Overfitting, Bias / Variance Analysis

Professor Ameet Talwalkar

Slide Credit: Professor Fei Sha

Outline

Announcements

- HW2 (and remaining HW1) will be returned today after class (see Nikos and Amogh)
- HW3 and HW4 due on Thursday
- No new HW assigned on Thursday (to give you time to work on project proposal)
- Project proposal guideline posted on course website last week

CS260: Project Proposal

Professor Ameet Talwalkar

Due: 11/5/15

Project proposals should be 1-2 pages long and contain the following information:

- **Motivation**: What high-level problem are you studying? Why is it important?
- **Background**: What previous work exists on your topic? How does your proposed project differ from these existing works? Note: If you have prior experience working on this problem, please clearly describe how your proposed work for this class project differs from your prior work.
- **Proposed Work**: What do you plan to do? What methods will you use and why will you use them?
- **Timeline**: What is your timeline for performing this work?
- **Deliverables / Evaluation**: What is the expected outcome of your work? How will you evaluate the quality of your work?
- **Data**: If your project involves data, what data are you using? How did you obtain this data? Why is this data interesting / relevant to your problem? What computing resources will you be using to analyze this data?
- **Software Tools / Libraries**: What (if any) software will be used for this project? Please describe any third-party software libraries you plan to use.
- **Team**: Who is working on this project (at most two students can work on a project)?
- **Prior Discussion**: You are required to discuss this project with Professor Talwalkar or one of the TAs prior to submitting your proposal. Please list who you spoke with and when. Note: You are also encouraged to speak with us as you continue working on the project if you have questions.

Outline

Linear regression 1D

Setup

- Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$ (covariates, predictors, features, etc)
- Output: $y \in \mathbb{R}$ (responses, targets, outcomes, outputs, etc)
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathsf{N}\}$
- Model: $f : \boldsymbol{x} \rightarrow y$, with $f(\boldsymbol{x}) = w_0 + \sum_d w_d x_d = w_0 + \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}$

  We also sometimes call $\tilde{\boldsymbol{w}} = [w_0 \; w_1 \; w_2 \; \cdots \; w_{\mathsf{D}}]^{\mathrm{T}}$ parameters too!

Linear regression 1D

Setup

- Input: $\boldsymbol{x} \in \mathbb{R}^{\mathrm{D}}$ (covariates, predictors, features, etc)
- Output: $y \in \mathbb{R}$ (responses, targets, outcomes, outputs, etc)
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, \mathrm{N}\}$
- Model: $f : \boldsymbol{x} \rightarrow y$, with $f(\boldsymbol{x}) = w_0 + \sum_d w_d x_d = w_0 + \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}$

  We also sometimes call $\tilde{\boldsymbol{w}} = [w_0 \; w_1 \; w_2 \; \cdots \; w_{\mathrm{D}}]^{\mathrm{T}}$ parameters too!

Least Mean Squares (LMS) Objective: Minimize squared difference on training data (or residual sum of squares)

$$RSS(\tilde{\boldsymbol{w}}) = \sum_n [y_n - f(\boldsymbol{x}_n)]^2 = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2$$

1D Solution: Identify stationary points by taking derivative with respect to parameters and setting to zero, yielding 'normal equations'

LMS when $\boldsymbol{x}$ is D-dimensional

$RSS(\tilde{w})$ in matrix form

$$RSS(\tilde{\boldsymbol{w}}) = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2 = \sum_n [y_n - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_n]^2$$

where we have redefined some variables (by augmenting)

$$\tilde{\boldsymbol{x}} \leftarrow [1 \; x_1 \; x_2 \; \ldots \; x_{\mathsf{D}}]^{\mathrm{T}}, \quad \tilde{\boldsymbol{w}} \leftarrow [w_0 \; w_1 \; w_2 \; \ldots \; w_{\mathsf{D}}]^{\mathrm{T}}$$

LMS when x is D-dimensional

$RSS(\tilde{w})$ in matrix form

$$RSS(\tilde{\boldsymbol{w}}) = \sum_n [y_n - (w_0 + \sum_d w_d x_{nd})]^2 = \sum_n [y_n - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_n]^2$$

where we have redefined some variables (by augmenting)

$$\tilde{\boldsymbol{x}} \leftarrow [1 \; x_1 \; x_2 \; \ldots \; x_{\mathsf{D}}]^{\mathrm{T}}, \quad \tilde{\boldsymbol{w}} \leftarrow [w_0 \; w_1 \; w_2 \; \ldots \; w_{\mathsf{D}}]^{\mathrm{T}}$$

which leads to

$$\begin{aligned} RSS(\tilde{\boldsymbol{w}}) &= \sum_n (y_n - \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_n)(y_n - \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}}) \\ &= \sum_n \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{x}}_n \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} - 2 y_n \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \tilde{\boldsymbol{w}} + \text{const.} \\ &= \left\{ \tilde{\boldsymbol{w}}^{\mathrm{T}} \left(\sum_n \tilde{\boldsymbol{x}}_n \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \right) \tilde{\boldsymbol{w}} - 2 \left(\sum_n y_n \tilde{\boldsymbol{x}}_n^{\mathrm{T}} \right) \tilde{\boldsymbol{w}} \right\} + \text{const.} \end{aligned}$$

$RSS(\tilde{\boldsymbol{w}})$ in new notation

Design matrix and target vector

$$\tilde{\boldsymbol{X}} = \begin{pmatrix} \tilde{\boldsymbol{x}}_1^{\mathrm{T}} \\ \tilde{\boldsymbol{x}}_2^{\mathrm{T}} \\ \vdots \\ \tilde{\boldsymbol{x}}_{\mathsf{N}}^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{\mathsf{N} \times (D+1)}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{\mathsf{N}} \end{pmatrix}$$

Compact expression

$$RSS(\tilde{\boldsymbol{w}}) = ||\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}} - \boldsymbol{y}||_2^2 = \left\{ \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - 2 \left(\tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y} \right)^{\mathrm{T}} \tilde{\boldsymbol{w}} \right\} + \text{const}$$

Solution in matrix form

Compact expression

$$RSS(\tilde{\boldsymbol{w}}) = ||\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}} - \boldsymbol{y}||_2^2 = \left\{ \tilde{\boldsymbol{w}}^{\mathrm{T}} \tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{w}} - 2 \left(\tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y} \right)^{\mathrm{T}} \tilde{\boldsymbol{w}} \right\} + \text{const}$$

Gradients of Linear and Quadratic Functions

- $\nabla_{\boldsymbol{x}} \boldsymbol{b}^{\top} \boldsymbol{x} = \boldsymbol{b}$
- $\nabla_{\boldsymbol{x}} \boldsymbol{x}^{\top} \boldsymbol{A} \boldsymbol{x} = 2\boldsymbol{A}\boldsymbol{x}$ (symmetric $\boldsymbol{A}$)

