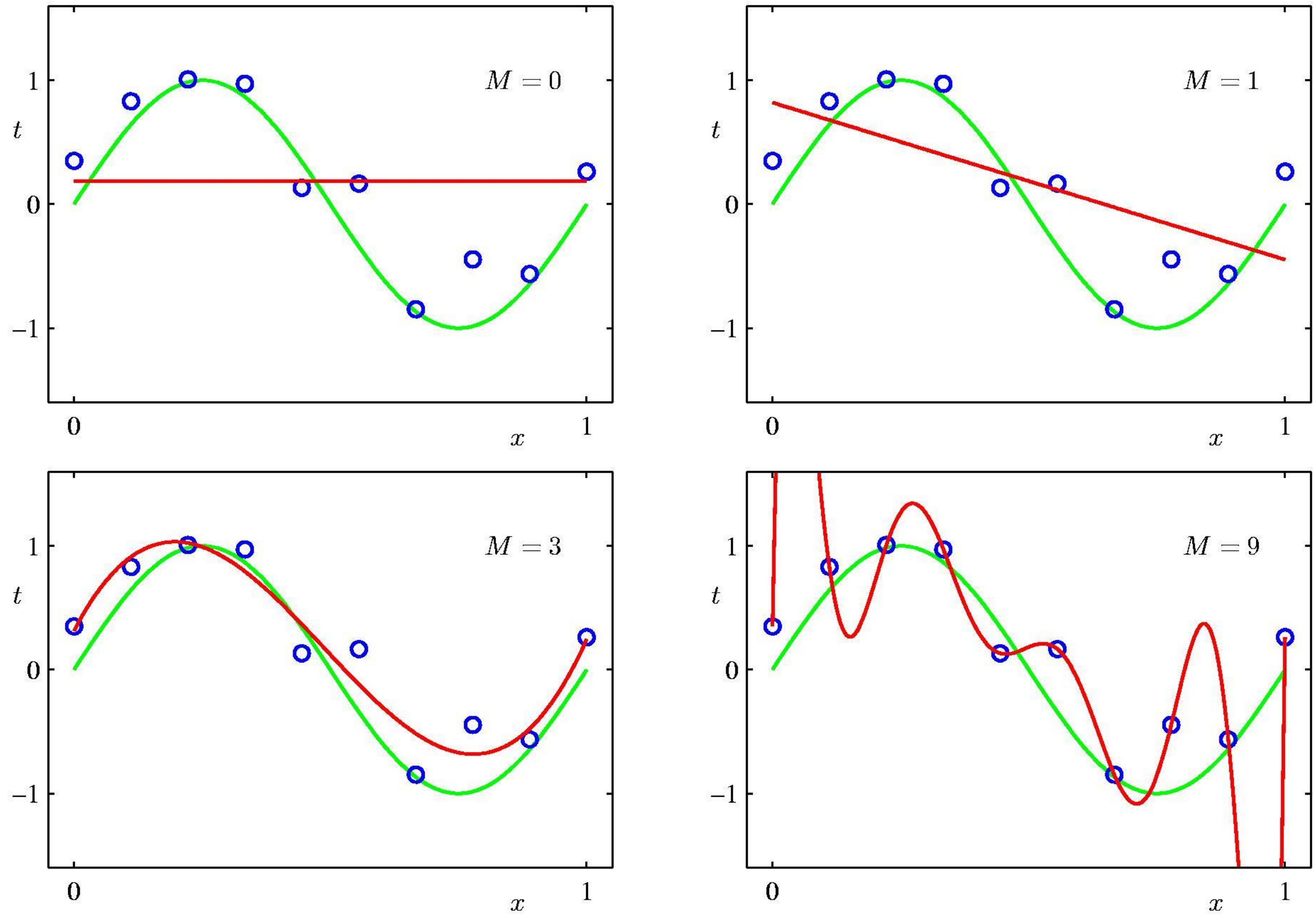
Linear Regression on a Polynomial



$$h(x, \mathbf{w}) = w_0 + w_1 x + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

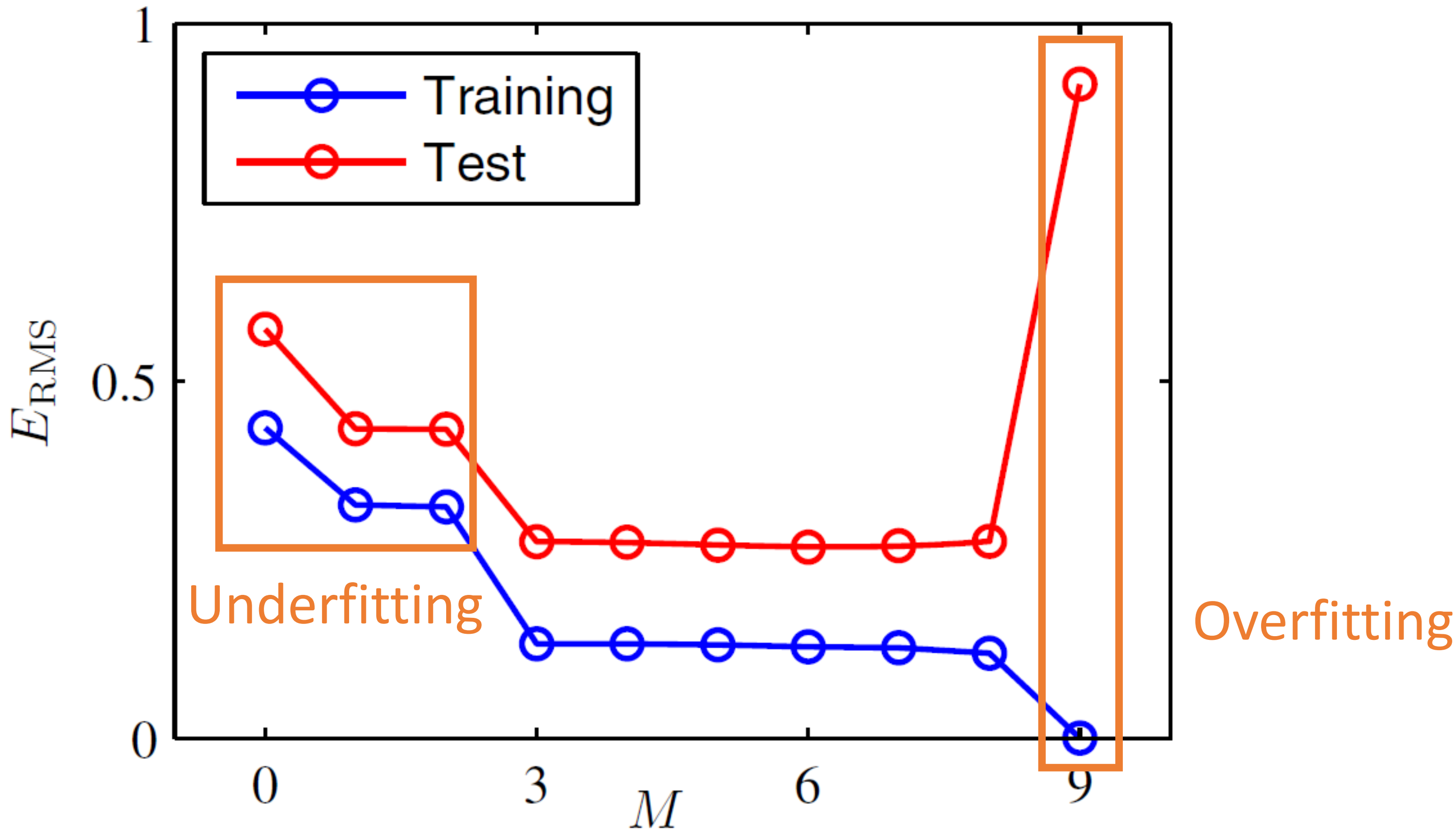
Linear Regression on a Polynomial

- Choosing the right complexity is important
  - To avoid underfitting/overfitting

