EECS 545: Machine Learning

Lecture 7. Regularization and model selection

Honglak Lee and Michał Dereziński

01/31/2022

Logistics

- Project proposal due tomorrow:
  - …but you get 3 free late days. It won't consume your existing late days if you submit by Friday 2/4 11:55 PM.
  - However, if you submit later than Friday, each additional late day will deduct 1 point from your final project score.
  - See announcement on Canvas/Piazza for details
- HW2 due next Tuesday at 11:55 PM
  - See minor changes on Piazza changelog @128

Outline

- ML, MAP
  - Maximum Likelihood
  - MAP
- Bias-Variance Tradeoff
- Model selection
  - Cross validation

MLE vs. MAP

- Maximum Likelihood Estimation (MLE)
  - Objective: Log-likelihood

    $\log P(D|\mathbf{w})$
  - Example: linear regression (w/o regularization)

- Maximum a Posteriori (MAP)
  - Objective: Log-likelihood + Log-Prior

    $\log P(D|\mathbf{w}) + \log P(\mathbf{w})$
  - Example: Regularized linear regression

Maximum Likelihood

- Objective function (to maximize): log-likelihood
  - discriminative model:

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

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

IID: Independent and Identically Distributed

  - generative model:

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

Note: Today, we will look into the discriminative setting, but the theory/analysis will similarly apply to generative models too.

Maximum Likelihood

- Objective function (to maximize): log-likelihood

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

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

IID: Independent and Identically Distributed

- Example:
  - Linear regression (without regularization)
  - Logistic regression (without regularization)
- Problems: risk of overfitting

MAP

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

- Assumes prior distribution *P*(**w**)
- Point estimate using Bayes rule:

$$\arg\max_{\mathbf{w}} P(\mathbf{w}|D) = \arg\max_{\mathbf{w}} P(\mathbf{w}|D)P(\mathbf{w})$$

- Objective function to maximize:

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

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

MAP

- Isotropic Gaussian (e.g. L2 norm) is a popular prior (regularizer):

$$P(\mathbf{w}) = \mathcal{N}(0, \lambda^{-1}\mathbf{I})$$

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

- Example:
  - L2-regularized linear regression
  - L2-regularized logistic regression

Check: Gaussian Prior and L2 regularization

- Gaussian prior distribution for **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 log

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

"Gaussian Prior" for **w** and L2 regularization are equivalent.

MAP for Regularized Least Squares

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

Derive: $$\log \mathrm{P}(\mathbf{D} \mid \mathbf{w})\mathrm{P}(\mathbf{w}) = \sum_{n=1}^{N} \log P(y^{(n)}|\phi(\mathbf{x}^{(n)}), \mathbf{w}) + \log \mathrm{P}(\mathbf{w})$$
$$= \sum_{n=1}^{N} \log \left(\sqrt{\frac{\beta}{2\pi}} \exp(-\frac{\beta}{2}||y^{(n)} - \mathbf{w}^T\phi(\mathbf{x}^{(n)})||^2) \right) + const - \frac{\lambda}{2}\|\mathbf{w}\|^2$$
$$= \frac{N}{2}\log\beta - \frac{N}{2}\log 2\pi - \sum_{n=1}^{N} \frac{\beta}{2}||y^{(n)} - \mathbf{w}^T\phi(\mathbf{x}^{(n)})||^2 + const - \frac{\lambda}{2}\|\mathbf{w}\|^2$$

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

Solving Regularized Least Squares

- Consider the error function:

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

Data term + Regularization term

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

Penalize large coefficient values

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

Remark: $\beta = 1$ for simplicity

λ is called the regularization coefficient.

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

Derivation

Objective function

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

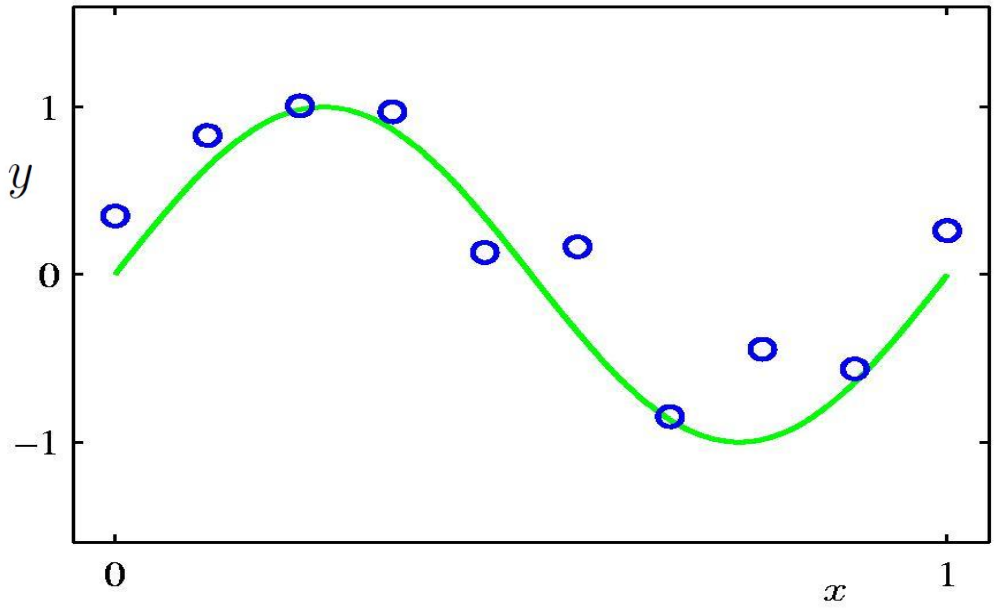
$$= \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}^T\Phi^T\mathbf{y} + \frac{1}{2}\mathbf{y}^T\mathbf{y} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

Compute gradient and set it zero:

