CSCI567 Machine Learning (Spring 2021)

Sirisha Rambhatla

University of Southern California

Jan 27, 2021

Outline

Outline

Logistics

- The schedule of lectures is available at
  https://sirisharambhatla.com/CSCI567/index.html

Outline

Regression

Predicting a continuous outcome variable using past observations

- temperature, amount of rainfall, house price, etc.

Key difference from classification

- continuous vs discrete
- measure *prediction errors* differently.
- lead to quite different learning algorithms.

Linear Regression: regression with linear models: $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$

Least square solution

$$
\begin{aligned}
\boldsymbol{w}^* &= \underset{\boldsymbol{w}}{\operatorname{argmin}} \operatorname{RSS}(\boldsymbol{w}) \\
&= \underset{\boldsymbol{w}}{\operatorname{argmin}} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 \\
&= \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}
\end{aligned}
$$

$$
\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} \\ \boldsymbol{x}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{x}_{\mathsf{N}}^{\mathrm{T}} \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{\mathsf{N}} \end{pmatrix}
$$

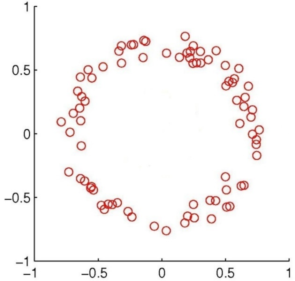
Two approaches to find the minimum:

- find **stationary points** by setting gradient $= \mathbf{0}$
- "**complete the square**"

Outline

What if linear model is not a good fit?

Example: a straight line is a bad fit for the following data



Solution: nonlinearly transformed features

1. Use a nonlinear mapping

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

to transform the data to a more complicated feature space

Solution: nonlinearly transformed features

1. Use a nonlinear mapping

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

to transform the data to a more complicated feature space

2. Then apply linear regression (hope: linear model is a better fit for the new feature space).

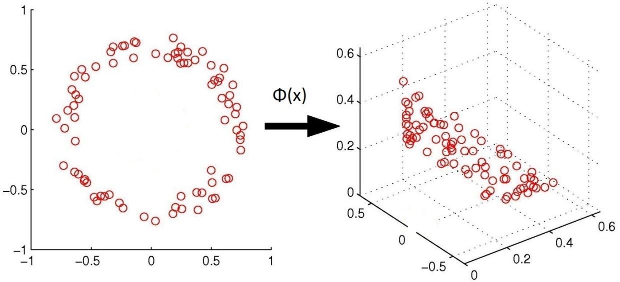
Solution: nonlinearly transformed features

1. Use a nonlinear mapping

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

to transform the data to a more complicated feature space

2. Then apply linear regression (hope: linear model is a better fit for the new feature space).



Regression with nonlinear basis

Model: $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$ where $\boldsymbol{w} \in \mathbb{R}^M$

Regression with nonlinear basis

Model: $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})$ where $\boldsymbol{w} \in \mathbb{R}^M$

Objective:

$$\mathrm{RSS}(\boldsymbol{w}) = \sum_n \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) - y_n \right)^2$$

Regression with nonlinear basis

Model: $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$ where $\boldsymbol{w} \in \mathbb{R}^{M}$

Objective:

$$\mathrm{RSS}(\boldsymbol{w}) = \sum_{n} \left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) - y_n\right)^2$$

Similar least square solution:

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

Example

Polynomial basis functions for $\mathsf{D} = 1$

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

Example

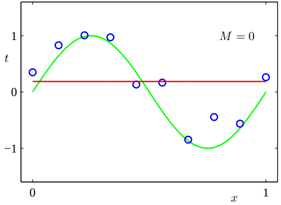
Polynomial basis functions for $\mathrm{D} = 1$

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

Learning a linear model in the new space
= learning an *M-degree polynomial model* in the original space

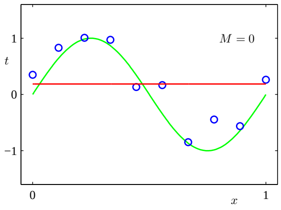
Example

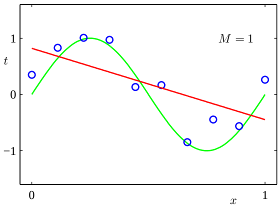
Fitting a noisy sine function with a polynomial ($M = 0, 1,$ or 3):