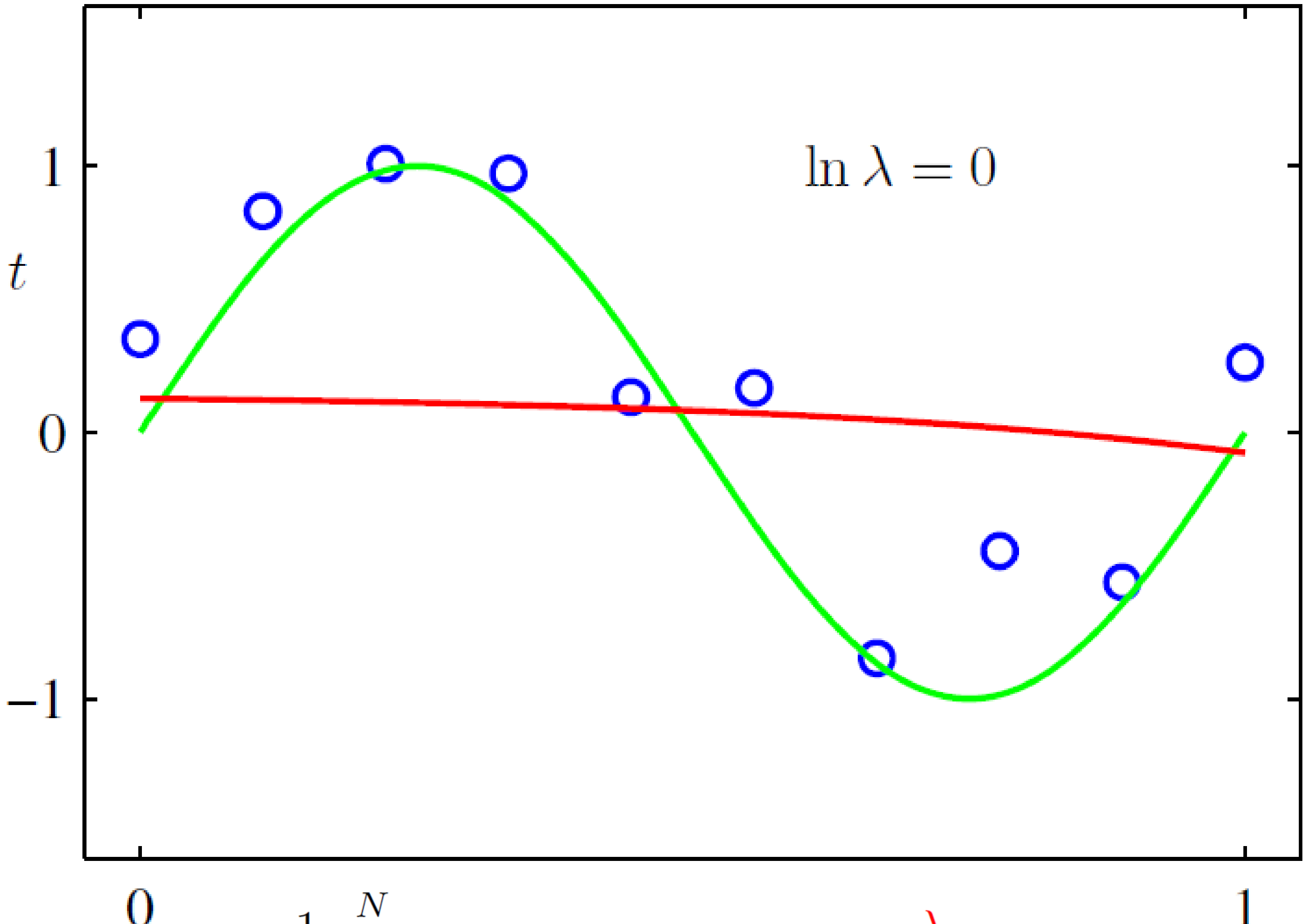


Underfitting vs. Overfitting

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

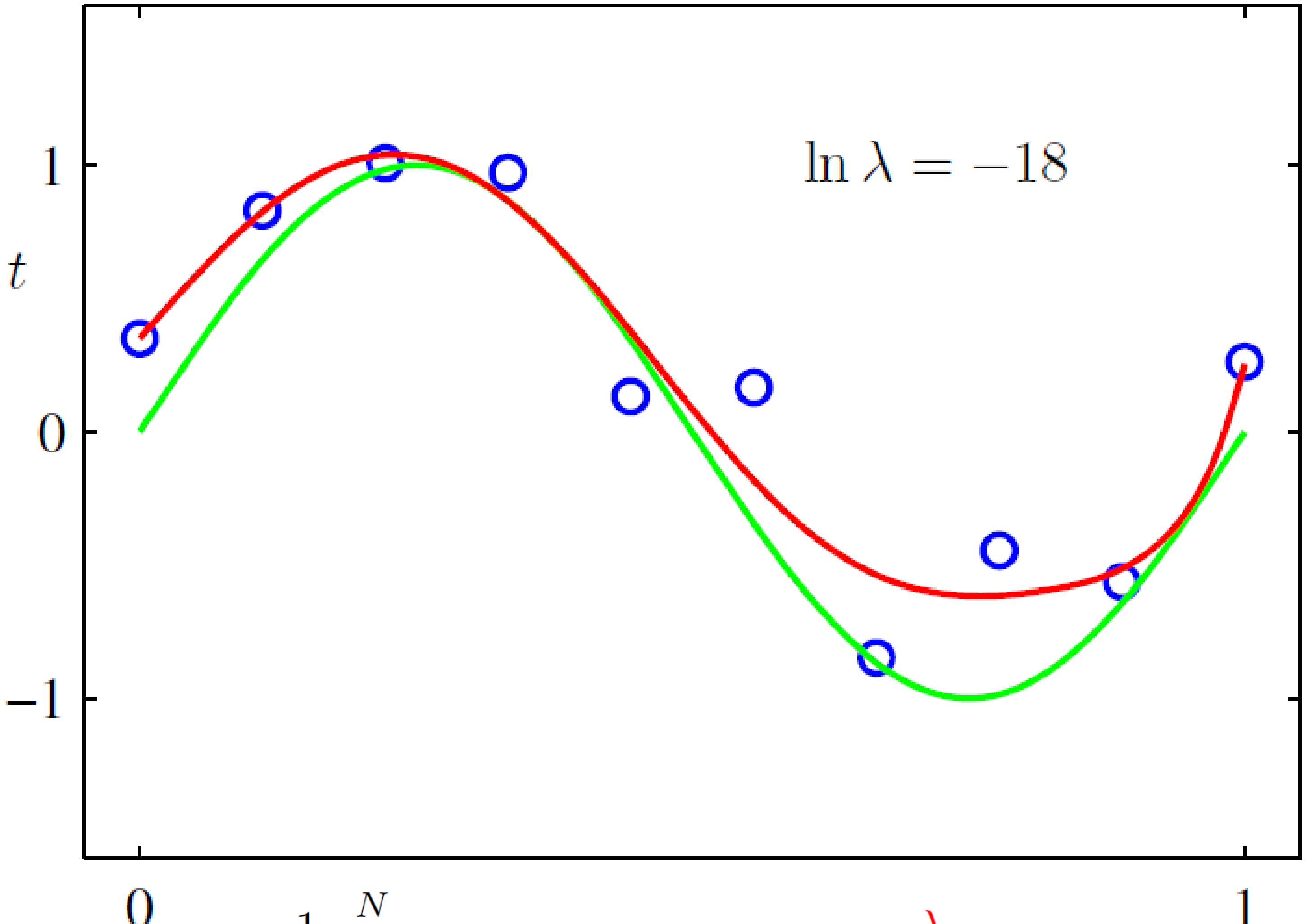


L2 Regularization when $\log \lambda = 0$



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

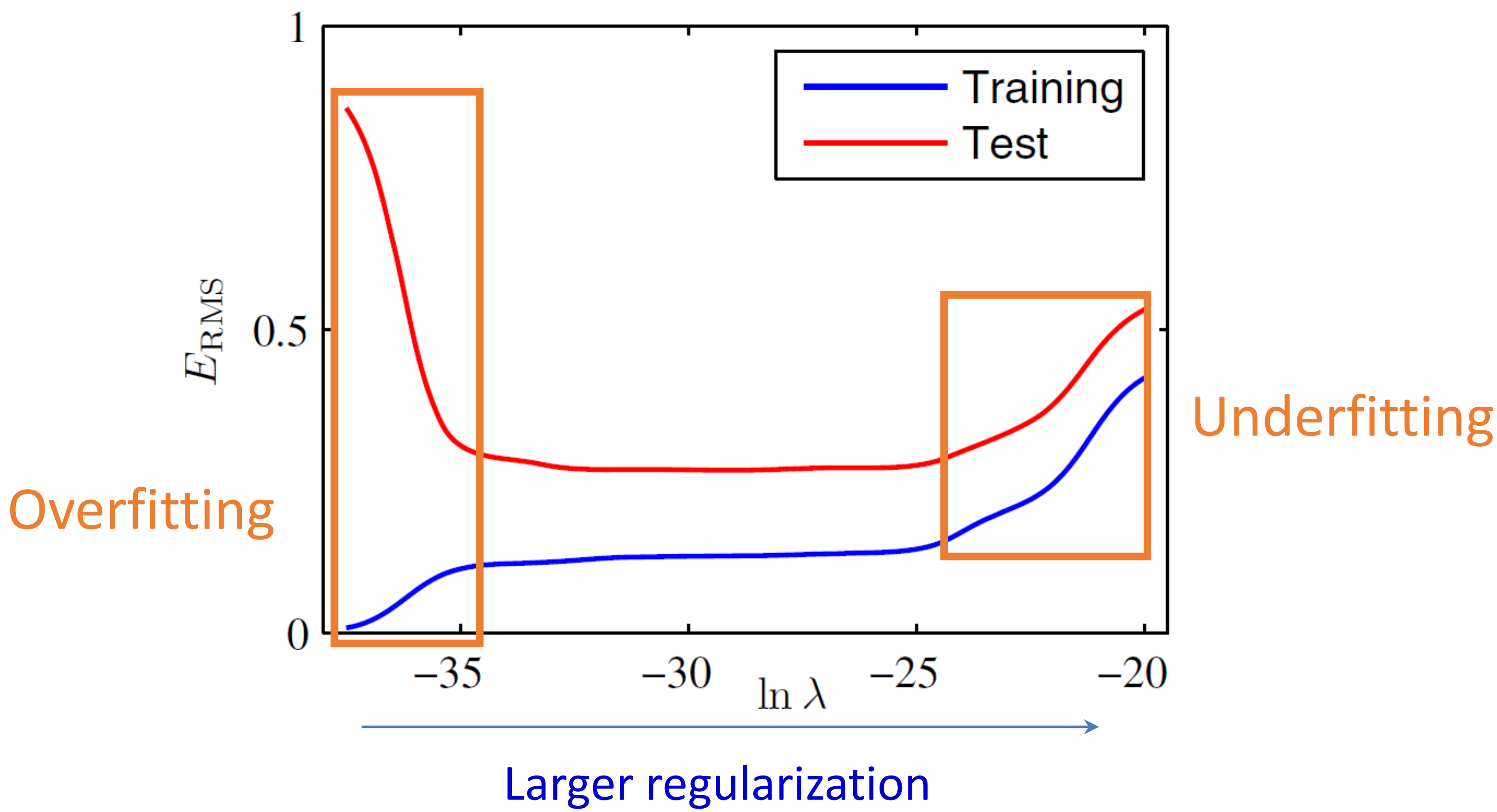
L2 Regularization when $\log\lambda = -18$



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

L2 Regularization: E_{RMS} vs. λ

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