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

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

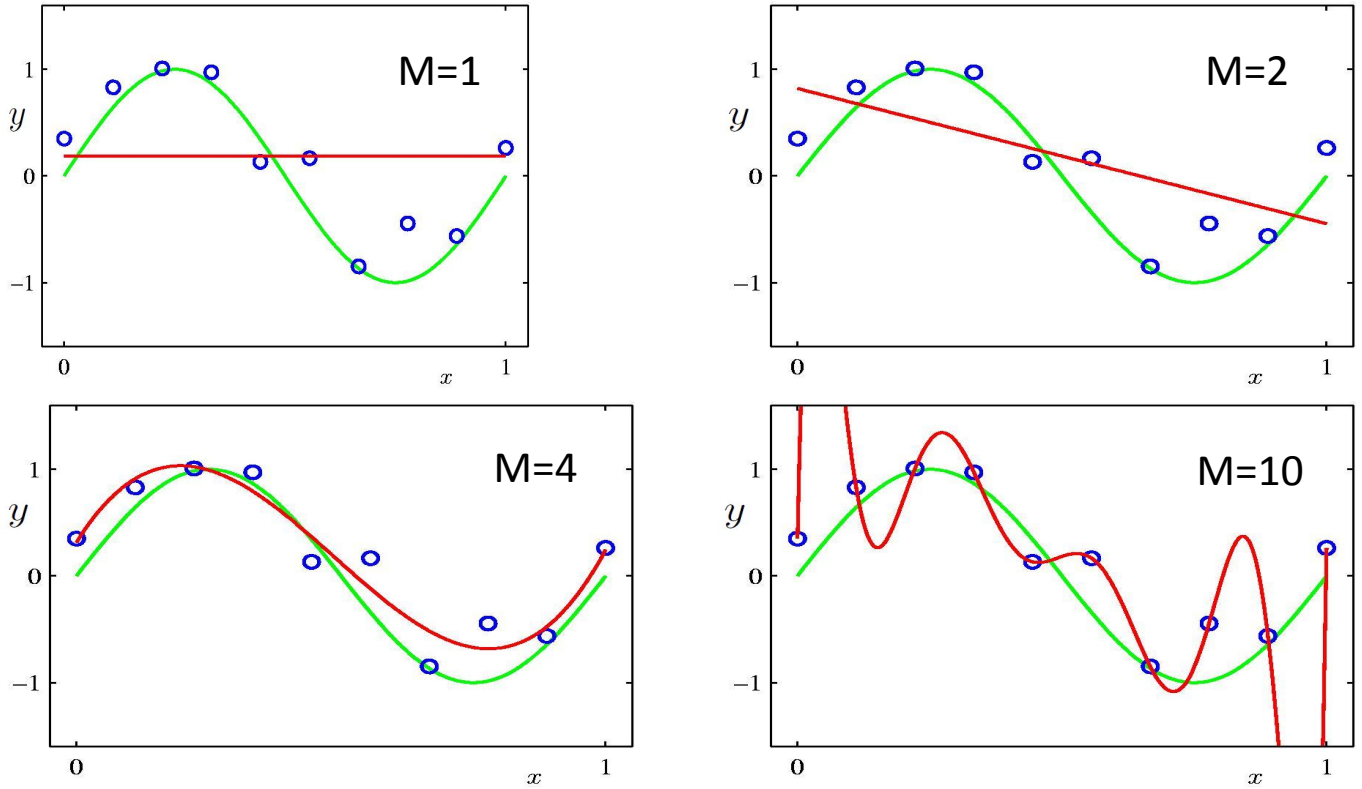
Revisiting Polynomial Curve Fitting



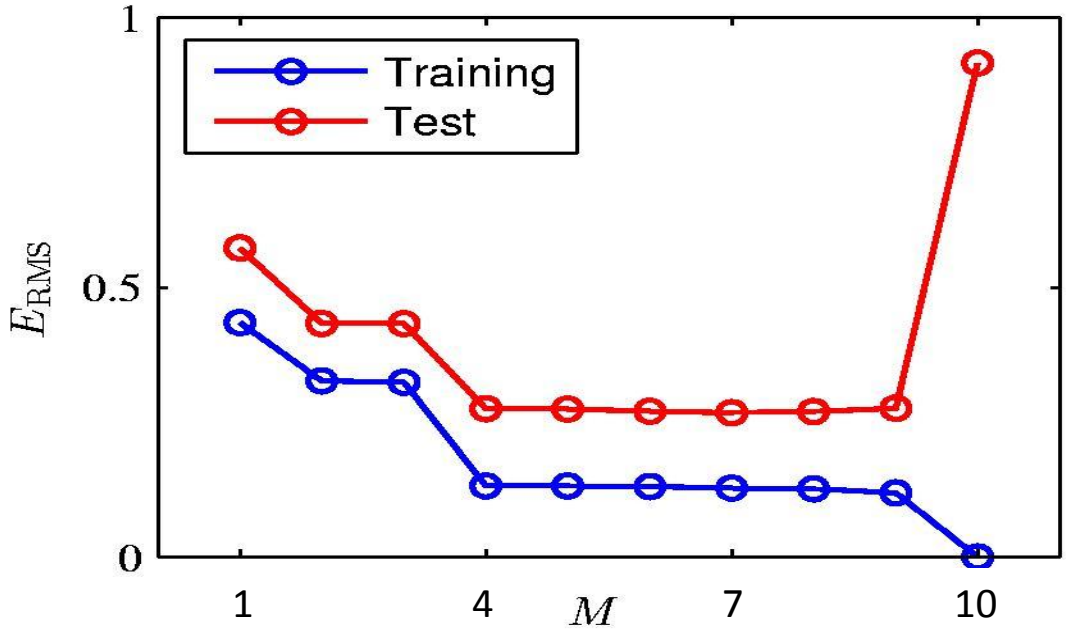
$$h(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_{M-1} x^{M-1} = \sum_{j=0}^{M-1} w_j x^j$$

Maximum Likelihood (in Linear Regression)

- Choosing the right complexity is important
  - Watch out for underfitting/overfitting