Normal equation

$$\nabla_{\tilde{\boldsymbol{w}}} RSS(\tilde{\boldsymbol{w}}) \propto \tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}} \boldsymbol{w} - \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y} = 0$$

This leads to the least-mean-square (LMS) solution

$$\tilde{\boldsymbol{w}}^{LMS} = \left(\tilde{\boldsymbol{X}}^{\mathrm{T}} \tilde{\boldsymbol{X}} \right)^{-1} \tilde{\boldsymbol{X}}^{\mathrm{T}} \boldsymbol{y}$$

Practical concerns

Bottleneck of computing the LMS solution

$$\boldsymbol{w} = \left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\right)^{-1}\tilde{\boldsymbol{X}}\boldsymbol{y}$$

Matrix multiply of $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} \in \mathbb{R}^{(\mathsf{D}+1)\times(\mathsf{D}+1)}$
Inverting the matrix $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$

Scalable methods

- Batch gradient descent
- Stochastic gradient descent

Stochastic gradient descent

Widrow-Hoff rule: update parameters using one example at a time

- Initialize $\tilde{\boldsymbol{w}}$ to $\tilde{\boldsymbol{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. random choose a training a sample $\boldsymbol{x}_t$
  2. Compute its contribution to the gradient

$$\boldsymbol{g}_t = (\tilde{\boldsymbol{x}}_t^{\mathrm{T}} \tilde{\boldsymbol{w}}^{(t)} - y_t)\tilde{\boldsymbol{x}}_t$$

  3. Update the parameters $\tilde{\boldsymbol{w}}^{(t+1)} = \tilde{\boldsymbol{w}}^{(t)} - \eta \boldsymbol{g}_t$

Stochastic gradient descent

Widrow-Hoff rule: update parameters using one example at a time

- Initialize $\tilde{\boldsymbol{w}}$ to $\tilde{\boldsymbol{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. random choose a training a sample $\boldsymbol{x}_t$
  2. Compute its contribution to the gradient

$$\boldsymbol{g}_t = (\tilde{\boldsymbol{x}}_t^{\mathrm{T}} \tilde{\boldsymbol{w}}^{(t)} - y_t)\tilde{\boldsymbol{x}}_t$$

  3. Update the parameters $\tilde{\boldsymbol{w}}^{(t+1)} = \tilde{\boldsymbol{w}}^{(t)} - \eta \boldsymbol{g}_t$
  4. $t \leftarrow t + 1$

Stochastic gradient descent

Widrow-Hoff rule: update parameters using one example at a time

- Initialize $\tilde{\boldsymbol{w}}$ to $\tilde{\boldsymbol{w}}^{(0)}$ (anything reasonable is fine); set $t = 0$; choose $\eta > 0$
- Loop *until convergence*
  1. random choose a training a sample $\boldsymbol{x}_t$
  2. Compute its contribution to the gradient

$$\boldsymbol{g}_t = (\tilde{\boldsymbol{x}}_t^{\mathrm{T}} \tilde{\boldsymbol{w}}^{(t)} - y_t)\tilde{\boldsymbol{x}}_t$$

  3. Update the parameters $\tilde{\boldsymbol{w}}^{(t+1)} = \tilde{\boldsymbol{w}}^{(t)} - \eta \boldsymbol{g}_t$
  4. $t \leftarrow t + 1$

How does the complexity per iteration compare with gradient descent?

- $O(\mathrm{ND})$ for gradient descent versus $O(\mathrm{D})$ for SGD

What if $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$ is not invertible

Can you think of any reasons why that could happen?

Answer 1: N $<$ D. Intuitively, not enough data to estimate all the parameters.

Answer 2: $\boldsymbol{X}$ columns are not linearly independent. Intuitively, there are two features that are perfectly correlated. In this case, solution is not unique.

Ridge regression

For $\tilde{X}^{\mathrm{T}}\tilde{X}$ that is not invertible

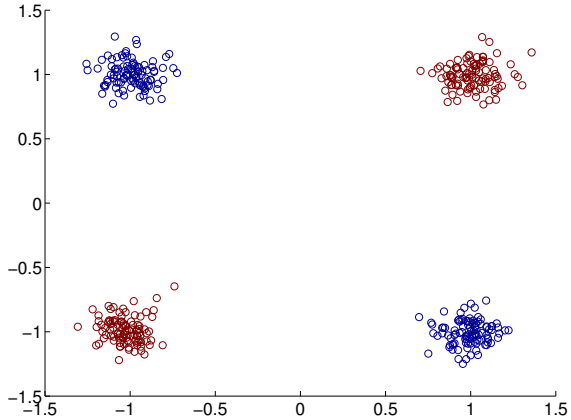
$$\tilde{\boldsymbol{w}} = \left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}} + \lambda \boldsymbol{I}\right)^{-1} \tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}$$

This is equivalent to adding an extra term to $RSS(\tilde{\boldsymbol{w}})$

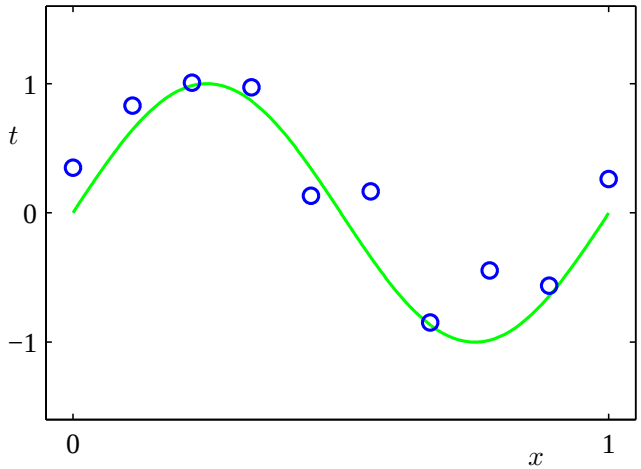
$$\overbrace{\frac{1}{2}\left\{\tilde{\boldsymbol{w}}^{\mathrm{T}}\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}\tilde{\boldsymbol{w}} - 2\left(\tilde{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{y}\right)^{\mathrm{T}}\tilde{\boldsymbol{w}}\right\}}^{RSS(\tilde{\boldsymbol{w}})} + \underbrace{\frac{1}{2}\lambda\|\tilde{w}\|_2^2}_{\text{regularization}}$$

What if data is not linearly separable or fits to a line

Example of nonlinear classification



Example of nonlinear regression



General nonlinear basis functions

We can use a nonlinear mapping

$$\boldsymbol{\phi}(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^D \rightarrow \boldsymbol{z} \in \mathbb{R}^M$$

where M is the dimensionality of the new feature/input $\boldsymbol{z}$ (or $\boldsymbol{\phi}(\boldsymbol{x})$). Note that M could be either greater than D or less than or the same.