NOTE: For simplicity of presentation, we divided the data into training set and test set. However, it's **<u>not</u>** 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 validation**.

Polynomial Coefficients

- Choose an appropriate λ to avoid under/overfitting

	Overfitting	Sweet spot	Underfitting
	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	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

Summary: Regularization

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

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

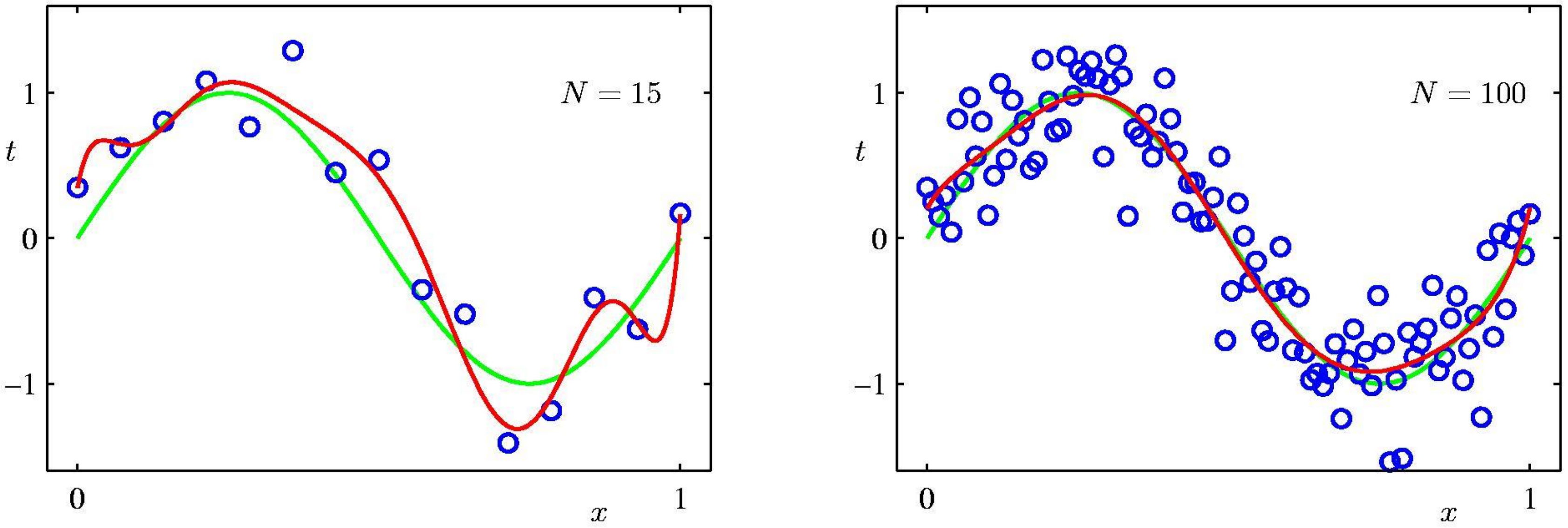
How to Avoid Overfitting?

