STAT 535 10/25/22

Lecture 8

Underfitting, overfitting, complexity
Neural Networks - 2 layers

Hw2 - due tommorow (some edits)
Hw3 - t.b. posted —"—

Q1 results

Lecture II: Prediction – Basic concepts

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

October, 2022

Parametric vs non-parametric ✓

Generative and discriminative models for classification ✓

- Generative classifiers
- Discriminative classifiers
- Generative vs discriminative classifiers

Loss functions ✓

- Bayes loss

Variance, bias and complexity $\leftarrow$

Reading HTF Ch.: 2.1–5,2.9, 7.1–4 bias-variance tradeoff, Murphy Ch.: 1., 8.6[1], Bach Ch.:

[1] Neither textbook is close to these notes except in a few places; take them as alternative perspectives or related reading

The "learning" problem

- **Given**
- a problem (e.g. recognize digits from $m \times m$ gray-scale images)
- a **sample** or (**training set**) of **labeled data**

$$\mathcal{D} = \{(x^1, y^1), (x^2, y^2), \ldots (x^n, y^n)\}$$

  drawn i.i.d. from an unknown P_{XY}
- **model class** $\mathcal{F} = \{f\}$ = set of predictors to choose from

- **Wanted**
- a predictor $f \in \mathcal{F}$ that performs well on future samples from the same P_{XY}
  - "choose a predictor $f \in \mathcal{F}$" = training/learning
  - "performs well on future samples" (i.e. f **generalizes** well) – how do we measure this? how can we "guarantee" it?
  - choosing $\mathcal{F}$ is the **model selection problem** – about this later

Bias and variance: definitions (never to be used again)

Preliminaries

- What we have a data source P_{XY} and a class of predictors $\mathcal{F}$
- From P_{XY} we sample i.i.d. $\mathcal{D}_N$ of size n. Hence $\mathcal{D}_N \sim P_{XY}^n$. []

Bias and Variance as in Intro Stat Theory

- We want to estimate a parameter $\theta \in \Theta \subseteq \mathbb{R}$
- We use $\mathcal{D}_N$ to obtain estimator $\hat{\theta}_{\mathcal{D}_N}$ which is a function of $\mathcal{D}_N$.
- $\mathcal{D}_N$ is random, hence so is $\hat{\theta}_{\mathcal{D}_N}$.
- Bias$\cancel{=}$ $(\hat{\theta}_{\mathcal{D}_N}) = E_{P^n}[\hat{\theta}_{\mathcal{D}_N}] - \theta$
- Variance$= Var_{P^n}(\hat{\theta}_{\mathcal{D}_N})$
  Both Bias and Variance are computed under the distribution from which we sampled $\mathcal{D}_N$, denoted by P^n.

Bias and Variance for us

- We use $\mathcal{D}_N$ to estimate $\hat{f}_N \in \mathcal{F}$

$$\hat{f}_{\mathcal{D}_N} = \operatorname*{argmin}_{f \in \mathcal{F}} \hat{L}(f, \mathcal{D}_N) \tag{15}$$

- $\mathcal{D}_N$ is random, hence so if $\hat{f}_N$.
- Main differences
  1. $\hat{f}$ is a function!
  2. We are interested in the predictions and not the parameters of $\hat{f}$.
- Several proposals to define bias and variance exist.
- Bias and variance are properties of $\mathcal{F}$.

What we need to know in this course is qualitative

Bias & Var in Prediction

$$P_{XY}^n \rightarrow \underset{\text{random}}{\mathcal{D}_n} \Big\} \mathcal{F} \xrightarrow[\text{alg}]{\text{Learning}} \hat{f}_{\mathcal{D}_n} : \overset{\text{random}}{\mathbb{R}^d} \rightarrow \underset{\text{output}}{y} \quad \underset{\text{labels}}{\mathbb{R}} \longrightarrow \underbrace{L(\hat{f}_{\mathcal{D}_n})}_{\text{random}} = E_{P_{XY}}[L(Y, \hat{f}_{\mathcal{D}_n}(X))]$$