With the new features, we can apply our learning techniques to minimize our errors on the transformed training data

- linear methods: prediction is based on $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$
- other methods: nearest neighbors, decision trees, etc

Regression with nonlinear basis

Residual sum squares

$$\sum_n [\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) - y_n]^2$$

where $\boldsymbol{w} \in \mathbb{R}^M$, the same dimensionality as the transformed features $\boldsymbol{\phi}(\boldsymbol{x})$.

The LMS solution can be formulated with the new design matrix

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}} \\ \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_N)^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{N \times M}, \quad \boldsymbol{w}^{\mathrm{LMS}} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{y}$$

Example with regression

Polynomial basis functions

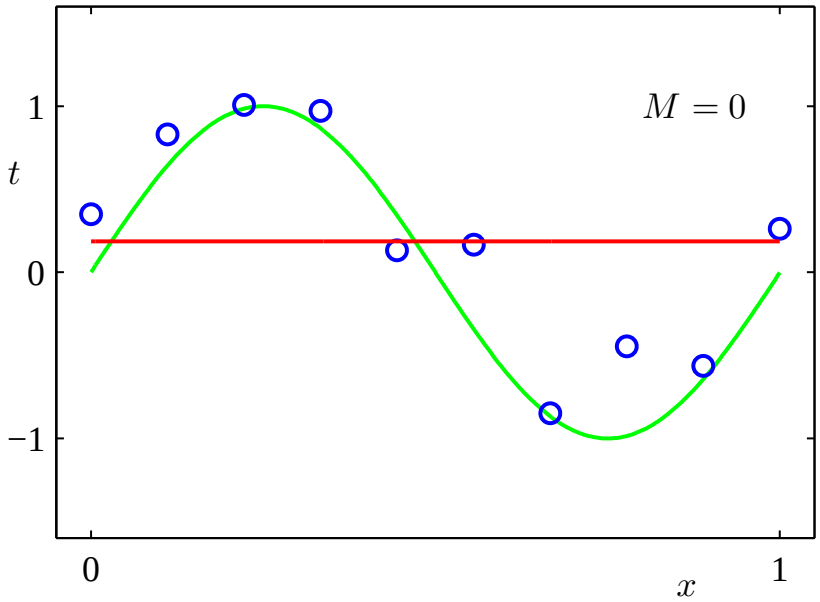
$$\boldsymbol{\phi}(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix} \Rightarrow f(x) = w_0 + \sum_{m=1}^{M} w_m x^m$$

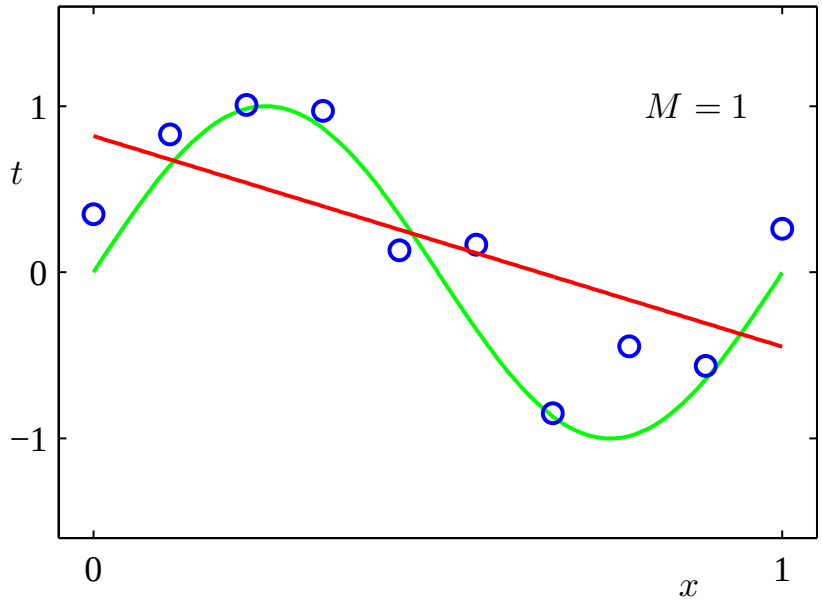
Example with regression

Polynomial basis functions

$$\boldsymbol{\phi}(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix} \Rightarrow f(x) = w_0 + \sum_{m=1}^{M} w_m x^m$$

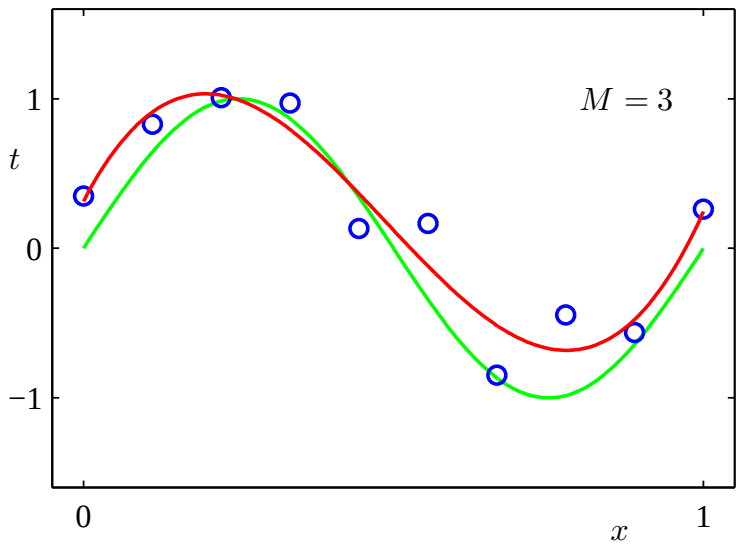
Fitting samples from a sine function: *underrfitting* as $f(x)$ is too simple