- More training data
  - Collecting a large training dataset is expensive.
  - Optimization takes a long time.

- **Regularization**
  - Penalize complex models
  - e.g., MAP for probabilistic classification models

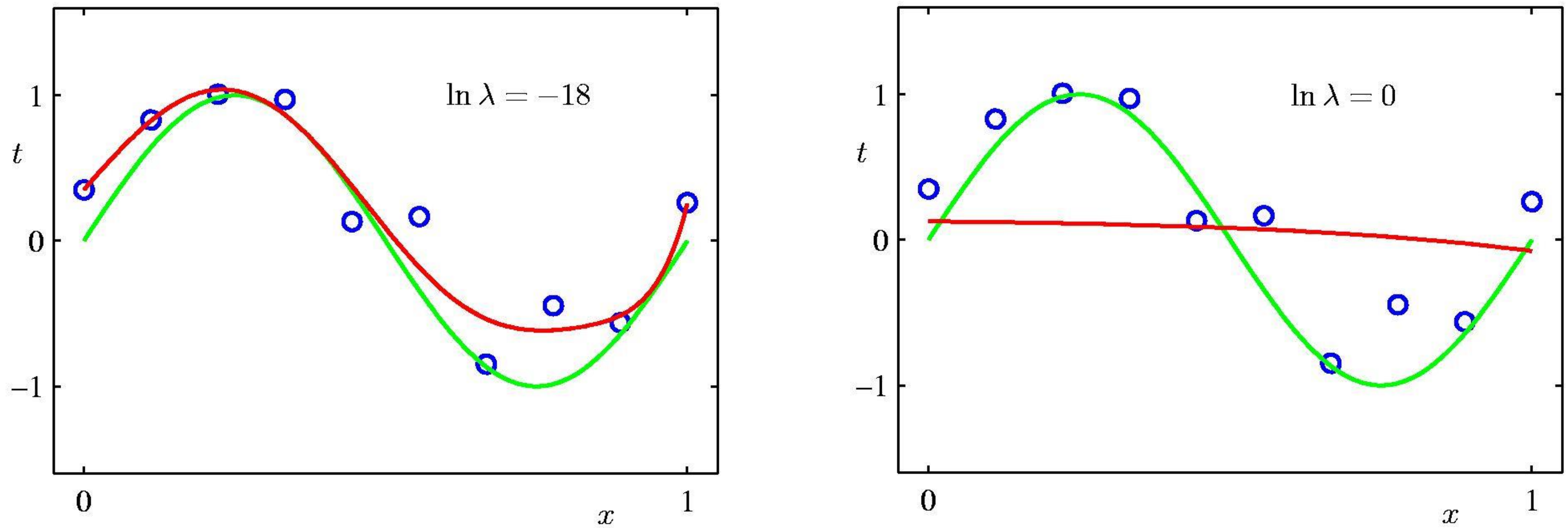
More Training Data

- Even complicated models can benefit by having large amount of data to avoid overfitting.
  - Example: 9th order polynomial



Regularization

- Regularization can implicitly control the complexity of models
  - Example: 9^{th} order polynomial
  - Choosing right level of regularization is important



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

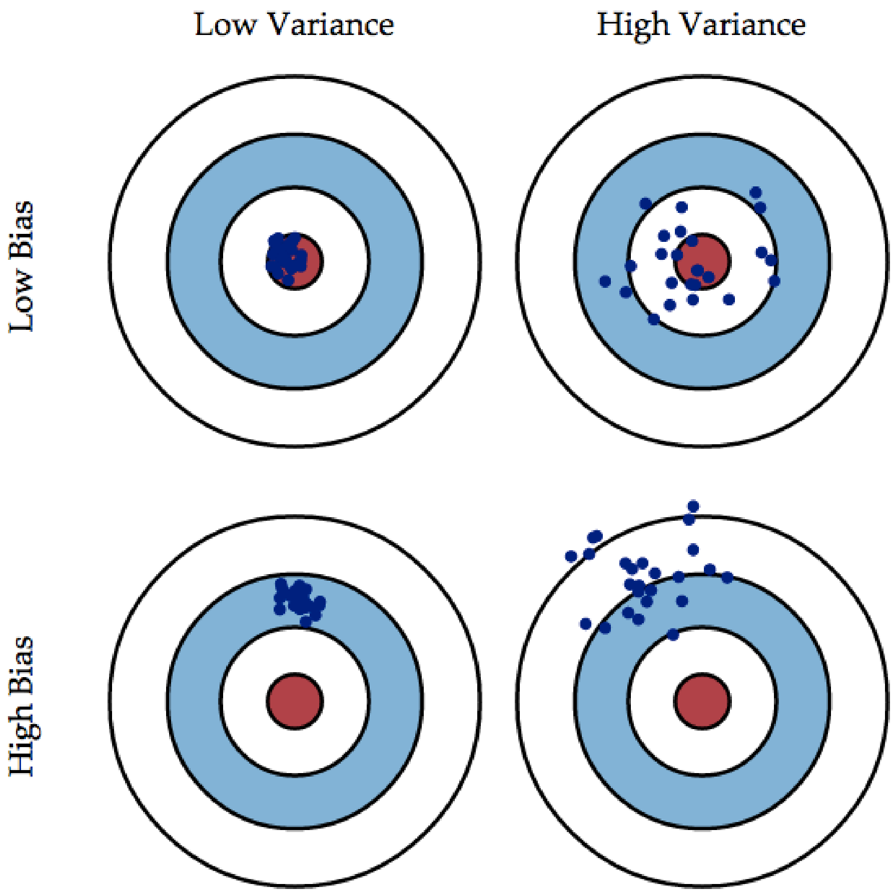
Data term + Regularization term

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

Bias-Variance Tradeoff

Bias and Variance

- Bias: How well a model fits the data on average
- Variance: How stable a model is w.r.t. data samples



The Bias-Variance Decomposition

- Assume a training dataset is sampled from a data distribution $P(\mathbf{x}, y)$:
$$D = \left\{\left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(n)}, y^{(n)}\right)\right\} \sim P(\mathbf{x}, y)$$
- Train an ML algorithm on the sampled dataset D.
- Depending on the sampling result, the algorithm can give different learning results.
- Ideally, we want the learned model with
  - Small bias: The model fits the data well on average.
  - Small variance: The model is stable w.r.t. data sampling.

The Bias-Variance Decomposition

- Expected squared loss

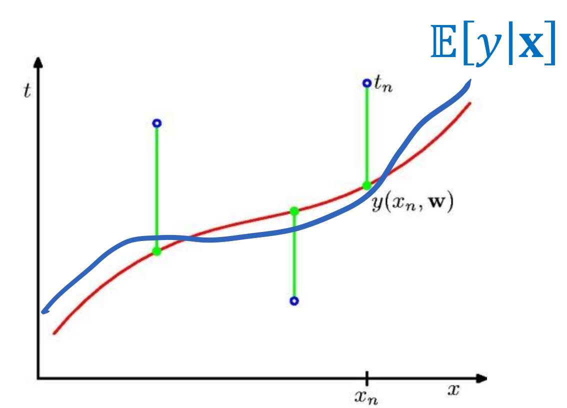
$$\mathbb{E}[L] = \int\int \{h(\mathbf{x}) - y\}^2 p(\mathbf{x}, y)\, d\mathbf{x}\, dy$$

$$\mathbb{E}[L] = \int \{h(\mathbf{x}) - \mathbb{E}[y|\mathbf{x}]\}^2 p(\mathbf{x})\, d\mathbf{x} + \int\int \{\mathbb{E}[y|\mathbf{x}] - y\}^2 p(\mathbf{x}, y)\, d\mathbf{x}\, dy$$

  - where $\mathbb{E}[y|\mathbf{x}] = \int y p(y|\mathbf{x})\, dy$

- The second term corresponds to the noise inherent in the random variable y.