Example

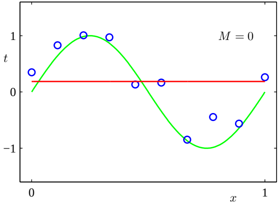
Fitting a noisy sine function with a polynomial ($M = 0, 1,$ or 3):

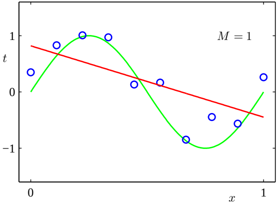


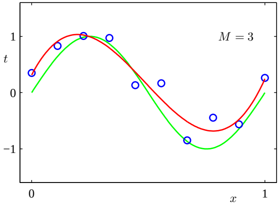


Example

Fitting a noisy sine function with a polynomial ($M = 0, 1,$ or 3):







Why nonlinear?

Can I use a fancy **linear feature map**?

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} x_1 - x_2 \\ 3x_4 - x_3 \\ 2x_1 + x_4 + x_5 \\ \vdots \end{bmatrix} = \boldsymbol{A}\boldsymbol{x} \quad \text{for some } \boldsymbol{A} \in \mathbb{R}^{\mathsf{M}\times\mathsf{D}}$$

Why nonlinear?

Can I use a fancy **linear feature map**?

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} x_1 - x_2 \\ 3x_4 - x_3 \\ 2x_1 + x_4 + x_5 \\ \vdots \end{bmatrix} = \boldsymbol{A}\boldsymbol{x} \quad \text{for some } \boldsymbol{A} \in \mathbb{R}^{\mathsf{M}\times\mathsf{D}}$$

No, it basically *does nothing* since

$$\min_{\boldsymbol{w}\in\mathbb{R}^{\mathsf{M}}} \sum_n \left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}_n - y_n\right)^2 = \min_{\boldsymbol{w}'\in\mathrm{Im}(\boldsymbol{A}^{\mathrm{T}})\subset\mathbb{R}^{\mathsf{D}}} \sum_n \left(\boldsymbol{w}'^{\mathrm{T}}\boldsymbol{x}_n - y_n\right)^2$$

Why nonlinear?

Can I use a fancy **linear feature map**?

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} x_1 - x_2 \\ 3x_4 - x_3 \\ 2x_1 + x_4 + x_5 \\ \vdots \end{bmatrix} = \boldsymbol{A}\boldsymbol{x} \quad \text{for some } \boldsymbol{A} \in \mathbb{R}^{\mathsf{M}\times\mathsf{D}}$$

No, it basically *does nothing* since

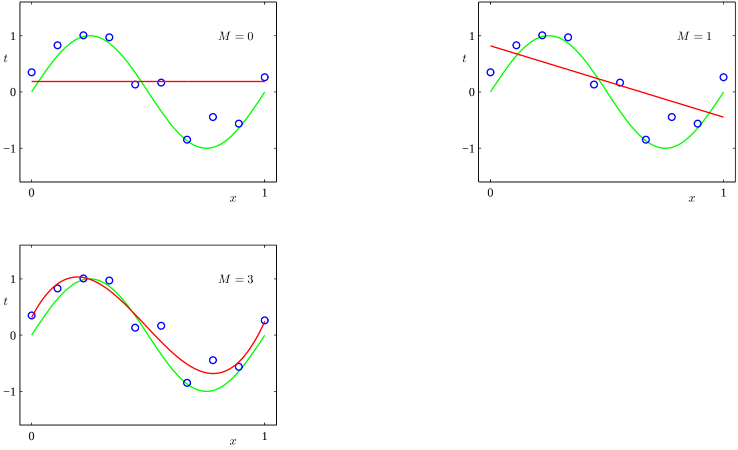
$$\min_{\boldsymbol{w}\in\mathbb{R}^{\mathsf{M}}} \sum_n \left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}_n - y_n\right)^2 = \min_{\boldsymbol{w'}\in\mathrm{Im}(\boldsymbol{A}^{\mathrm{T}})\subset\mathbb{R}^{\mathsf{D}}} \sum_n \left(\boldsymbol{w'}^{\mathrm{T}}\boldsymbol{x}_n - y_n\right)^2$$

We will see more nonlinear mappings soon.