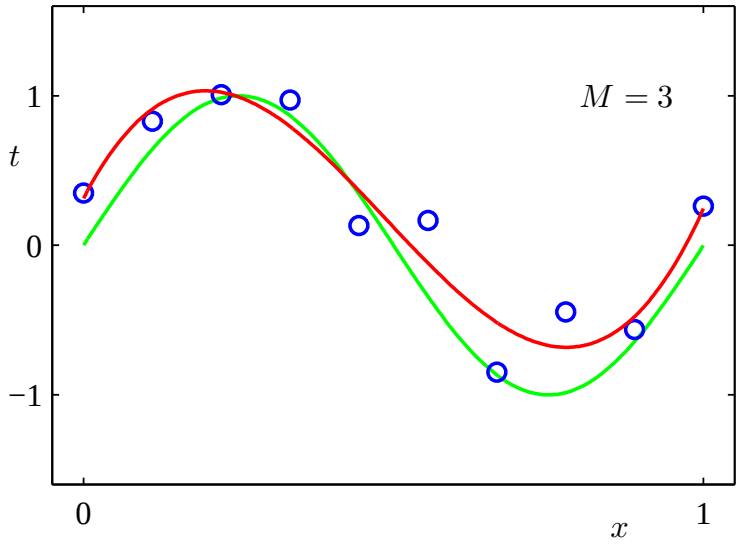
Adding high-order terms

M=3

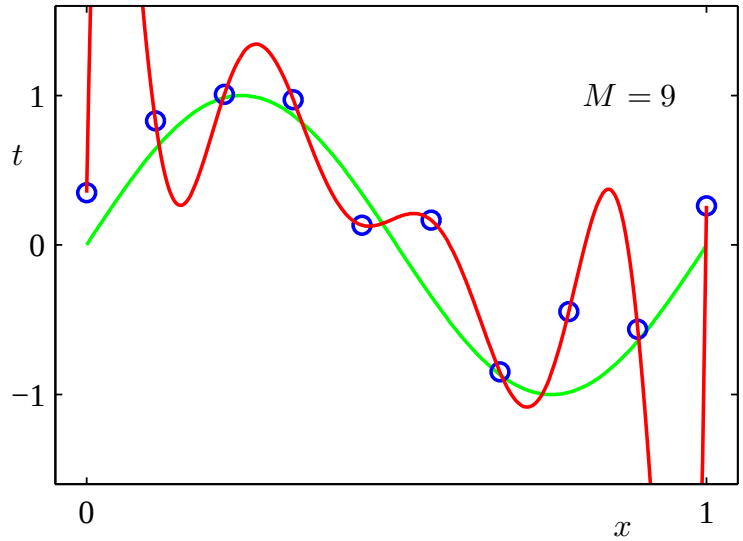


Adding high-order terms

M=3



M=9: *overfitting*



More complex features lead to better results on the training data, but potentially worse results on new data, e.g., test data!

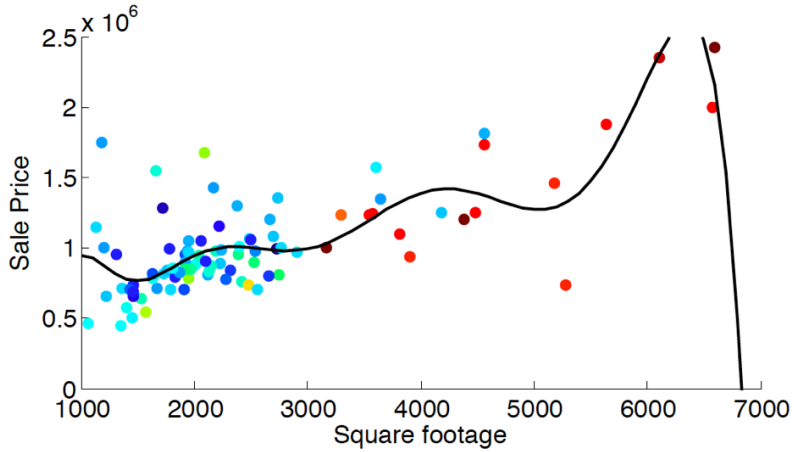
Overfitting

Parameters for higher-order polynomials are very large

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

Overfitting can be quite disastrous

Fitting the housing price data with $M = 3$



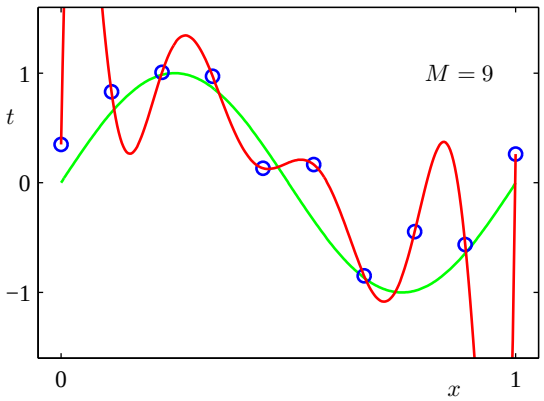
Note that the price would goes to zero (or negative) if you buy bigger ones! This is called poor generalization/overfitting.

How might we prevent overfitting?

Outline

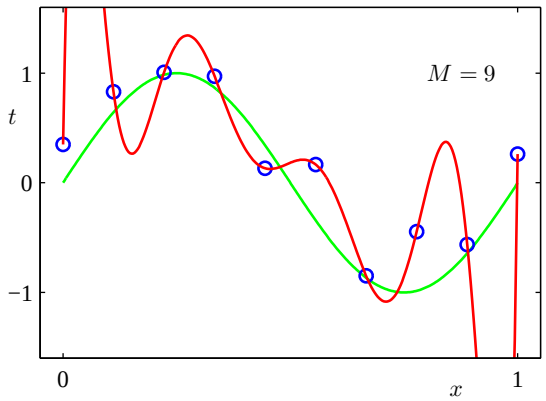
Use more training data to prevent over fitting