- What does the first term stand for?



The Bias-Variance Decomposition

- Suppose we sampled multiple datasets from a distribution, each of size N.
- Each dataset D will give a learned model $h(\mathbf{x}; D)$.
- The bias-variance decomposition:

$$\mathbb{E}_D[\{h(\mathbf{x}; D) - \mathbb{E}[y|\mathbf{x}]\}^2]$$
$$= \underbrace{\left(\mathbb{E}_D[h(\mathbf{x}; D)] - \mathbb{E}[y|\mathbf{x}]\right)^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_D\left[\{h(\mathbf{x}; D) - \mathbb{E}_D[h(\mathbf{x}; D)]\}^2\right]}_{\text{variance}}$$

The Bias-Variance Decomposition

- Expected squared loss

$$\mathbb{E}[L] = \int\int \{h(\mathbf{x}) - y\}^2 p(\mathbf{x}, y) d\mathbf{x} dy$$

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

- where

$$\mathbb{E}[y|\mathbf{x}] = \int y p(y|\mathbf{x}) dy$$

$$\begin{aligned}
(\text{bias})^2 &= \int \{\mathbb{E}_D[h(\mathbf{x}; D)] - \mathbb{E}[y|\mathbf{x}]\}^2 p(\mathbf{x}) d\mathbf{x} \\
\text{variance} &= \int \mathbb{E}_D\left[\{h(\mathbf{x}; D) - \mathbb{E}_D[h(\mathbf{x}; D)]\}^2\right] p(\mathbf{x}) d\mathbf{x} \\
\text{noise} &= \int\int \{\mathbb{E}[y|\mathbf{x}] - y\}^2 p(\mathbf{x}, y) d\mathbf{x} dy
\end{aligned}$$

The Bias-Variance Decomposition: Derivation

D: training dataset; $\mathbf{x}$: test example; y: label of $\mathbf{x}$

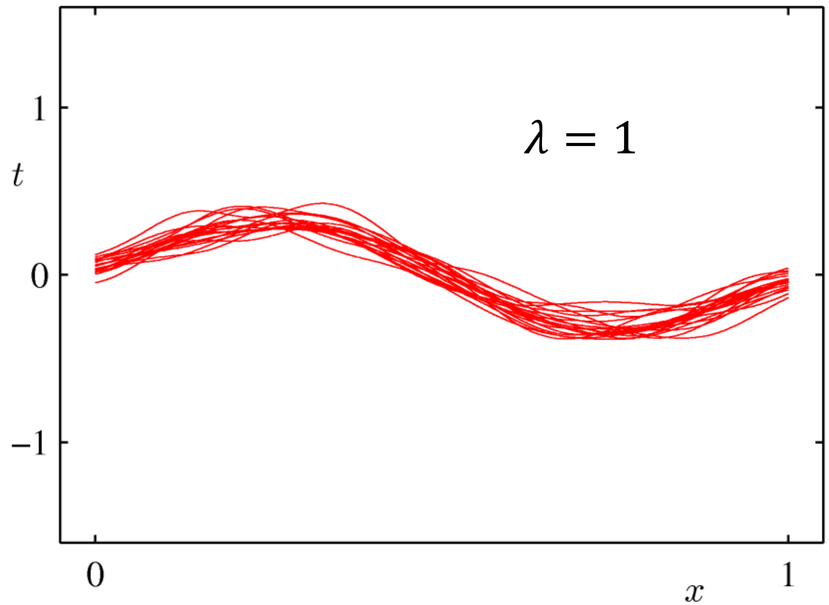
$$
\begin{aligned}
\mathbb{E}[L] &= \mathbb{E}_{\mathbf{x},y,D}\left[(h(\mathbf{x};D) - y)^2\right] \\
&= \mathbb{E}_{\mathbf{x},y,D}\left[(h(\mathbf{x};D) - \mathbb{E}[y|\mathbf{x}])^2\right] + \underline{\mathbb{E}_{\mathbf{x},y,D}\left[(y - \mathbb{E}[y|\mathbf{x}])^2\right]} \\
&= \mathbb{E}_{\mathbf{x},D}\left[(h(\mathbf{x};D) - \mathbb{E}[y|\mathbf{x}])^2\right] + const
\end{aligned}
$$

Noise is constant

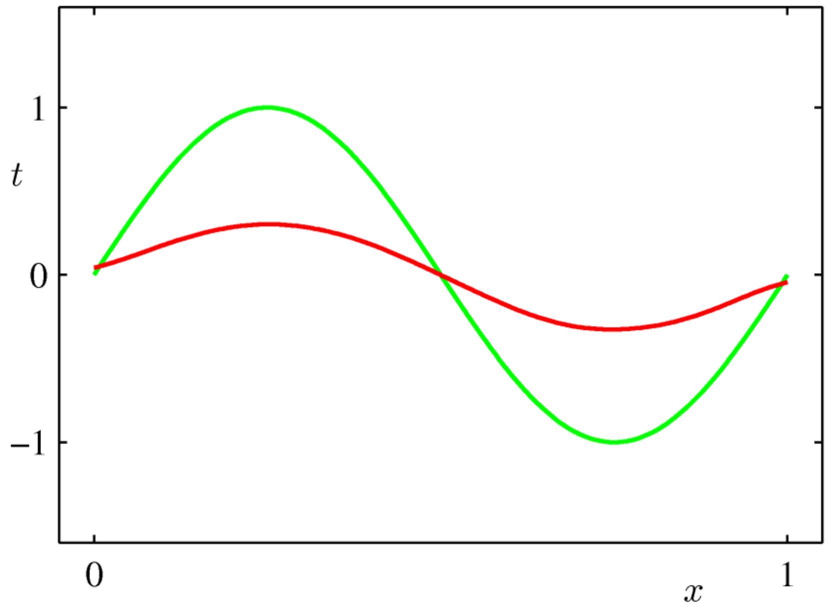
$$
\begin{aligned}
&= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_D\left[(h(\mathbf{x};D) - \mathbb{E}[y|\mathbf{x}])^2\right]\right] + const \\
&= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_D\left[(h(\mathbf{x};D) - \mathbb{E}_D[h(\mathbf{x};D)] + \mathbb{E}_D[h(\mathbf{x};D)] - \mathbb{E}[y|\mathbf{x}])^2\right]\right] + const \\
&= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_D\left[\{h(\mathbf{x};D) - \mathbb{E}_D[h(\mathbf{x};D)]\}^2\right]\right] \\
&\quad + 2\mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_D\left[\{h(\mathbf{x};D) - \mathbb{E}_D[h(\mathbf{x};D)]\}\{\mathbb{E}_D[h(\mathbf{x};D)] - \mathbb{E}[y|\mathbf{x}]\}\right]\right] \\
&\quad + \mathbb{E}_{\mathbf{x}}\left[\{\mathbb{E}_D[h(\mathbf{x};D)] - \mathbb{E}[y|\mathbf{x}]\}^2\right] + const \\
&= \underbrace{\mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_D\left[\{h(\mathbf{x};D) - \mathbb{E}_D[h(\mathbf{x};D)]\}^2\right]\right]}_{\text{Variance}} + \underbrace{\mathbb{E}_{\mathbf{x}}\left[\{\mathbb{E}_D[h(\mathbf{x};D)] - \mathbb{E}[y|\mathbf{x}]\}^2\right]}_{\text{Bias}} + \underbrace{const}_{\text{Noise}}
\end{aligned}
$$

Example: Regularized Linear Regression

- Example: 25 datasets sampled from the sinusoidal, varying the degree of regularization strength λ.



