Fundamentals of Machine Learning

MODEL FITTING & PARAMETER ESTIMATION

BIAS-VARIANCE; VALIDATION

Amit K Roy-Chowdhury

Acknowledgments: Adapted from slides at https://probml.github.io/pml-book/teaching1.html by Prof. Saw Shier Nee
Adapted from slides of Prof. Fowler @ Binghamton University;
Modified by Prof. Roy-Chowdhury @ UC Riverside

Model Fitting / Training

$$\hat{\theta} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \boxed{\mathcal{L}(\theta)}$$

Loss function / objective function

Maximum likelihood estimation

$$\hat{\boldsymbol{\theta}}_{\text{mle}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \log p(\boldsymbol{y}_n | \boldsymbol{x}_n, \boldsymbol{\theta})$$

Estimated Parameter

Probability

Model

Since most optimization algorithms are designed to minimize cost functions, we redefine the objective function to be the (conditional) negative log likelihood or NLL and we minimize NLL

$$\text{NLL}(\boldsymbol{\theta}) \triangleq -\log p(\mathcal{D}|\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(\boldsymbol{y}_n | \boldsymbol{x}_n, \boldsymbol{\theta})$$

Notation and Form for MAP

Notation: $\hat{\theta}_{MAP}$ maximizes the posterior PDF

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \; p(\theta \mid \mathbf{x})$$

Equivalent Form (via Bayes' Rule): $\boxed{\hat{\theta}_{MAP} = \arg\max_{\theta} \, [p(\mathbf{x} \mid \theta) \; p(\theta)]}$

Proof: Use $p(\theta \mid \mathbf{x}) = \dfrac{p(\mathbf{x} \mid \theta) p(\theta)}{p(\mathbf{x})}$

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \left[\frac{p(\mathbf{x} \mid \theta) p(\theta)}{\boxed{p(\mathbf{x})}} \right] = \arg\max_{\theta} \, [p(\mathbf{x} \mid \theta) \; p(\theta)]$$

Least Squares Approach

All the previous methods we've studied… required a <u>probabilistic</u> model for the data: <u>Needed the PDF</u> $p(\mathbf{x};\boldsymbol{\theta})$

For a Signal + Noise problem we needed:

Signal Model & Noise Model

Least-Squares is <u>not</u> statistically based!!!