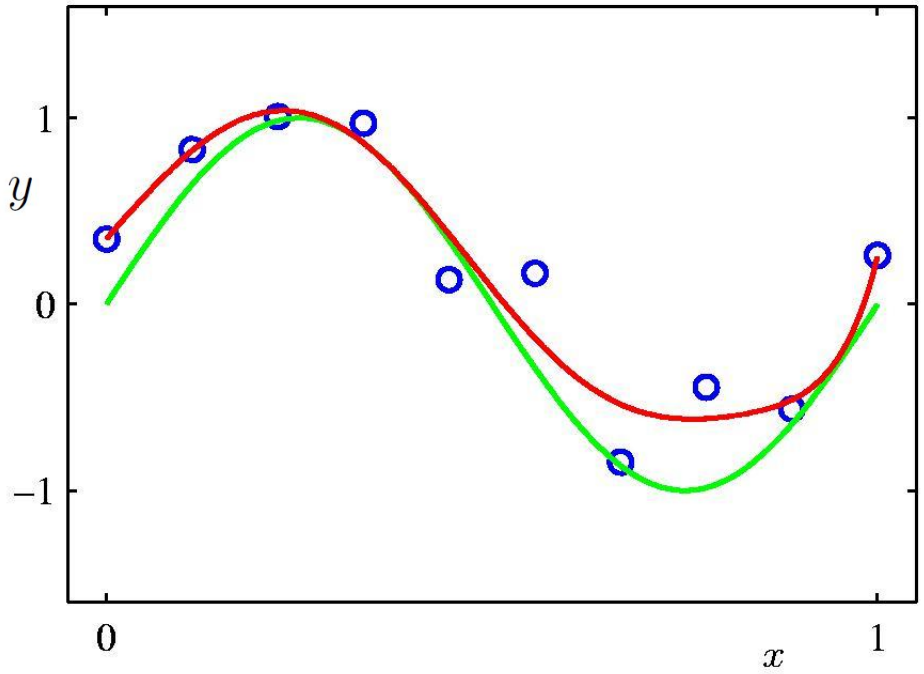


Underfitting vs. overfitting

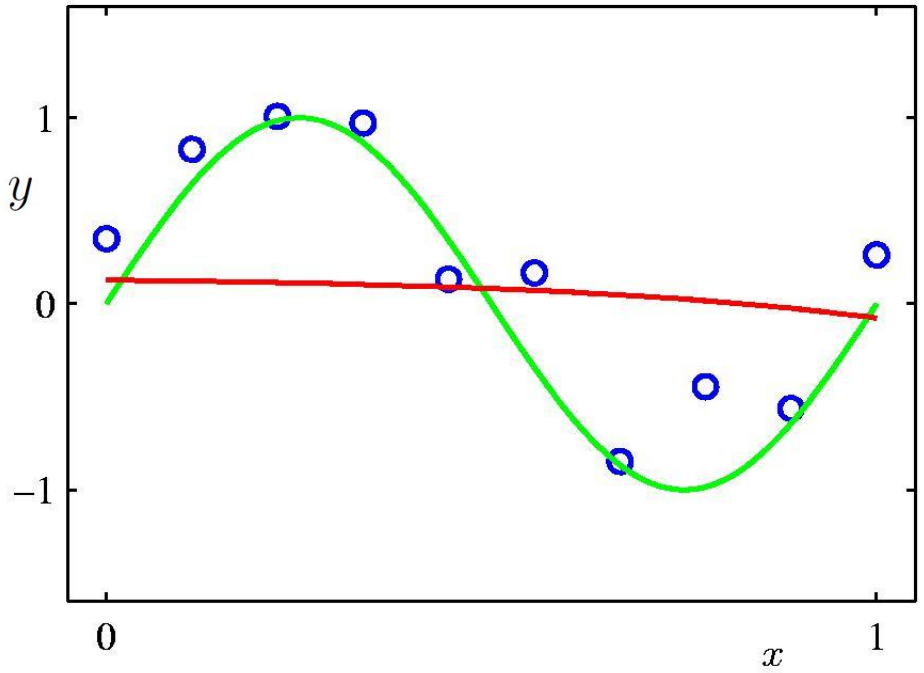


Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

L2 Regularization: $\lambda = 0.001$

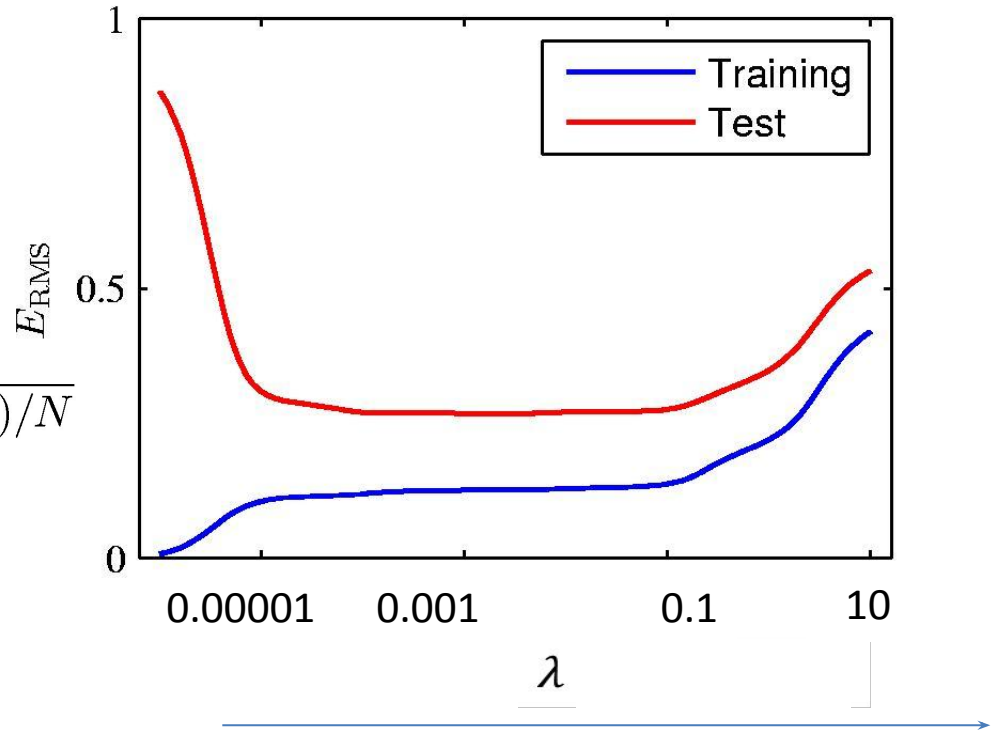


L2 Regularization: $\lambda = 10$



L2 Regularization: E_{RMS} vs λ

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



Larger regularization

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

	λ=0	λ=0.001	λ=10
$w_0^\star$	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: L2 regularized linear regression

- Simple modification of linear regression
- L2 regularization controls the tradeoff between "fitting error" and "complexity".
  - Small L2 regularization results in complex models (but with risk of overfitting)
  - Large L2 regularization results in simple models (but with risk of underfitting).

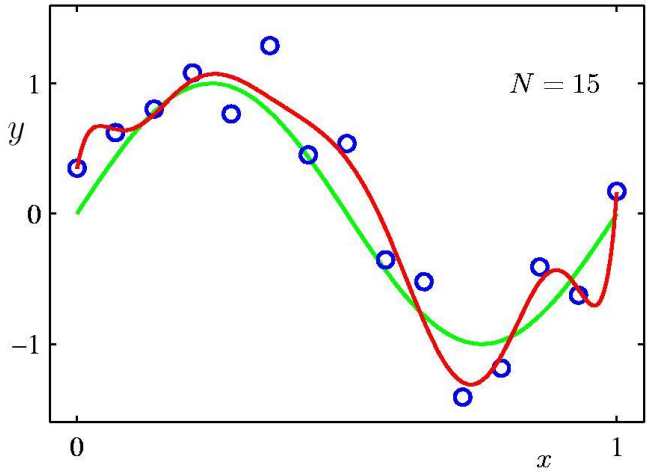
- It is important to find an optimal regularization that balances between the two.