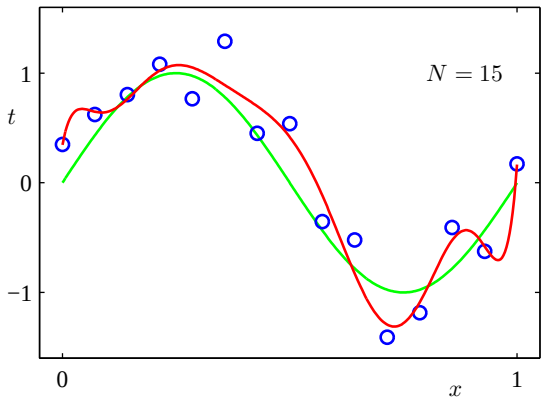
The more, the merrier

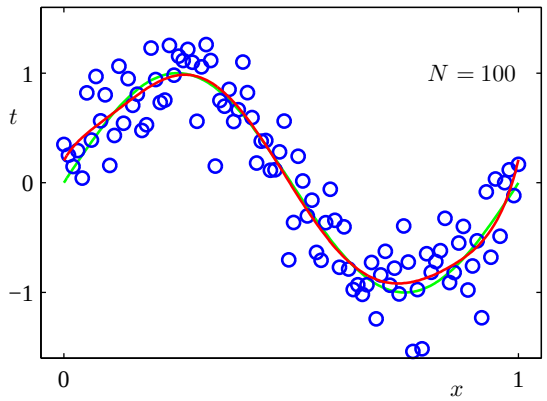


Use more training data to prevent over fitting

The more, the merrier

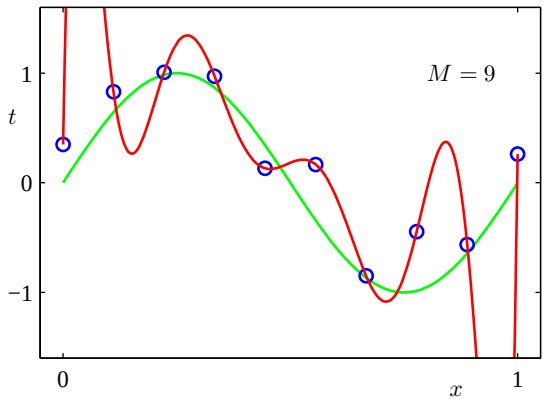


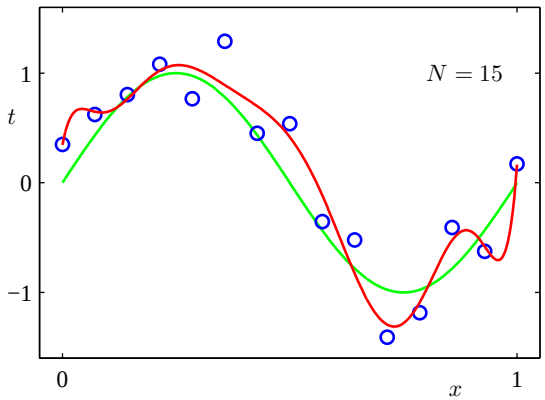


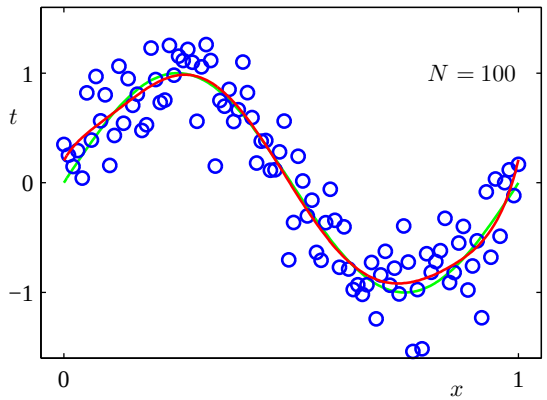


Use more training data to prevent over fitting

The more, the merrier







What if we do not have a lot of data?

Regularization methods

Intuition: For a linear model for regression

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$

we can try to identify 'simpler' models. But what does it mean for a model to *be simple*?

Regularization methods

Intuition: For a linear model for regression

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$

we can try to identify 'simpler' models. But what does it mean for a model to *be simple*?

Assumption We can place a *prior* on our weights, assuming that w_d is centered around zero.

With this reasoning, we will interpret $\boldsymbol{w}$ as a random variable and we will use the observed data $\mathcal{D}$ to update our prior belief on $\boldsymbol{w}$

Review: Probabilistic interpretation for LMS

Review: Probabilistic interpretation for LMS

- LMS model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $\eta \sim N(0, \sigma_0^2)$ is a Gaussian random variable
  - Thus, $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$

Review: Probabilistic interpretation for LMS

- LMS model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $\eta \sim N(0, \sigma_0^2)$ is a Gaussian random variable
  - Thus, $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$
- We assume that $\boldsymbol{w}$ is fixed (*Frequentist* interpretation)

Review: Probabilistic interpretation for LMS

- LMS model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $\eta \sim N(0, \sigma_0^2)$ is a Gaussian random variable
  - Thus, $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$
- We assume that $\boldsymbol{w}$ is fixed (*Frequentist* interpretation)
- We define $p(y|\boldsymbol{x}, \boldsymbol{w}, \sigma_0^2)$ as the sampling distribution given fixed values for the parameters $\boldsymbol{w}, \sigma_0^2$

Review: Probabilistic interpretation for LMS

- LMS model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $\eta \sim N(0, \sigma_0^2)$ is a Gaussian random variable
  - Thus, $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$
- We assume that $\boldsymbol{w}$ is fixed (*Frequentist* interpretation)
- We define $p(y|\boldsymbol{x}, \boldsymbol{w}, \sigma_0^2)$ as the sampling distribution given fixed values for the parameters $\boldsymbol{w}, \sigma_0^2$
- The likelihood function maps parameters to probabilities

$$L : \boldsymbol{w}, \sigma_0^2 \mapsto p(\boldsymbol{y}|\mathcal{D}, \boldsymbol{w}, \sigma_0^2) = \prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{w}, \sigma_0^2)$$

Review: Probabilistic interpretation for LMS

- LMS model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $\eta \sim N(0, \sigma_0^2)$ is a Gaussian random variable
  - Thus, $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$
- We assume that $\boldsymbol{w}$ is fixed (*Frequentist* interpretation)
- We define $p(y|\boldsymbol{x}, \boldsymbol{w}, \sigma_0^2)$ as the sampling distribution given fixed values for the parameters $\boldsymbol{w}, \sigma_0^2$
- The likelihood function maps parameters to probabilities

$$L : \boldsymbol{w}, \sigma_0^2 \mapsto p(\boldsymbol{y}|\mathcal{D}, \boldsymbol{w}, \sigma_0^2) = \prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{w}, \sigma_0^2)$$

- Maximizing likelihood with respect to $\boldsymbol{w}$ minimizes RSS and yields the LMS solution:

$$\boldsymbol{w}^{\text{LMS}} = \boldsymbol{w}^{\text{ML}} = \arg\max_{\boldsymbol{w}} L(\boldsymbol{w}, \sigma_0^2)$$

Probabilistic interpretation of Ridge Regression

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$ is a Gaussian random variable (as before)
  - $w_d \sim N(0, \sigma^2)$ are i.i.d. Gaussian random variables (unlike before)
  - Note that all w_d share the same variance σ^2

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$ is a Gaussian random variable (as before)
  - $w_d \sim N(0, \sigma^2)$ are i.i.d. Gaussian random variables (unlike before)
  - Note that all w_d share the same variance σ^2
- $\boldsymbol{w}$ is a random variable with a prior distribution (*Bayesian* interpretation)

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$ is a Gaussian random variable (as before)
  - $w_d \sim N(0, \sigma^2)$ are i.i.d. Gaussian random variables (unlike before)
  - Note that all w_d share the same variance σ^2
- $\boldsymbol{w}$ is a random variable with a prior distribution (*Bayesian* interpretation)
- To find $\boldsymbol{w}$ given data $\mathcal{D}$, we can compute posterior distribution of $\boldsymbol{w}$:

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

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$ is a Gaussian random variable (as before)
  - $w_d \sim N(0, \sigma^2)$ are i.i.d. Gaussian random variables (unlike before)
  - Note that all w_d share the same variance σ^2
- $\boldsymbol{w}$ is a random variable with a prior distribution (*Bayesian* interpretation)
- To find $\boldsymbol{w}$ given data $\mathcal{D}$, we can compute posterior distribution of $\boldsymbol{w}$:

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

- Maximum a posterior (MAP) estimate:

$$\boldsymbol{w}^{\text{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\mathcal{D}) = \arg\max_{\boldsymbol{w}} p(\mathcal{D}, \boldsymbol{w})$$

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$ is a Gaussian random variable (as before)
  - $w_d \sim N(0, \sigma^2)$ are i.i.d. Gaussian random variables (unlike before)
  - Note that all w_d share the same variance σ^2
- $\boldsymbol{w}$ is a random variable with a prior distribution (*Bayesian* interpretation)
- To find $\boldsymbol{w}$ given data $\mathcal{D}$, we can compute posterior distribution of $\boldsymbol{w}$:

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

- Maximum a posterior (MAP) estimate:

$$\boldsymbol{w}^{\text{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\mathcal{D}) = \arg\max_{\boldsymbol{w}} p(\mathcal{D}, \boldsymbol{w})$$

- What's the relationship between MAP and MLE?