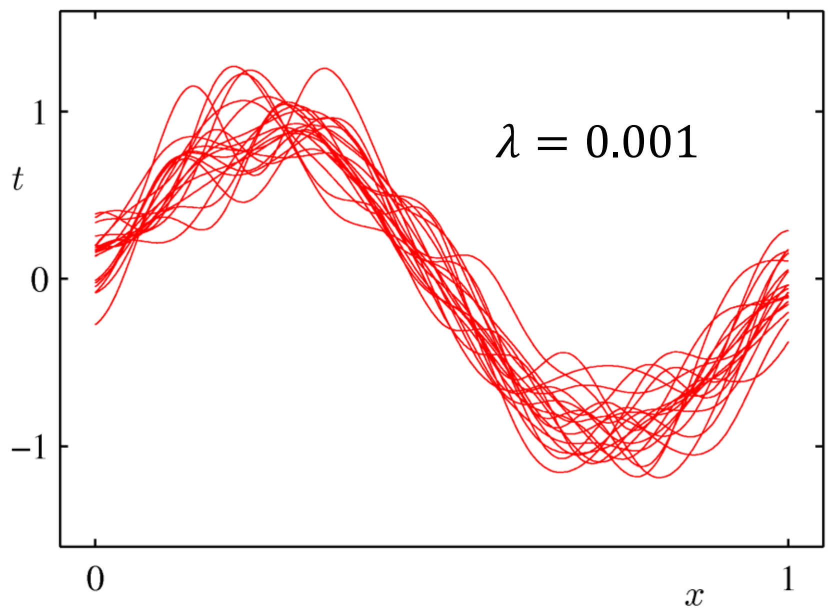
Learned $h(\mathbf{x}; D)$



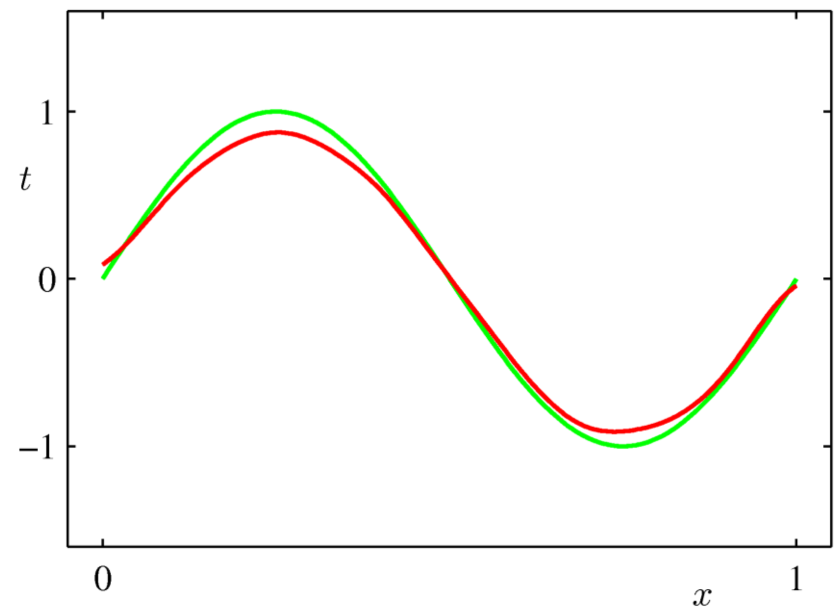
$\mathbb{E}_D[h(\mathbf{x}; D)]$ vs. $\mathbb{E}[y|\mathbf{x}]$

Example: Regularized Linear Regression

- Example: 25 datasets sampled from the sinusoidal, varying the degree of regularization strength λ.



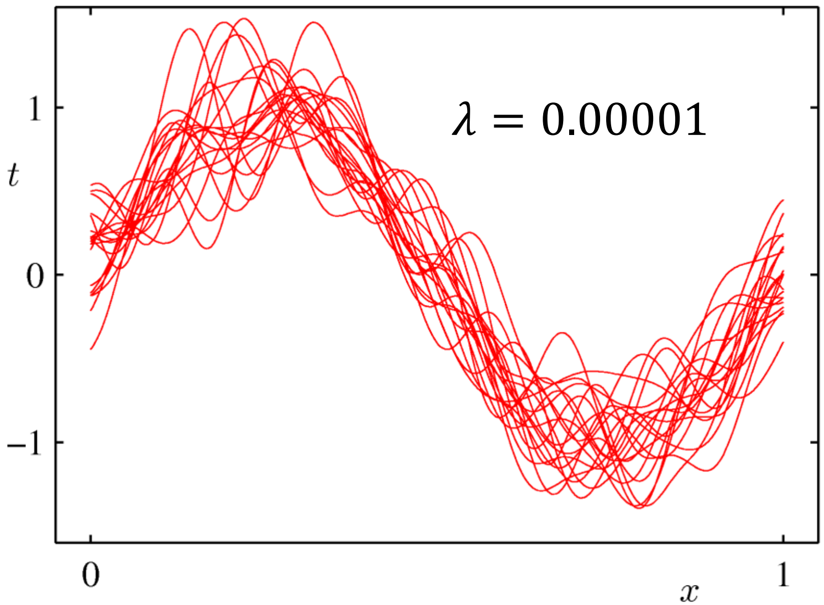
Learned $h(\mathbf{x}; D)$



$\mathbb{E}_D[h(\mathbf{x}; D)]$ vs. $\mathbb{E}[y|\mathbf{x}]$

Example: Regularized Linear Regression

- Example: 25 datasets sampled from the sinusoidal, varying the degree of regularization strength λ.