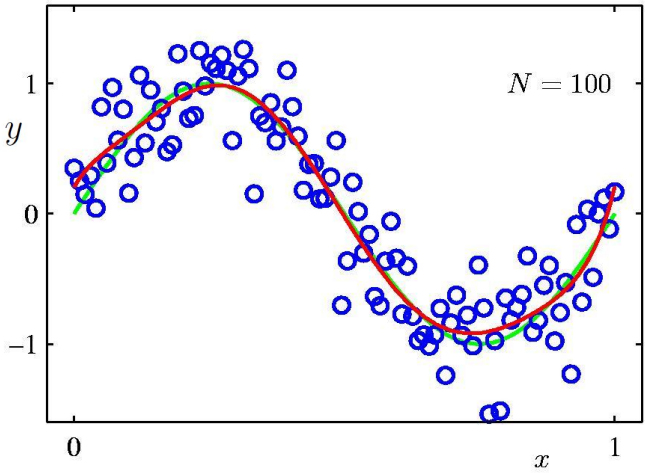
How can we avoid overfitting?

- More training data
  - Always helps
- Regularization (e.g. MAP)
  - Penalize complex models

Recap: More training data

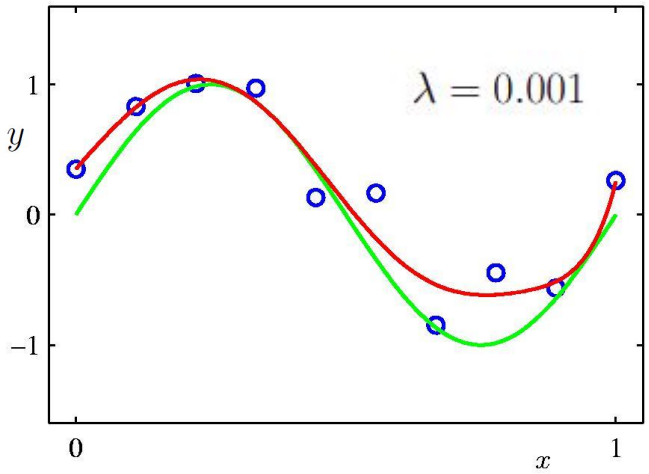
- The more complicated the model, the more it benefits from more data (by avoiding overfitting)
  - Example: 9th order polynomial

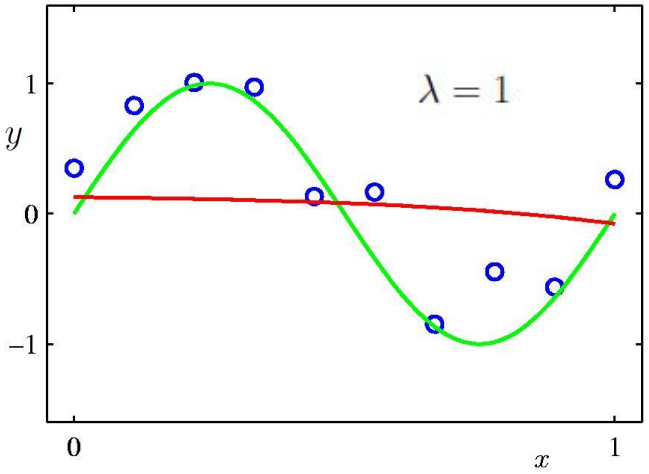




Recap: Regularization

- Regularization can implicitly control the complexity of models
  - Example: 9th 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

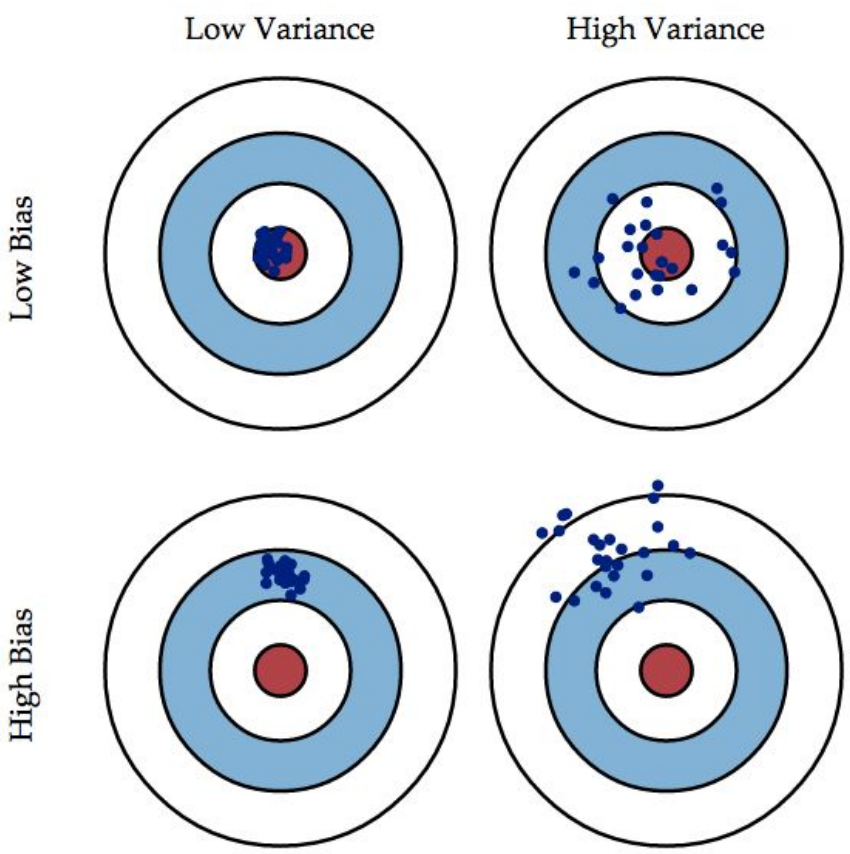
The Bias-Variance Decomposition

- Setting:
  - Given a distribution of $P(\mathbf{x}, y)$
  - Sample training data

$$D_{train} = \{(\mathbf{x}^{(n)}, y^{(n)}) : n = 1, ..., N\} \sim P(\mathbf{x}, y)$$