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$ is a Gaussian random variable (as before)
  - $w_d \sim N(0, \sigma^2)$ are i.i.d. Gaussian random variables (unlike before)
  - Note that all w_d share the same variance σ^2
- $\boldsymbol{w}$ is a random variable with a prior distribution (*Bayesian* interpretation)
- To find $\boldsymbol{w}$ given data $\mathcal{D}$, we can compute posterior distribution of $\boldsymbol{w}$:

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

- Maximum a posterior (MAP) estimate:

$$\boldsymbol{w}^{\text{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\mathcal{D}) = \arg\max_{\boldsymbol{w}} p(\mathcal{D}, \boldsymbol{w})$$

- What's the relationship between MAP and MLE?
  - MAP reduces to MLE if we assume uniform prior for $p(\boldsymbol{w})$

Estimating $\boldsymbol{w}$

- Let $X_1, \ldots, X_N$ be IID with $y|\boldsymbol{w}, \boldsymbol{x} \sim N(\boldsymbol{w}^\top \boldsymbol{x}, \sigma_0^2)$
- Let w_d be IID with $w_d \sim N(0, \sigma^2)$

Joint likelihood of data and parameters (given σ_0, σ)

$$p(\mathcal{D}, \boldsymbol{w}) = p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w}) \quad = \prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) \prod_d p(w_d)$$

Estimating $\boldsymbol{w}$

- Let $X_1, \ldots, X_N$ be IID with $y|\boldsymbol{w}, \boldsymbol{x} \sim N(\boldsymbol{w}^\top \boldsymbol{x}, \sigma_0^2)$
- Let w_d be IID with $w_d \sim N(0, \sigma^2)$

Joint likelihood of data and parameters (given σ_0, σ)

$$p(\mathcal{D}, \boldsymbol{w}) = p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w}) \quad = \prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) \prod_d p(w_d)$$

Joint log likelihood Plugging in Gaussian PDF, we get:

$$\begin{aligned} \log p(\mathcal{D}, \boldsymbol{w}) &= \sum_n \log p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) + \sum_d \log p(w_d) \\ &= -\frac{\sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n)^2}{2\sigma_0^2} - \sum_d \frac{1}{2\sigma^2} w_d^2 + \text{const} \end{aligned}$$

Estimating $\boldsymbol{w}$

- Let $X_1, \ldots, X_N$ be IID with $y|\boldsymbol{w}, \boldsymbol{x} \sim N(\boldsymbol{w}^\top \boldsymbol{x}, \sigma_0^2)$
- Let w_d be IID with $w_d \sim N(0, \sigma^2)$

Joint likelihood of data and parameters (given σ_0, σ)

$$p(\mathcal{D}, \boldsymbol{w}) = p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w}) \quad = \prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) \prod_d p(w_d)$$

Joint log likelihood Plugging in Gaussian PDF, we get:

$$\begin{aligned} \log p(\mathcal{D}, \boldsymbol{w}) &= \sum_n \log p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) + \sum_d \log p(w_d) \\ &= -\frac{\sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n)^2}{2\sigma_0^2} - \sum_d \frac{1}{2\sigma^2} w_d^2 + \text{const} \end{aligned}$$

MAP estimate: $\boldsymbol{w}^{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} \log p(\mathcal{D}, \boldsymbol{w})$

- As with LMS, set gradient equal to zero and solve (for $\boldsymbol{w}$)

Maximum a posterior (MAP) estimate

Regularized linear regression: a new error to minimize

$$\mathcal{E}(\boldsymbol{w}) = \sum_n (\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n - y_n)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$ is used to denote σ_0^2/σ^2. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the model complexity.

Maximum a posterior (MAP) estimate

Regularized linear regression: a new error to minimize

$$\mathcal{E}(\boldsymbol{w}) = \sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$ is used to denote σ_0^2/σ^2. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the model complexity.

Intuitions

Maximum a posterior (MAP) estimate

Regularized linear regression: a new error to minimize

$$\mathcal{E}(\boldsymbol{w}) = \sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$ is used to denote σ_0^2/σ^2. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the model complexity.

Intuitions

- If $\lambda \to +\infty$, then $\sigma_0^2 \gg \sigma^2$. That is, the variance of noise is far greater than what our prior model can allow for $\boldsymbol{w}$. In this case, our prior model on $\boldsymbol{w}$ would be more accurate than what data can tell us. Thus, we are getting a *simple* model. Numerically,

$$\boldsymbol{w}^{\mathrm{MAP}} \to \mathbf{0}$$

Maximum a posterior (MAP) estimate

Regularized linear regression: a new error to minimize

$$\mathcal{E}(\boldsymbol{w}) = \sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$ is used to denote σ_0^2/σ^2. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the model complexity.

Intuitions

- If $\lambda \to +\infty$, then $\sigma_0^2 \gg \sigma^2$. That is, the variance of noise is far greater than what our prior model can allow for $\boldsymbol{w}$. In this case, our prior model on $\boldsymbol{w}$ would be more accurate than what data can tell us. Thus, we are getting a *simple* model. Numerically,

$$\boldsymbol{w}^{\mathrm{MAP}} \to \mathbf{0}$$

- If $\lambda \to 0$, then we trust our data more. Numerically,

$$\boldsymbol{w}^{\mathrm{MAP}} \to \boldsymbol{w}^{\mathrm{LMS}} = \arg\min \sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n)^2$$

Closed-form solution

For regularized linear regression: the solution changes very little (in form) from the LMS solution

$$\arg\min \sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n)^2 + \lambda \|\boldsymbol{w}\|_2^2 \Rightarrow \boldsymbol{w}^{\mathrm{MAP}} = \left(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{y}$$

and reduces to the LMS solution when $\lambda = 0$, as expected.

Closed-form solution

For regularized linear regression: the solution changes very little (in form) from the LMS solution

$$\arg\min \sum_n (\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n - y_n)^2 + \lambda \|\boldsymbol{w}\|_2^2 \Rightarrow \boldsymbol{w}^{\mathrm{MAP}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

and reduces to the LMS solution when $\lambda = 0$, as expected.