Outline

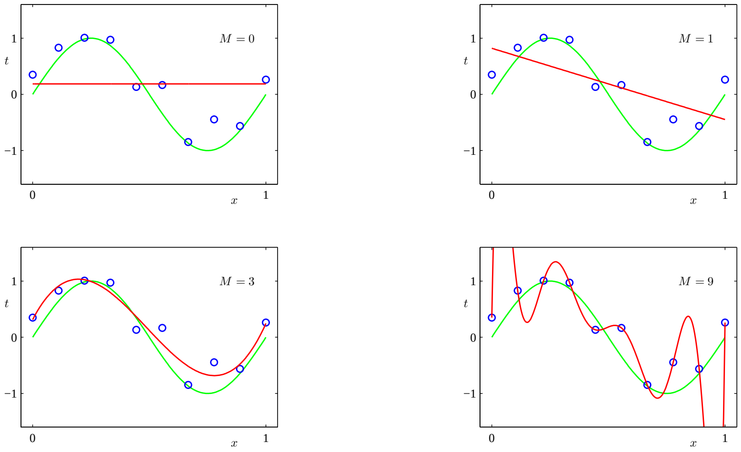
Should we use a very complicated mapping?

Ex: fitting a noisy sine function with a polynomial:



Should we use a very complicated mapping?

Ex: fitting a noisy sine function with a polynomial:



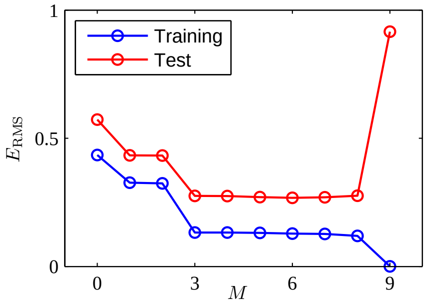
Underfitting and Overfitting

$M \leq 2$ is *underfitting* the data

- large training error
- large test error

$M \geq 9$ is *overfitting* the data

- small training error
- **large test error**



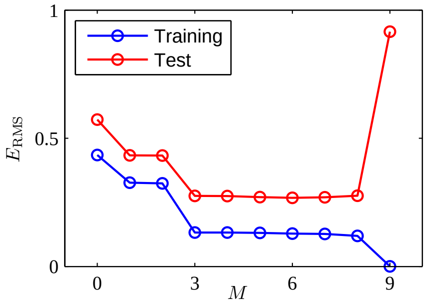
Underfitting and Overfitting

$M \leq 2$ is *underfitting* the data

- large training error
- large test error

$M \geq 9$ is *overfitting* the data

- small training error
- **large test error**



More complicated models $\Rightarrow$ larger gap between training and test error

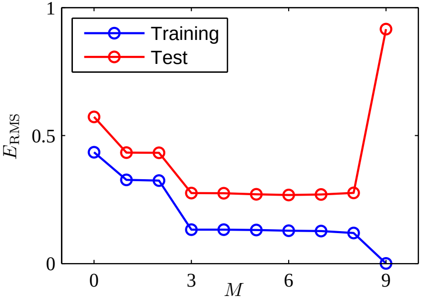
Underfitting and Overfitting

$M \leq 2$ is *underfitting* the data

- large training error
- large test error

$M \geq 9$ is *overfitting* the data

- small training error
- **large test error**

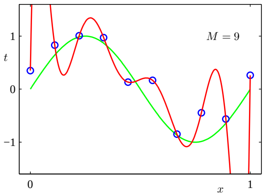


More complicated models $\Rightarrow$ larger gap between training and test error

How to prevent overfitting?