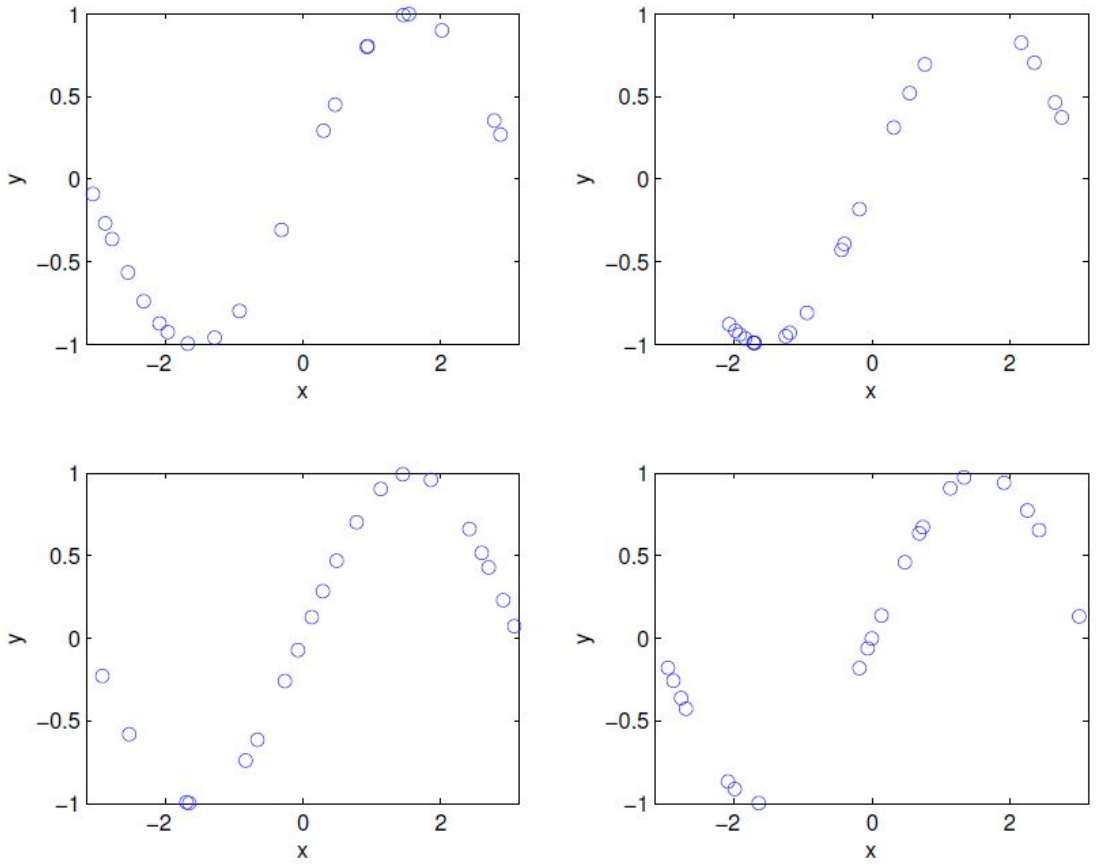
  - Train a learning algorithm on D
- Depending on samples, learning algorithm can still give different results (MLE, MAP, etc.)
- Goal: We want to learn a model with
  - Small bias (i.e. how well a model fits the data on average?)
  - Small variance (i.e. how stable a model is wrt data samples?)

Bias and Variance



The Bias-Variance Decomposition

- Example: samples from the sinusoidal function y=sin(x)



The Bias-Variance Decomposition (advanced)

- Expected squared loss:

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

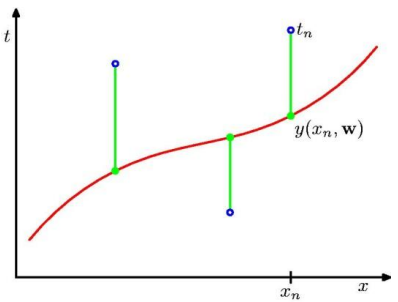
$$\mathbb{E}[L] = \int \{h(\mathbf{x}) - \mathbb{E}[y|\mathbf{x}]\}^2 p(\mathbf{x}) d\mathbf{x} + \int\int \{\mathbb{E}[y|\mathbf{x}] - y\}^2 p(\mathbf{x}, y) d\mathbf{x} dy$$

  - 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 about the first term?

E[y|**x**]



The Bias-Variance Decomposition

- Suppose we were given multiple data sets, each of size N. Any particular data set (sampled from a distribution), D, will give a particular function h(x;D). We then have

$$\mathbb{E}_D \boxed{\int \{h(\mathbf{x}; D) - \mathbb{E}[y|\mathbf{x}]\}^2 p(\mathbf{x}) d\mathbf{x}}$$

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

The Bias-Variance Decomposition

- Expected 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

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

$$(\text{bias})^2 =$$

$$\text{variance} =$$

$$\text{noise} =$$

The Bias-Variance Decomposition

- Expected 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

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

$$(\text{bias})^2 = \int \{\mathbb{E}_D[h(\mathbf{x}; D)] - \mathbb{E}[y|\mathbf{x}]\}^2 p(\mathbf{x}) d\mathbf{x}$$

$$\text{variance} = \int \mathbb{E}_D\left[\{h(\mathbf{x}; D) - \mathbb{E}_D[h(\mathbf{x}; D)]\}^2\right] p(\mathbf{x}) d\mathbf{x}$$

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

Proof: Details

D: training data; x: test example; y: label of x

$$\mathbb{E}[L] = \mathbb{E}_{\mathbf{x},y,D}\left[(h(\mathbf{x};D)-y)^2\right]$$

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

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

because this second term doesn't depend on the model

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

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

$$= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_D\left[\{h(\mathbf{x};D)-\mathbb{E}_D[h(\mathbf{x};D)]\}^2\right]\right]$$

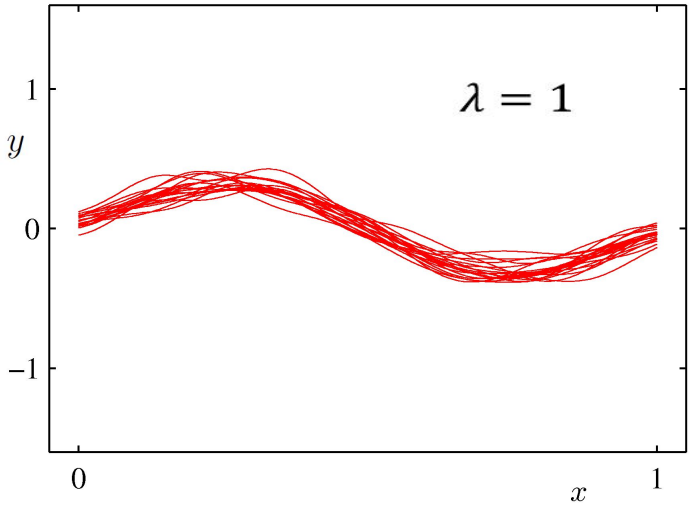
$$+2\mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_D\left[\{h(\mathbf{x};D)-\mathbb{E}_D[h(\mathbf{x};D)]\}\{\mathbb{E}_D[h(\mathbf{x};D)]-\mathbb{E}[y|\mathbf{x}]\}\right]\right]$$