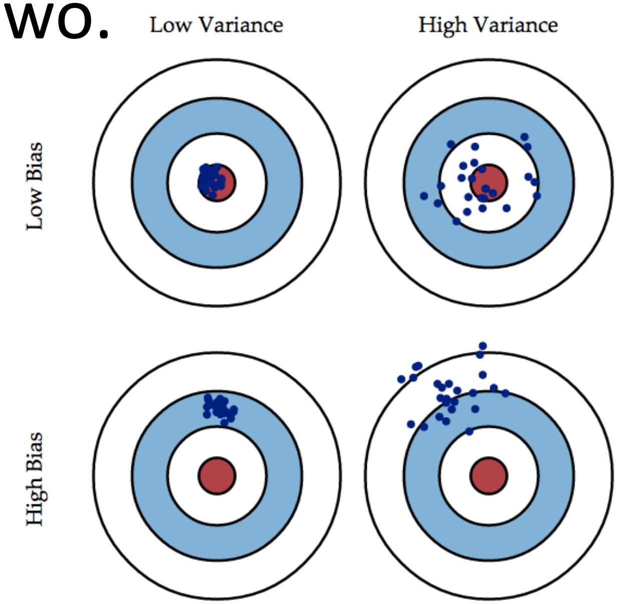


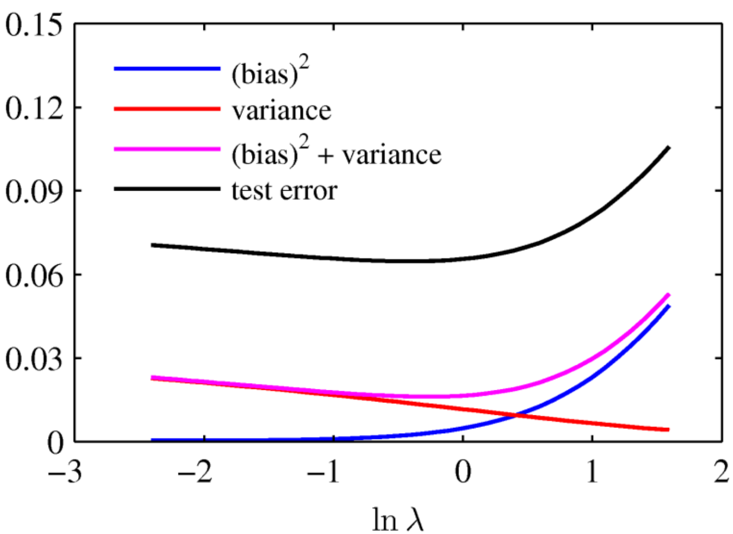
Learned $h(\mathbf{x}; D)$

$\mathbb{E}_D[h(\mathbf{x}; D)]$ vs. $\mathbb{E}[y|\mathbf{x}]$

The Bias-Variance Tradeoff

- An over-regularized model (large λ) will have a high bias and low variance.
- An under-regularized model (small λ) will have a high variance and low bias.
- It is important to find a good balance between the two.

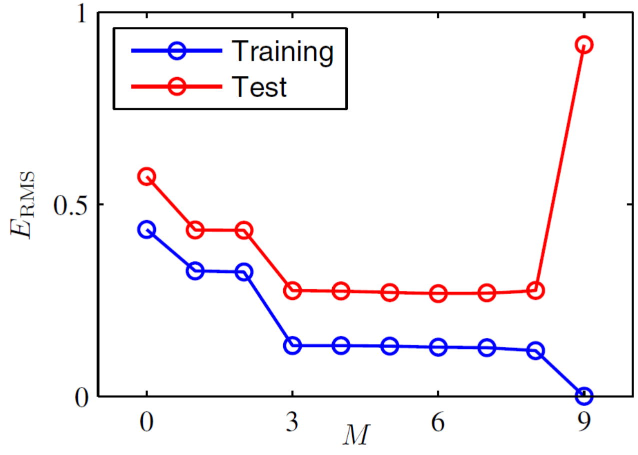


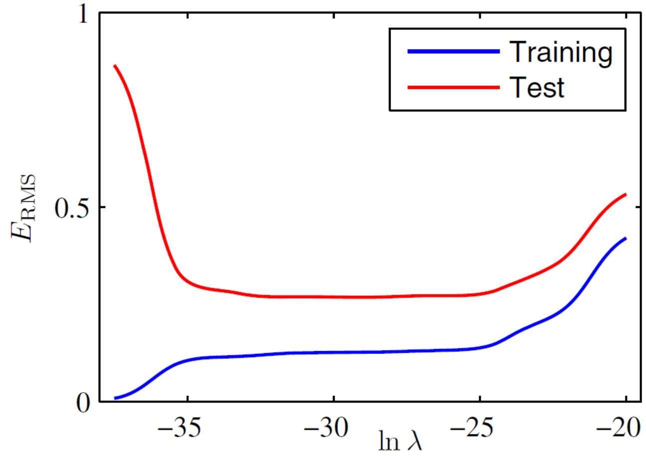


Validation for Model Selection

Model Selection

- For linear regression on a polynomial, which value of the polynomial order M should we choose?
- For regularized linear/logistic regression, which value of the regularization strength λ should we choose?
- Generally, given a set of models, how can we choose the optimal model?





Model Selection

- Generally, given a set of models, how can we choose the optimal model?

- Model: Learning algorithm, hyperparameter, etc.
  - Should be pre-determined before training
  - Fixed during training

- Parameters: Weights
  - e.g., $\mathbf{w}$ for linear/logistic regression
  - Updated during training

Model Selection

Idea #1: Choose a model that work best on the data

BAD: No regularization always works best on training data

Your Dataset

Idea #2: Split data into **train** and **test**; choose a model that work best on test data

BAD: No idea how it will perform on new data

train	test

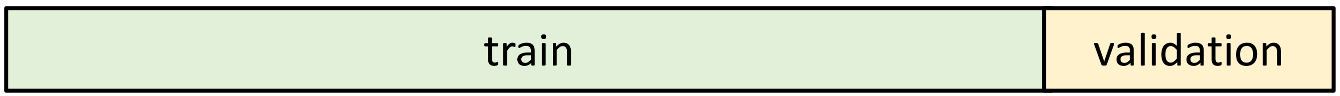
Idea #3: Split data into **train**, **val**, and **test**; choose a model on val and evaluate on test

Better!

train	validation	test

Hold-Out Validation

1. Randomly split D into D_{train} and D_{val} (e.g., 70% and 30%); D_{val} is the hold-out validation set.



2. Train each model M_i on D_{train} only, to get some hypothesis h_i.
3. Select and output the hypothesis h_i that had the smallest error on the hold-out validation set.

- Disadvantage:
  - Waste 30% of the data (less training examples available).

Slide credit: Justin Johnson

Model Selection

Your Dataset

Idea #4: **Cross-validation**: Split data into **folds**, try each fold as validation and average the results

fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

Useful for small datasets, but (unfortunately) not used too frequently in deep learning

K-Fold Cross-Validation

- Split dataset into K-folds
  - Take one fold (yellow) as validation and the rest of K-1 folds (green) for training.

Trial 1	fold 1	fold 2	fold 3	fold 4
Trial 2	fold 1	fold 2	fold 3	fold 4
Trial 3	fold 1	fold 2	fold 3	fold 4
Trial 4	fold 1	fold 2	fold 3	fold 4

- The final validation error is estimated as the average error rate.

K-Fold Cross-Validation

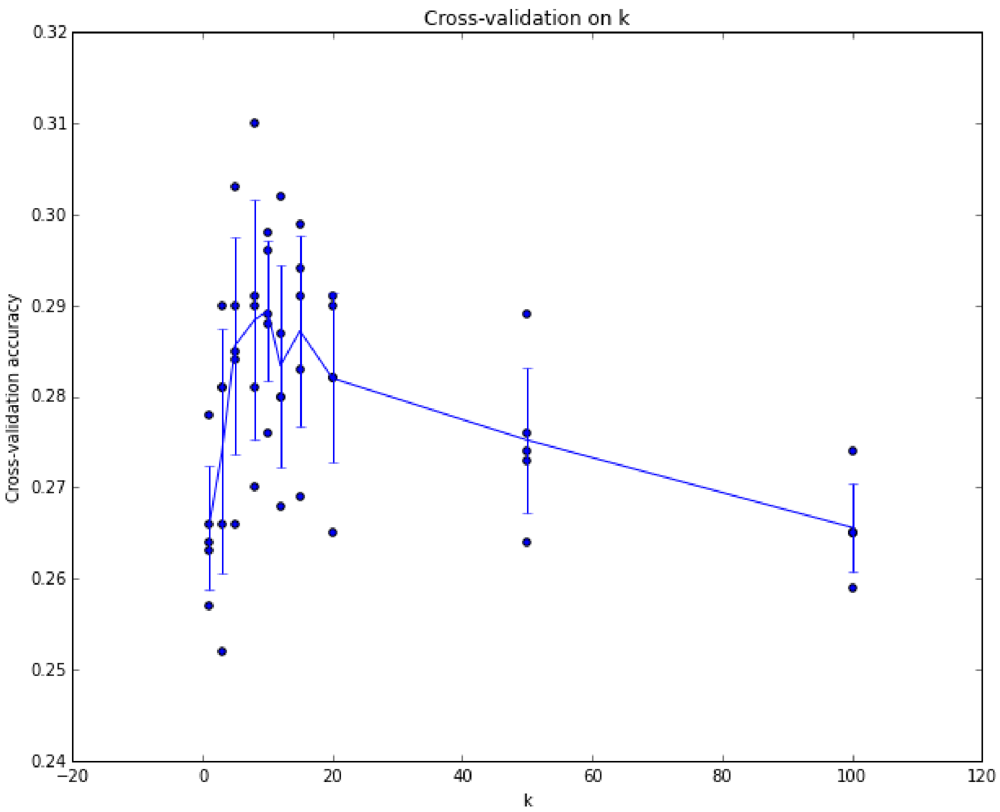
- Special case: $K = N$ (all data in D)
  - Leave-one-out cross-validation (LOOCV)
  - Expensive, but wastes least amount of training data for cross validation.
- Which K value should we use?
  - For large data, $K = 3$ might be enough.
  - For small amount of data, you may need LOOCV to utilize as many training examples as possible.
  - Popular choice of $K = 5, 10$
- Not used too frequently in deep learning
  - Simple hold-out validation is fast and good enough when the dataset is very large & training takes a long time.