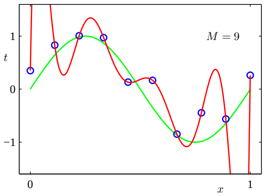
Method 1: use more training data

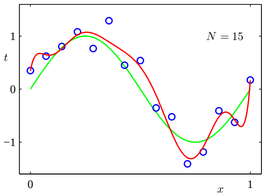
The more, the merrier



Method 1: use more training data

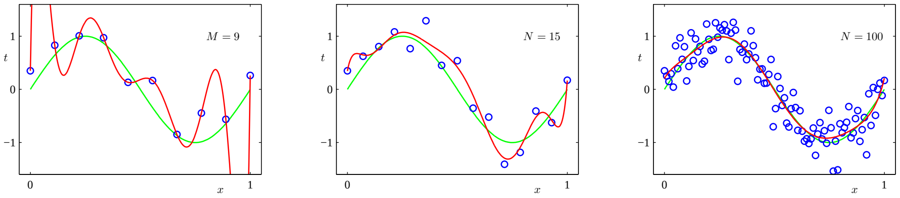
The more, the merrier





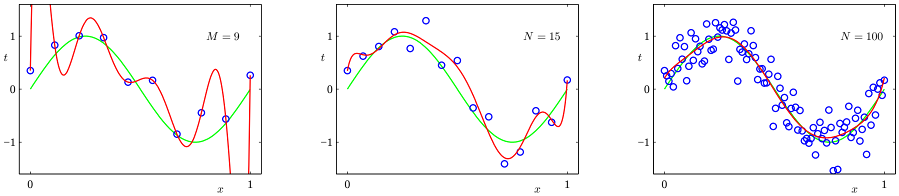
Method 1: use more training data

The more, the merrier



Method 1: use more training data

The more, the merrier



More data ⇒ smaller gap between training and test error

Method 2: control the model complexity

For polynomial basis, the **degree** M clearly controls the complexity

- use cross-validation to pick hyper-parameter M

Method 2: control the model complexity

For polynomial basis, the **degree** M clearly controls the complexity

- use cross-validation to pick hyper-parameter M

When M or in general Φ is fixed, are there still other ways to control complexity?

Magnitude of weights

Least square solution for the polynomial example:

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
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

Magnitude of weights

Least square solution for the polynomial example:

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
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

Intuitively, **large weights $\Rightarrow$ more complex model**

How to make $\boldsymbol{w}$ small?

Regularized linear regression: new objective

$$\mathcal{E}(\boldsymbol{w}) = \mathrm{RSS}(\boldsymbol{w}) + \lambda R(\boldsymbol{w})$$

Goal: find $\boldsymbol{w}^* = \mathrm{argmin}_w \, \mathcal{E}(\boldsymbol{w})$

How to make $\boldsymbol{w}$ small?

Regularized linear regression: new objective

$$\mathcal{E}(\boldsymbol{w}) = \text{RSS}(\boldsymbol{w}) + \lambda R(\boldsymbol{w})$$

Goal: find $\boldsymbol{w}^* = \text{argmin}_w \, \mathcal{E}(\boldsymbol{w})$

- $R : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}^+$ is the *regularizer*
  - measure how complex the model $\boldsymbol{w}$ is
  - common choices: $\|\boldsymbol{w}\|_2^2$, $\|\boldsymbol{w}\|_1$, etc.

How to make $\boldsymbol{w}$ small?

Regularized linear regression: new objective

$$\mathcal{E}(\boldsymbol{w}) = \mathrm{RSS}(\boldsymbol{w}) + \lambda R(\boldsymbol{w})$$

Goal: find $\boldsymbol{w}^* = \mathrm{argmin}_w \, \mathcal{E}(\boldsymbol{w})$

- $R : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}^+$ is the *regularizer*
  - measure how complex the model $\boldsymbol{w}$ is
  - common choices: $\|\boldsymbol{w}\|_2^2$, $\|\boldsymbol{w}\|_1$, etc.
- $\lambda > 0$ is the *regularization coefficient*
  - $\lambda = 0$, no regularization
  - $\lambda \to +\infty$, $\boldsymbol{w} \to \mathrm{argmin}_w \, R(\boldsymbol{w})$
  - i.e. control **trade-off** between training error and complexity

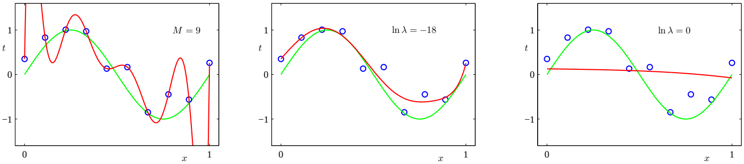
The effect of λ

when we increase regularization coefficient λ

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
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

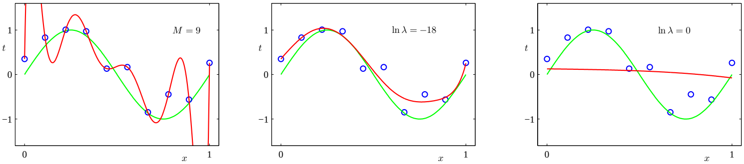
The trade-off

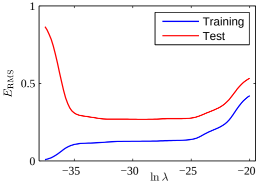
when we increase regularization coefficient λ



The trade-off

when we increase regularization coefficient λ





How to solve the new objective?