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

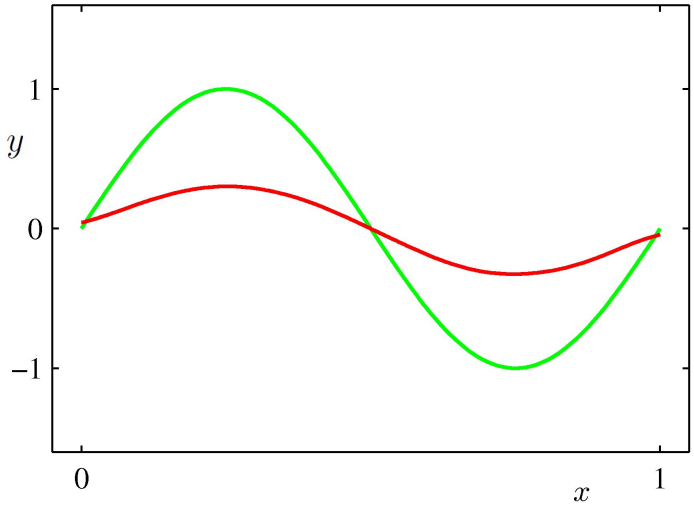
$$= \underbrace{\mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_D\left[\{h(\mathbf{x};D)-\mathbb{E}_D[h(\mathbf{x};D)]\}^2\right]\right]}_{\text{Variance}} + \underbrace{\mathbb{E}_{\mathbf{x}}\left[\{\mathbb{E}_D[h(\mathbf{x};D)]-\mathbb{E}[y|\mathbf{x}]\}^2\right]}_{\text{Bias}} + const$$

Example: regularized linear regression

- Example: 25 data sets from the sinusoidal, varying the degree of regularization, λ.



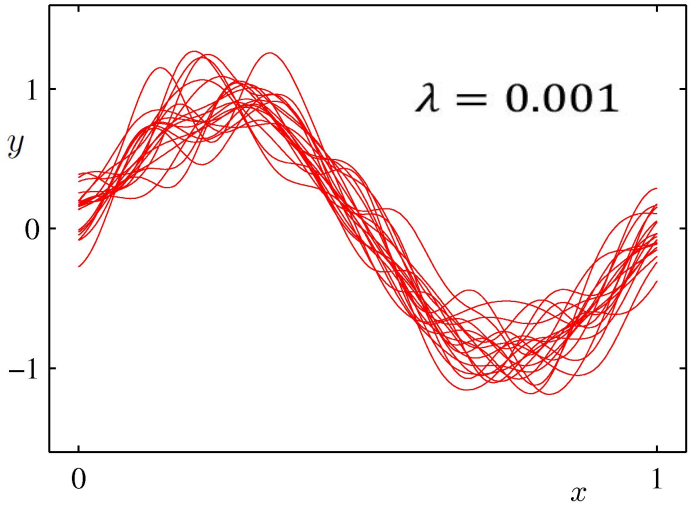
Samples of $h(\mathbf{x}; D)$



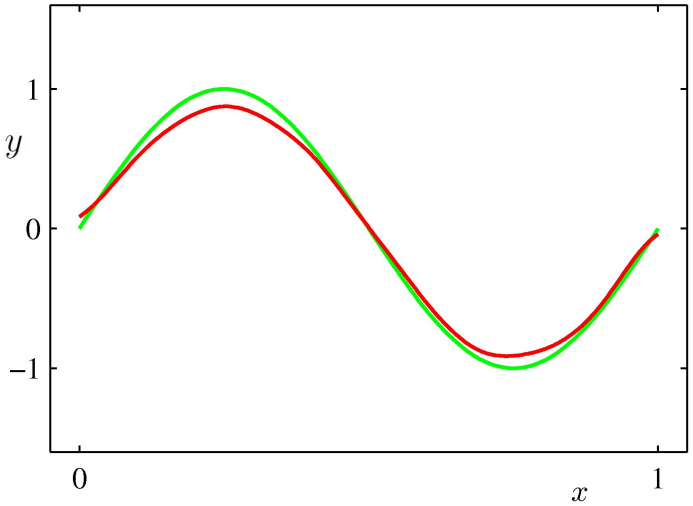
$\mathbb{E}_D[h(\mathbf{x}; D)]$ vs $\mathbb{E}[y|\mathbf{x}] = \mathbb{E}_D[y|\mathbf{x}]$

Example: regularized linear regression

- Example: 25 data sets from the sinusoidal, varying the degree of regularization, λ.



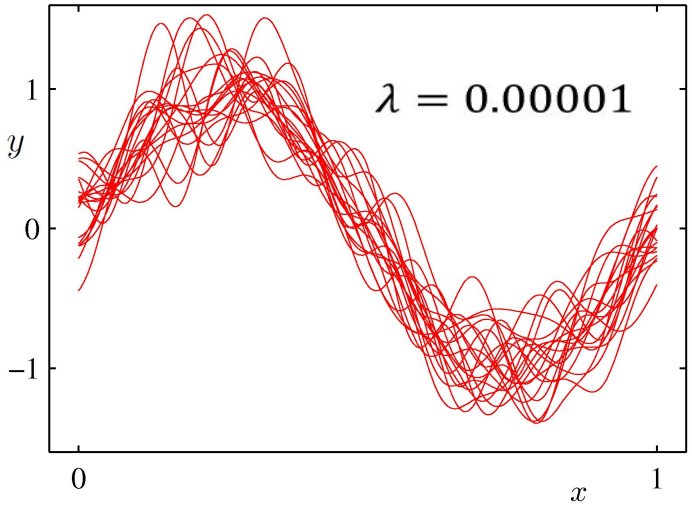
Samples of $y(\mathbf{x}; \mathcal{D})$



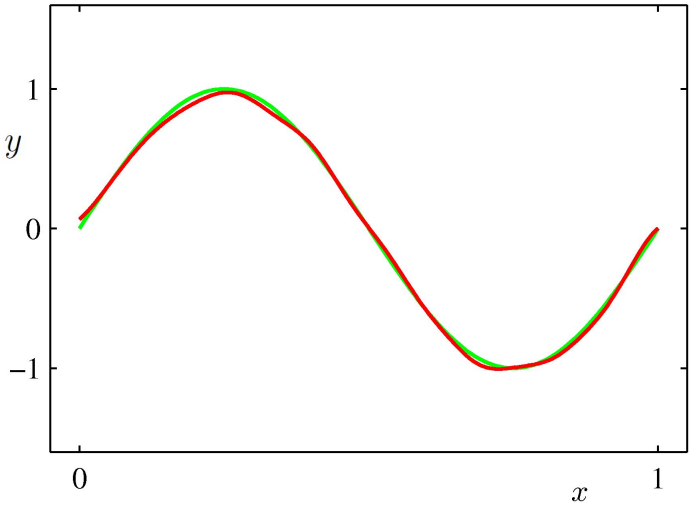
$\mathbb{E}_D[h(\mathbf{x}; D)]$ vs $\mathbb{E}[y|\mathbf{x}] = \mathbb{E}_D[y|\mathbf{x}]$

Example: regularized linear regression

- Example: 25 data sets from the sinusoidal, varying the degree of regularization, λ.