Recap: k-NN Hyperparameters

- What is the best value of k to use?
- What is the best distance metric $D(\mathbf{x}, \mathbf{x}')$ to use?
  - These are **hyperparameters**.
    - C.f. Learning rate and regularization coefficient are also hyperparameters.
  - We set them at the start of the learning process, instead of learning from the training data.
- **Answer**: Very problem-dependent. In general, we need to try them all and see what works best for our data/task.
  - Need **validation** to find the best hyperparameters
    - (We will discuss validation in the lecture on model selection)

Slide credit: Justin Johnson

Example: k-Nearest Neighbors



Example of 5-fold cross-validation for the value of k.

Each point: single outcome.

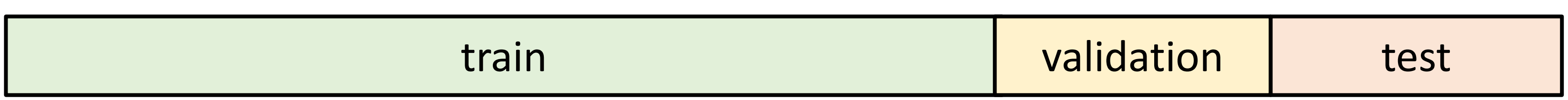
The line goes through the mean, bars indicated standard deviation

(Seems that k ~ 7 works best for this data)

Slide credit: Justin Johnson

Three-Way Data Splits

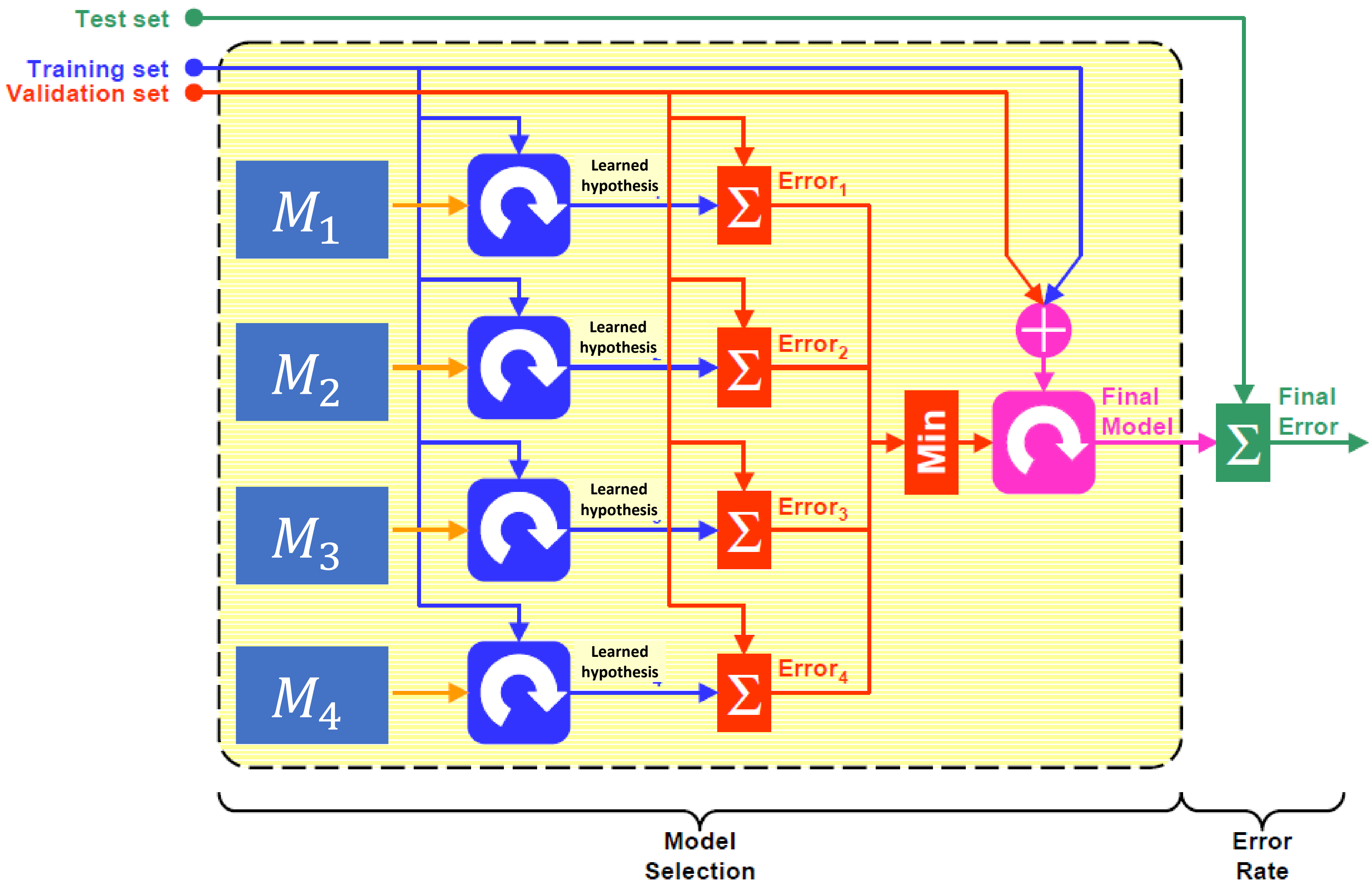
- If model selection and true error estimates are to be computed simultaneously, data needs to be divided into three disjoint sets.
- **Training set** to fit the parameters
  - Given a fixed hyperparameters
- **Validation set** to tune/choose the model and hyperparameters
- **Test set** to evaluate the final model performance
  - You must **NOT** tune the model on test set.
  - Test set is **NOT** for model selection.



ML Procedure

1. Divide the available data into training, validation, and test set
2. Select a model (and hyperparameters)
3. Train the model on the training set
4. Evaluate the model on the validation set
5. Repeat steps 2 through 4 with different models
6. Select the best model; optionally, train the model on both training and validation set
7. Assess the final model on the test set

ML Procedure Illustration



Next: Kernel Methods