ML Algo

Theory

$$E_{P_{XY}^n}[\hat{f}_{\mathcal{D}_n}] - f^{true} \xrightarrow{??} \text{No } f^{true} \text{ !!}$$

No $E[\quad]$ on y when $y \notin \mathbb{R}$

- **Case I** $\quad y = f^{true}(x) + \varepsilon \qquad \varepsilon \sim \text{iid} \quad E[\varepsilon] = 0, \ \text{Var}\,\varepsilon = \sigma^2 \qquad y, f(x) \in \mathbb{R}$

typical for L_{LS}, Regression

$$\underbrace{E_{P_{XY}^n}[(\hat{f}_{\mathcal{D}_n} - f^{true})^2]}_{} = E_{P_{XY}^n}\Big[\boxed{\Big(\hat{f}_{\mathcal{D}_n} - \boxed{E_{P_{XY}^n}[\hat{f}_{\mathcal{D}_n}]}\Big)^2} + \Big(\boxed{E_{P_{XY}^n}[\hat{f}_{\mathcal{D}_n}]} - f^{true}\Big)^2 - 2\underbrace{(\quad)(\quad)}_{E_{P_{XY}^n} = 0}\Big]$$

at $x \in \mathbb{R}$: L_{LS} at x

integrated over x : $E_{P_{XY}^n}[L_{LS}(\hat{f}_{\mathcal{D}_n})]$

fction of x

$$= \underbrace{E_{P_{XY}^n}[(\hat{f}_{\mathcal{D}_n} - E_{P_{XY}^n}[f_{\mathcal{D}_n}])^2]}_{\text{Var } \hat{f}_{\mathcal{D}_n}} + \underbrace{(E_{P_{XY}^n}[\hat{f}_{\mathcal{D}_n}] - f^{true})^2}_{\text{Bias}^2}$$

smoother ⇒ "easier to fit"

Bias–Var decomposition of L_{LS}

Bias as model (mis)fit

Case II (general case)

The qualitative meaning of bias we will use has to do with the ability of the model class $\mathcal{F}$ to fit the data $\mathcal{D}_N$

- We measure the misfit by the loss L associated with the task, i.e $\hat{L}(\hat{f}_{\mathcal{D}_N}, \mathcal{D}_N)$
- **Bias**$(\mathcal{F}) = E_{P(X,Y)^n}[\hat{L}(\hat{f}_{\mathcal{D}_N}, \mathcal{D}_N)]$ (hence, bias is expected empirical loss).

How well training set is fit

any L

no f^{true}

- Richer model classes have less bias

$$\mathcal{F} \subset \mathcal{F}' \quad \text{then bias}(\mathcal{F}) \geq \text{bias}(\mathcal{F}')$$

- Larger data are harder to fit (hence more bias on average)[3]

- Qualitative analysis
  - Variance
  - Bias

$$\underbrace{E_{P_{XY}^N}}_{\text{over } \mathcal{D}_n} \left[\left(\hat{f}_{\mathcal{D}_n}(x) - E[\hat{f}_{\mathcal{D}_n}(x)] \right)^2 \right] = V(x)$$

$$\mathrm{Var}(\hat{f}_{\mathcal{D}_n}) = \int_{\mathbb{R}^d} V(x)\, dP_X$$

[3]Not trivial, to find a reference. $\rightarrow$ VC dim

Bias as model (mis)fit

The qualitative meaning of bias we will use has to do with the ability of the model class $\mathcal{F}$ to fit the data $\mathcal{D}_N$.

- We measure the misfit by the loss L associated with the task, i.e $\hat{L}(\hat{f}_{\mathcal{D}_N}, \mathcal{D}_N)$
- **Bias**($\mathcal{F}$)= $E_{P(X,Y)^n}[\underline{\hat{L}(\hat{f}_{\mathcal{D}_n}, \mathcal{D}_n)}]$ (hence, bias is expected empirical loss).
- Richer model classes have less bias

$$\text{bias} \searrow$$

$$\underline{\mathcal{F}} \subset \underline{\mathcal{F}'} \quad \text{then bias}(\mathcal{F}) \geq \text{bias}(\mathcal{F}')$$

- Larger data are harder to fit (hence more bias on average)[3]

$$\begin{tikzcd} \mathcal{D}_n \arrow[r, "\text{train}"] & \hat{f}_{\mathcal{D}_n} \arrow[r, "\text{want}"] \arrow[dr] & L(\hat{f}_{\mathcal{D}_n}) \\ \mathcal{F} & & \underline{\hat{L}(\hat{f}_{\mathcal{D}_n})} \end{tikzcd}$$

"training error"

known

"good fit" $\approx \hat{L} \approx 0$

[3]Not trivial, to find a reference.