If we have to use numerical procedure, the gradients and the Hessian matrix would change nominally too,

$$\nabla \mathcal{E}(\boldsymbol{w}) = 2(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} + \lambda \boldsymbol{w}), \qquad \boldsymbol{H} = 2(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \lambda \boldsymbol{I})$$

As long as $\lambda \geq 0$, the optimization is convex.

Example: fitting data with polynomials

Our regression model

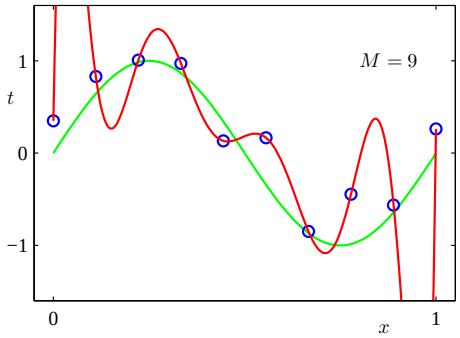
$$y = \sum_{m=1}^{M} w_m x^m$$

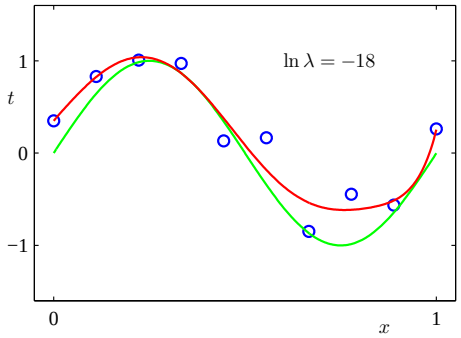
Regularization would discourage large parameter values as we saw with the LMS solution, thus potentially preventing overfitting.

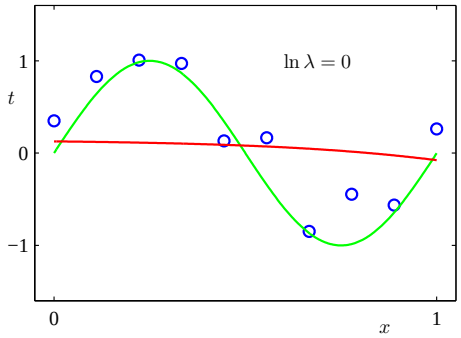
	$M=0$	$M=1$	$M=3$	$M=9$
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

Overfitting in terms of λ

Overfitting is reduced from complex model to simpler one with the help of increasing regularizers

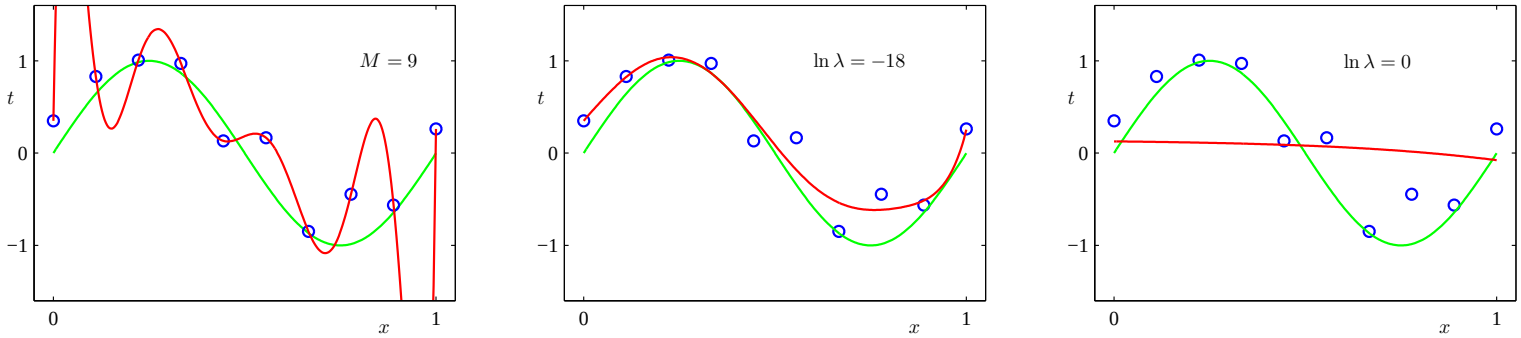




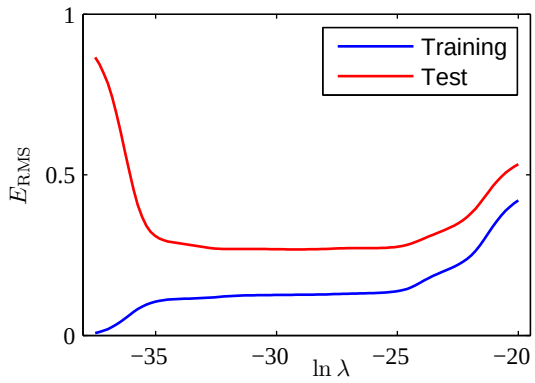


Overfitting in terms of λ

Overfitting is reduced from complex model to simpler one with the help of increasing regularizers



λ **vs. residual error** shows the difference of the model performance on training and testing dataset



The effect of λ

Large λ attenuates parameters towards 0

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

Regularized methods for classification

Adding regularizer to the cross-entropy functions used for binary and multinomial logistic regression

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) + (1-y_n)\log[1-\sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)]\} + \lambda\|\boldsymbol{w}\|_2^2$$

$$\mathcal{E}(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K) = -\sum_n \sum_k \log P(C_k|\boldsymbol{x}_n) + \lambda \sum_k \|\boldsymbol{w}_k\|_2^2$$

Regularized methods for classification

Adding regularizer to the cross-entropy functions used for binary and multinomial logistic regression

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) + (1-y_n)\log[1-\sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)]\} + \lambda\|\boldsymbol{w}\|_2^2$$

$$\mathcal{E}(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K) = -\sum_n \sum_k \log P(C_k|\boldsymbol{x}_n) + \lambda \sum_k \|\boldsymbol{w}_k\|_2^2$$

Numerical optimization

- Objective functions remain to be convex as long as $\lambda \geq 0$.
- Gradients and Hessians are changed marginally and can be easily derived.

How to choose the right amount of regularization?

Can we tune λ on the training dataset?

How to choose the right amount of regularization?

Can we tune λ on the training dataset?

No: as this will set λ to zero, i.e., without regularization, defeating our intention to use it to control model complexity and to gain better generalization.

λ is thus a hyperparmeter. To tune it,

- We can use a development/holdout dataset independent of training and testing dataset.
- We can do leave-one-out (LOO)

The procedure is similar to choose K in the nearest neighbor classifiers.

How to choose the right amount of regularization?