⇒ Do <u>NOT need</u> a PDF Model

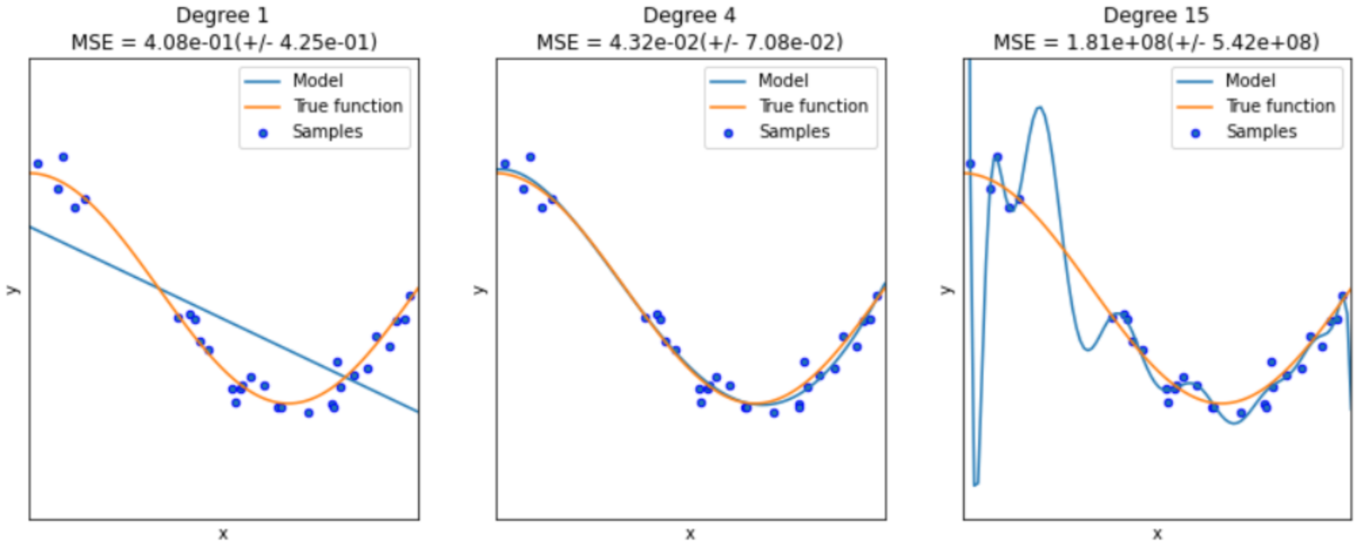
Empirical Risk Minimization

We can generalize MLE by replacing the (conditional) log loss term, with any other loss function.

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \ell(\boldsymbol{y}_n, \boldsymbol{\theta}; \boldsymbol{x}_n)$$

This is known as empirical risk minimization or ERM, since it is the expected loss where the expectation is taken wrt the empirical distribution.

Bias-Variance Tradeoffs

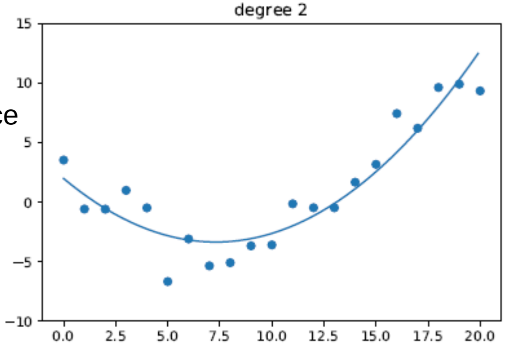


- Enough parameters to perfectly fit the training data.
- Most of the time, empirical distribution ≠ true distribution
- Model unable to predict novel future data Overfitting

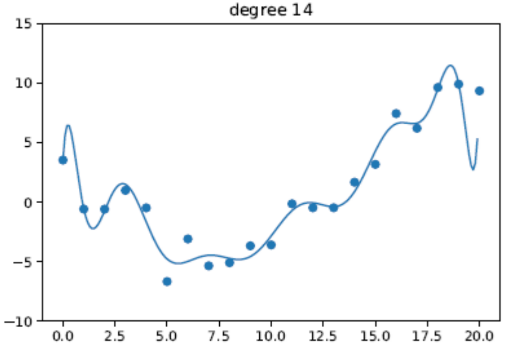
Bias-Variance Tradeoffs

Bias-Variance Tradeoffs

High bias,
low variance

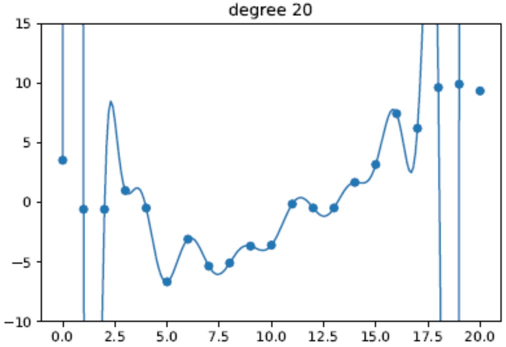


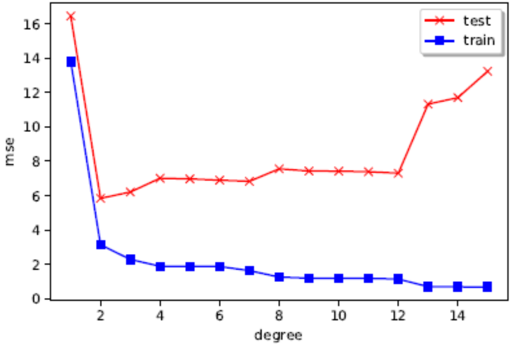
(a)