Simple for $R(\boldsymbol{w}) = \|\boldsymbol{w}\|_2^2$:

$$\mathcal{E}(\boldsymbol{w}) = \mathrm{RSS}(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_2^2 = \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$$

How to solve the new objective?

Simple for $R(\boldsymbol{w}) = \|\boldsymbol{w}\|_2^2$:

$$\mathcal{E}(\boldsymbol{w}) = \mathrm{RSS}(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_2^2 = \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$$

$$\nabla\mathcal{E}(\boldsymbol{w}) = 2(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}) + 2\lambda\boldsymbol{w} = 0$$

How to solve the new objective?

Simple for $R(\boldsymbol{w}) = \|\boldsymbol{w}\|_2^2$:

$$\mathcal{E}(\boldsymbol{w}) = \mathrm{RSS}(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_2^2 = \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$$

$$\nabla\mathcal{E}(\boldsymbol{w}) = 2(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}) + 2\lambda\boldsymbol{w} = 0$$
$$\Rightarrow \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I}\right)\boldsymbol{w} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$$

How to solve the new objective?

Simple for $R(\boldsymbol{w}) = \|\boldsymbol{w}\|_2^2$:

$$\mathcal{E}(\boldsymbol{w}) = \mathrm{RSS}(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_2^2 = \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$$

$$\begin{aligned}
\nabla\mathcal{E}(\boldsymbol{w}) &= 2(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}) + 2\lambda\boldsymbol{w} = 0 \\
&\Rightarrow \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I}\right)\boldsymbol{w} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} \\
&\Rightarrow \boldsymbol{w}^* = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}
\end{aligned}$$

How to solve the new objective?

Simple for $R(\boldsymbol{w}) = \|\boldsymbol{w}\|_2^2$:

$$\mathcal{E}(\boldsymbol{w}) = \mathrm{RSS}(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_2^2 = \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$$

$$\nabla\mathcal{E}(\boldsymbol{w}) = 2(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}) + 2\lambda\boldsymbol{w} = 0$$
$$\Rightarrow \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I}\right)\boldsymbol{w} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$$
$$\Rightarrow \boldsymbol{w}^* = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$$

Note the same form as in the fix when $\boldsymbol{X}^T\boldsymbol{X}$ is not invertible!

How to solve the new objective?

Simple for $R(\boldsymbol{w}) = \|\boldsymbol{w}\|_2^2$:

$$\mathcal{E}(\boldsymbol{w}) = \mathrm{RSS}(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_2^2 = \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$$

$$\begin{aligned}
\nabla\mathcal{E}(\boldsymbol{w}) &= 2(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}) + 2\lambda\boldsymbol{w} = 0 \\
&\Rightarrow \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I}\right)\boldsymbol{w} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} \\
&\Rightarrow \boldsymbol{w}^* = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}
\end{aligned}$$

Note the same form as in the fix when $\boldsymbol{X}^T\boldsymbol{X}$ is not invertible!

For other regularizers, as long as it's **convex**, standard optimization algorithms can be applied.

Equivalent form

Regularization is also sometimes formulated as

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \operatorname{RSS}(w) \quad \textbf{subject to } R(\boldsymbol{w}) \leq \beta$$

where β is some hyper-parameter.

Equivalent form

Regularization is also sometimes formulated as

$$\underset{\boldsymbol{w}}{\operatorname{argmin}}\, \mathrm{RSS}(w) \quad \textbf{subject to } R(\boldsymbol{w}) \leq \beta$$

where β is some hyper-parameter.

Finding the solution becomes a *constrained optimization problem*.