Can we tune λ on the training dataset?

No: as this will set λ to zero, i.e., without regularization, defeating our intention to use it to control model complexity and to gain better generalization.

λ is thus a hyperparmeter. To tune it,

- We can use a development/holdout dataset independent of training and testing dataset.
- We can do leave-one-out (LOO)

The procedure is similar to choose K in the nearest neighbor classifiers.

For different λ, we get $\boldsymbol{w}^{\text{MAP}}$ and evaluate the model on the development/holdout dataset (or, the samples being left in LOO).

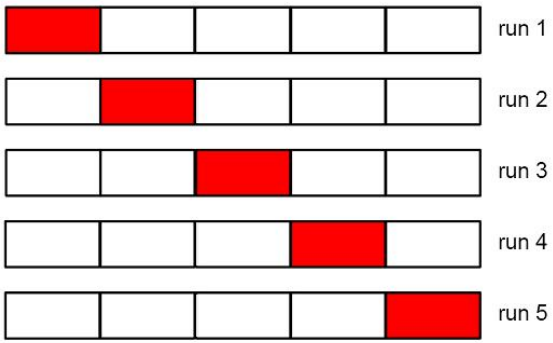
We then plot the curve λ versus prediction error (accuracy, classification error) and find the place that the performance on the holdout/LOO is the best.

Use cross-validation to choose λ

Procedure

- Randomly partition training data into K *disjoint* parts
  Normally, K is chosen to be 10, 5, etc.
- For each possible value of λ
  1. Use one part as holdout; use other $(K-1)$ parts as training
  2. Evaluate the model on the holdout
  3. Do this K times, and average the performance on the holdouts
- Choose the λ with the best performance

When $K = N$ (the number of training examples), this becomes LOO.



Outline

Basic and important machine learning concepts

Supervised learning

We aim to build a function $h(\boldsymbol{x})$ to predict the true value y associated with $\boldsymbol{x}$. If we make a mistake, we incur a *loss*

$$\ell(h(\boldsymbol{x}), y)$$

Basic and important machine learning concepts

Supervised learning

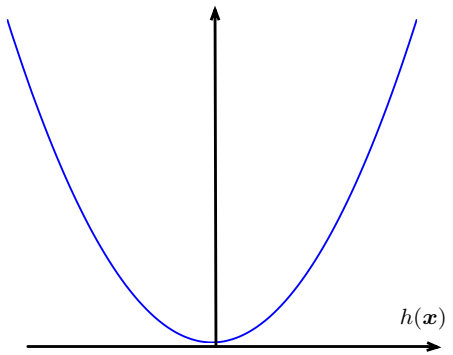
We aim to build a function $h(\boldsymbol{x})$ to predict the true value y associated with $\boldsymbol{x}$. If we make a mistake, we incur a *loss*

$$\ell(h(\boldsymbol{x}), y)$$

Example: quadratic loss function for regression when y is continuous

$$\ell(h(\boldsymbol{x}), y) = [h(\boldsymbol{x}) - y]^2$$

Ex: when $y = 0$

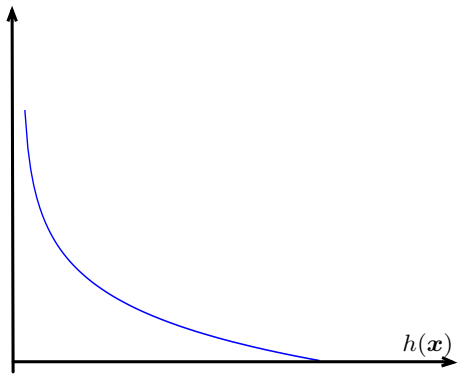


Other types of loss functions

For classification: cross-entropy loss (also called $logistic$ loss)

$$\ell(h(\boldsymbol{x}), y) = -y \log h(\boldsymbol{x}) - (1-y) \log[1-h(\boldsymbol{x})]$$

Ex: when $y = 1$



Measure how good our predictor is

Risk: assume we know the true distribution of data $p(\boldsymbol{x}, y)$, the *risk* is

$$R[h(\boldsymbol{x})] = \int_{\boldsymbol{x},y} \ell(h(\boldsymbol{x}), y) p(\boldsymbol{x}, y) d\boldsymbol{x} d\, y$$

Measure how good our predictor is

Risk: assume we know the true distribution of data $p(\boldsymbol{x}, y)$, the *risk* is

$$R[h(\boldsymbol{x})] = \int_{\boldsymbol{x},y} \ell(h(\boldsymbol{x}), y) p(\boldsymbol{x}, y) d\boldsymbol{x} d\, y$$

However, we cannot compute $R[h(\boldsymbol{x})]$, so we use *empirical risk*, given a training dataset $\mathcal{D}$

$$R^{\text{EMP}}[h(\boldsymbol{x})] = \frac{1}{N} \sum_n \ell(h(\boldsymbol{x}_n), y_n)$$

Measure how good our predictor is

Risk: assume we know the true distribution of data $p(\boldsymbol{x}, y)$, the *risk* is

$$R[h(\boldsymbol{x})] = \int_{\boldsymbol{x},y} \ell(h(\boldsymbol{x}), y) p(\boldsymbol{x}, y) d\boldsymbol{x} d\, y$$

However, we cannot compute $R[h(\boldsymbol{x})]$, so we use *empirical risk*, given a training dataset $\mathcal{D}$

$$R^{\text{EMP}}[h(\boldsymbol{x})] = \frac{1}{N} \sum_n \ell(h(\boldsymbol{x}_n), y_n)$$

Intuitively, as $N \to +\infty$,

$$R^{\text{EMP}}[h(\boldsymbol{x})] \to R[h(\boldsymbol{x})]$$

How this relates to what we have learned?

So far, we have been doing empirical risk minimization (ERM)

- For linear regression, $h(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$, and we use squared loss
- For logistic regression, $h(\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$, and we use cross-entropy loss

How this relates to what we have learned?

So far, we have been doing empirical risk minimization (ERM)

- For linear regression, $h(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$, and we use squared loss
- For logistic regression, $h(\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$, and we use cross-entropy loss

ERM might be problematic

- If $h(\boldsymbol{x})$ is complicated enough,

$$R^{\mathrm{EMP}}[h(\boldsymbol{x})] \to 0$$

- But then $h(\boldsymbol{x})$ is unlikely to do well in predicting things out of the training dataset $\mathcal{D}$
- This is called *poor generalization* or *overfitting*. We have just discussed approaches to address this issue.
- Let's try to understand why regularization might work from the context of the bias-variance tradeoff, focusing on regression / squared loss

Bias/variance tradeoff (Looking ahead)

Error decomposes into 3 terms

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \text{VARIANCE} + \text{BIAS}^2 + \text{NOISE}$$

We will prove this result, and interpret what it means...