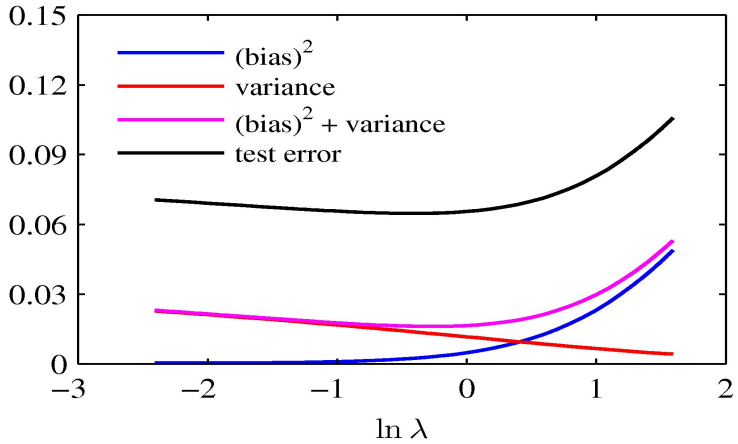
Samples of $y(\mathbf{x}; \mathcal{D})$



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

The Bias-Variance Trade-off

- 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.
  - typically done by cross validation (will be covered later)



Model Selection

Choosing the right model

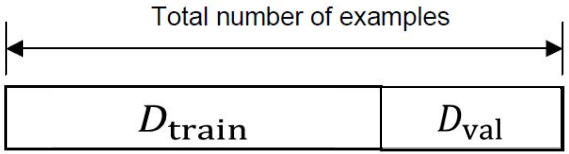
- For polynomial curve fitting (least squares), which value of M should we choose?
- For regularized least squares / logistic regression, which λ value should we choose?
- Generally, given a set of models $M = \{M_1, M_2, \ldots, M_d\}$, how can we choose optimal M_*?
  - Model:
    - Class (or set) of hypotheses: learning algorithm, hyperparameters, etc.
    - Fixed during training
  - Parameters:
    - Aka hypothesis (e.g. **w** for logistic regression/linear regression)
    - Can be trained based on data

Simple idea that doesn't work

- Given data D, train each model M_i on D, to get a hypothesis h_i (for model i)
- Pick the hypothesis with the smallest training error
  - Problematic: this leads to overfitting

Cross validation

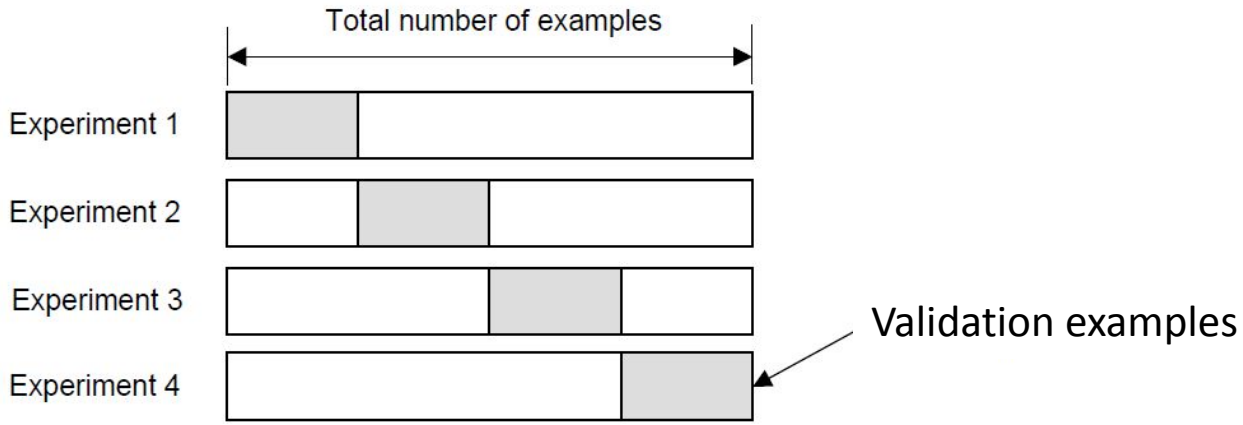
- Hold out cross validation
  - 1. Randomly split D into D_{train} and D_{val}
    - D_{train} should have more than D_{val} – something like 70%:30% is a typical split
    - D_{val} also known as the hold-out validation set



  - 2. Train each model M_i on D_{train} only to get some hypothesis h_i
  - 3. Select and output hypothesis h_i that had the smallest error on the hold out validation set
- Disadvantage:
  - Wastes the data in D_{val} (can't train on it)
  - Some data only used only for training, some only for validation

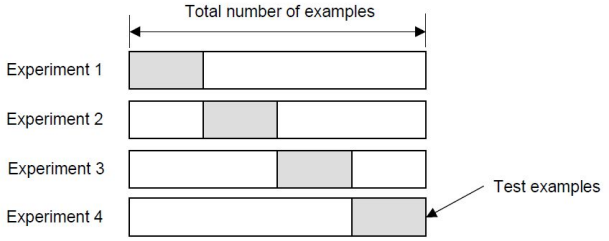
K-fold Cross validation

- Create a K-fold partition of the dataset
  - For each of K experiments, use K-1 folds for training and the remaining one for validating