(b)

Low bias,
high
variance





Regularization

- Add penalty term to loss function

$$\mathcal{L}(\boldsymbol{\theta}; \lambda) = \left[\frac{1}{N}\sum_{n=1}^{N} \ell(\boldsymbol{y}_n, \boldsymbol{\theta}; \boldsymbol{x}_n)\right] + \lambda C(\boldsymbol{\theta})$$

where $\lambda \geq 0$ is the **regularization parameter**, and $C(\boldsymbol{\theta})$ is some form of **complexity penalty**.
A common complexity penalty is to use $C(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta})$, where $p(\boldsymbol{\theta})$ is the **prior** for $\boldsymbol{\theta}$. If ℓ is the log loss, the regularized objective becomes

$$\mathcal{L}(\boldsymbol{\theta}; \lambda) = -\frac{1}{N}\sum_{n=1}^{N} \log p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{\theta}) - \lambda \log p(\boldsymbol{\theta}) \tag{4.90}$$

Validation

run 1

1. Fit the model on D_{train} (for each setting of λ) with loss function

$$R_\lambda(\boldsymbol{\theta}, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \ell(\boldsymbol{y}, f(\boldsymbol{x};\boldsymbol{\theta})) + \lambda C(\boldsymbol{\theta})$$

1. For each λ, we compute parameter estimate

$$\hat{\boldsymbol{\theta}}_\lambda(\mathcal{D}_{\text{train}}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, R_\lambda(\boldsymbol{\theta}, \mathcal{D}_{\text{train}})$$

1. Then evaluate its performance on D_{valid} using validation data.

$$R_\lambda^{\text{val}} \triangleq R_0(\hat{\boldsymbol{\theta}}_\lambda(\mathcal{D}_{\text{train}}), \mathcal{D}_{\text{valid}})$$

1. We then pick the value of λ that results in the best validation performance.

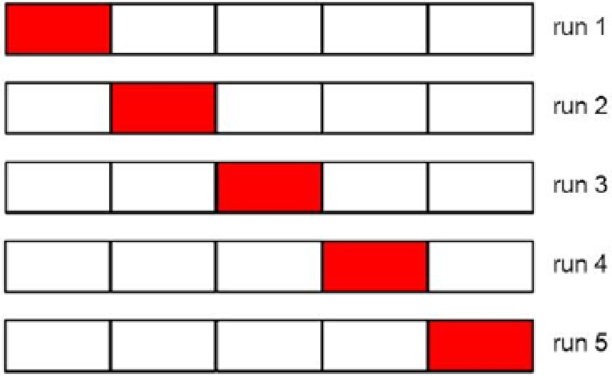
$$\lambda^* = \underset{\lambda\in\mathcal{S}}{\operatorname{argmin}}\, R_\lambda^{\text{val}}$$

5. Fit the model on with D_{train} and D_{valid} using λ*

$$\hat{\boldsymbol{\theta}}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, R_{\lambda^*}(\boldsymbol{\theta}, \mathcal{D})$$

If training data sample size is very small, will have problem.

Cross Validation



For first CV,

1. Fit the model on D_{train} (for each setting of λ) with loss function
2. For each λ, we compute parameter estimate, θ
3. Then evaluate its performance on D_{valid} (red) using validation data.
4. We then pick the value of λ that results in the best validation performance.
5. We have λ_1

6. Repeat five times.
We will have
CV1: $\lambda_{1,}$ $Loss_1$; **CV2**: $\lambda_{2,}$ $Loss_2$; **CV3**: $\lambda_{3,}$ $Loss_3$; **CV4**: $\lambda_{4,}$ $Loss_4$;
CV5: $\lambda_{5,}$ $Loss_5$

7. We then pick the value of λ^* that results in the best validation performance (lowest loss).

8. Fit the model on with D_{train} and D_{valid} using λ^*