Equivalent form

Regularization is also sometimes formulated as

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \operatorname{RSS}(w) \quad \textbf{subject to } R(\boldsymbol{w}) \leq \beta$$

where β is some hyper-parameter.

Finding the solution becomes a *constrained optimization problem*.

Choosing either λ or β can be done by cross-validation.

Summary

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

Summary

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

Important to understand the derivation than remembering the formula

Summary

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

Important to understand the derivation than remembering the formula

Overfitting: small training error but large test error

Summary

$$w^* = \left(\Phi^{\mathrm{T}}\Phi + \lambda I\right)^{-1} \Phi^{\mathrm{T}} y$$

Important to understand the derivation than remembering the formula

Overfitting: small training error but large test error

Preventing Overfitting: more data + regularization

Recall the question

Typical steps of developing a machine learning system:

- Collect data, split into training, development, and test sets.
- *Train a model with a machine learning algorithm.* Most often we apply cross-validation to tune hyper-parameters.
- Evaluate using the test data and report performance.
- Use the model to predict future/make decisions.

How to do the *red part* exactly?

General idea to derive ML algorithms

1. Pick a set of **models** $\mathcal{F}$
   - e.g. $\mathcal{F} = \{f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} \mid \boldsymbol{w} \in \mathbb{R}^{\mathsf{D}}\}$
   - e.g. $\mathcal{F} = \{f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Phi}(\boldsymbol{x}) \mid \boldsymbol{w} \in \mathbb{R}^{\mathsf{M}}\}$

General idea to derive ML algorithms

1. Pick a set of **models** $\mathcal{F}$
   - e.g. $\mathcal{F} = \{f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} \mid \boldsymbol{w} \in \mathbb{R}^{\mathsf{D}}\}$
   - e.g. $\mathcal{F} = \{f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Phi}(\boldsymbol{x}) \mid \boldsymbol{w} \in \mathbb{R}^{\mathsf{M}}\}$

2. Define **error/loss** $L(y', y)$

General idea to derive ML algorithms

1. Pick a set of **models** $\mathcal{F}$
   - e.g. $\mathcal{F} = \{f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} \mid \boldsymbol{w} \in \mathbb{R}^{\mathsf{D}}\}$
   - e.g. $\mathcal{F} = \{f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Phi}(\boldsymbol{x}) \mid \boldsymbol{w} \in \mathbb{R}^{\mathsf{M}}\}$

2. Define **error/loss** $L(y', y)$

3. Find **empirical risk minimizer (ERM)**:

$$\boldsymbol{f}^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \sum_{n=1}^{N} L(f(x_n), y_n)$$

General idea to derive ML algorithms

1. Pick a set of **models** $\mathcal{F}$
   - e.g. $\mathcal{F} = \{f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} \mid \boldsymbol{w} \in \mathbb{R}^{\mathsf{D}}\}$
   - e.g. $\mathcal{F} = \{f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Phi}(\boldsymbol{x}) \mid \boldsymbol{w} \in \mathbb{R}^{\mathsf{M}}\}$
2. Define **error/loss** $L(y', y)$
3. Find **empirical risk minimizer (ERM)**:

$$\boldsymbol{f}^* = \operatorname*{argmin}_{f\in\mathcal{F}} \sum_{n=1}^{N} L(f(x_n), y_n)$$

or **regularized empirical risk minimizer**:

$$\boldsymbol{f}^* = \operatorname*{argmin}_{f\in\mathcal{F}} \sum_{n=1}^{N} L(f(x_n), y_n) + \lambda R(f)$$

General idea to derive ML algorithms

1. Pick a set of **models** $\mathcal{F}$
   - e.g. $\mathcal{F} = \{f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} \mid \boldsymbol{w} \in \mathbb{R}^{\mathsf{D}}\}$
   - e.g. $\mathcal{F} = \{f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Phi}(\boldsymbol{x}) \mid \boldsymbol{w} \in \mathbb{R}^{\mathsf{M}}\}$

2. Define **error/loss** $L(y', y)$

3. Find **empirical risk minimizer (ERM)**:

$$\boldsymbol{f}^* = \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{n=1}^{N} L(f(x_n), y_n)$$

or **regularized empirical risk minimizer**:

$$\boldsymbol{f}^* = \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{n=1}^{N} L(f(x_n), y_n) + \lambda R(f)$$

ML becomes optimization