- The true error is estimated as the average error rate

K fold cross validation

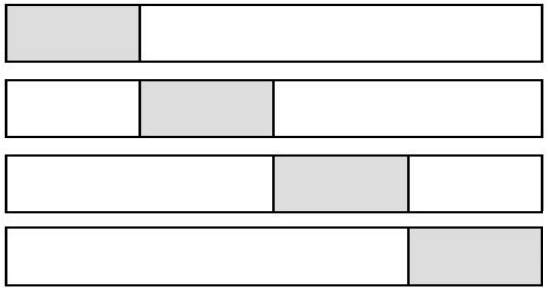


- 1. Randomly split D into K disjoint subsets of N/K training samples each
  - Subsets are named $D_1, \ldots, D_K$
- 2. For each model M_i, we evaluate as follows:
  - For each fold (k = 1, …, K):
    - Train M_i on $D_1 \cup \ldots \cup D_{k-1} \cup D_{k+1} \cup \ldots \cup D_K$ (i.e. train on all data except D_k) to get some hypothesis $h_i(k)$
    - Test the hypothesis $h_i(k)$ on D_k to get error/loss $L_i(k)$
    - Estimated generalization error of M_i is average of all errors:

$$\hat{\epsilon}_i = \frac{1}{K} \sum_k \epsilon_i(k)$$

- 3. Pick M_* with the lowest estimated generalization error $\hat{\epsilon}_i$ and retrain M_* on the entire training set S.
  - Resulting hypothesis is our final answer

K-fold Cross validation



- Special case: K=N
  - Called, leave-one-out cross validation (LOO CV)
  - Expensive, but wastes least amount of training data for cross validation. Could be useful when the data size is not big.
  - Computationally unsuitable when you have a large data

- Which K value should we use?
  - Popular choice of K = 10, 5
  - For large data, then K = 3 could be enough.
  - For small amount of data, you may need LOOCV to utilize as many training examples as possible.

Three way data splits

- If model selection and true error estimates are to be computed simultaneously, the data needs to be divided into three disjoint sets

- **Training set**: a set of examples used for learning: to fit the parameters of the classifier
  - Used for training parameters (**w** in logistic regression) given a fixed hyperparameters
- **Validation set**: a set of examples used to tune the hyperparameters of a classifier
  - we would use the validation set to find the "optimal" hyperparameters
  - E.g., L2 regularization parameter for L2 logistic regression
- **Test set**: a set of examples used only to assess the performance of a fully-trained classifier
  - After assessing the final model with the test set, **YOU MUST NOT** further tune the model
  - i.e., test set must be used only for evaluation, **NOT** for "tuning" your models & hyperparameters.

Practical Recipe for Training, Model Selection, Evaluation (all put together)

1. Divide the available data into training, validation and test set

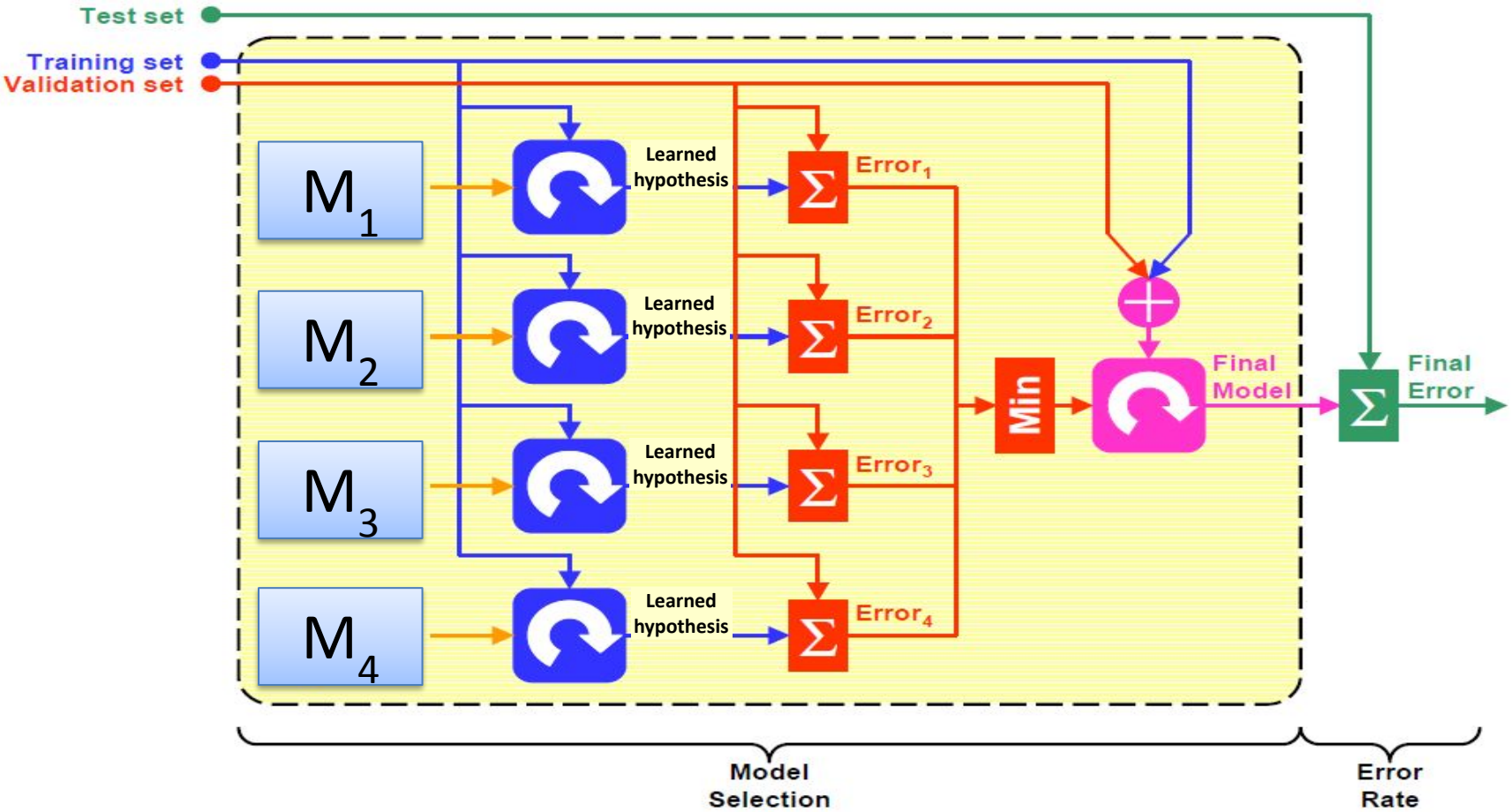
2. Repeat the following steps with different models (and hyperparameters)
   a. Select a model (and hyperparameters)
   b. Train the model using the training set
   c. Evaluate the model using the validation set

3. Select the best model (and hyperparameter)

4. (optional) Re-train the best model above using data from the training set and validation set combined

5. Assess this final model using the test set

Procedure Illustration



Quiz and what's coming up

- https://tinyurl.com/eecs545-07

- Beginning next lecture and up through next week, we will talk about kernel methods
- Remember to submit your project proposals!