Sampling variance

- Intuition: if we draw two different data sets $\mathcal{D}, \mathcal{D}' \sim P_{XY}$ (from the same distribution) we will obtain different predictors f, f'. Variance measures how different the predictions of f, f' can be on average.
- **Variance** at $x = Var_{P^n_{XY}}(\hat{f}_{\mathcal{D}_N}(x))$, where the randomness is over the sample $\mathcal{D}_N$
- **Variance** associated with predictor class $\mathcal{F}$ is the expectation over P_X of the variance at x, i.e $E_{P_X}[Var_{P^n_{XY}}(\hat{f}_{\mathcal{D}_N}(x))]$
- Variance depends on n, $\mathcal{F}$, and the data distribution P_{XY} Exercise If $P_{Y|X}$ is deterministic for all x, does it mean that the variance is 0?
- Richer model classes are subject to more variance

$$\mathcal{F} \subset \mathcal{F}' \quad \text{then}\, Var(\mathcal{F}) \leq Var(\mathcal{F}') \quad \text{for any } f^*$$

Var ↗ for $\mathcal{F}'$ larger

Bias, Var depend on
- $\mathcal{F}$
- P_{XY} (or on P^w_{XY})
- n

$n \nearrow \Rightarrow$ Bias ↗ (or stay ct)
Var ↘

Variance, bias and model complexity

- Synonyms: rich class = complex model = flexible model = high modeling power = many degrees of freedom = many parameters

- Evaluating the model complexity[4]/number of free parameters of a model class $\mathcal{F}$ is usually a difficult problem!
  Non-parametric models # parameters depends on P_{XY}, smoothing parameter and n
  Parametric models # parameters NOT always equal to the number of parameters of f!

Example the classifier $f(x) = \operatorname{sgn}(\alpha x), x, \alpha \in \mathbb{R}$ depends on one parameter α but has ∞ degrees of freedom[5]!

Example the linear classifier and regressor on $\mathbb{R}^d$ has (no more than) $n+1$ degrees of freedom

Example the complexity of a two layer neural net with m fixed is not known (but there are approximation results); the number of weights in f is obviously $(m+1)(n+1)+1$

Example For K-NN, the $\overset{\text{complexity}}{\text{variance}}$ increases when K decreases

Example For pruned Decision Tree, the $\underset{\text{complexity}}{\text{variance}}$ increases whith the number of levels

- The variance of a predictor increases with the complexity of $\mathcal{F}$.
- But complexity is the opposite of bias, so bias decrease with the complexity of $\mathcal{F}$
- This is known as the **Bias-Variance tradeoff**

[4] There are several definitions of model complexity, but this holds for all definitions I know

[5] See VC-dimension later

The Bias-Variance tradeoff

Wanted property (for an $\mathcal{F}$)	unwanted consequence of $\mathcal{F}$ not satisfying this property	what to do richer $\mathcal{F}$
to fit $\mathcal{D}$ well	Bias	increase complexity
to be robust to sampling noise	Variance	decrease complexity more restricted $\mathcal{F}$

The **bias-variance tradeoff** is the observation that the better a predictor class $\mathcal{F}$ is able to fit any given sample, the more sensitive the selected f will be to sampling noise.
In this course we will learn some ways of balancing these desired properties (or these undesired consequences).

n fixed

$n \nearrow$ $\quad$ $Var(\mathcal{F}) \searrow$

$Bias(\mathcal{F}) \nearrow$ or same

Examples, examples...

Example (K-nearest neighbor classifiers)

The 1-NN can fit any data set perfectly (every data point is it's own nearest neighbor). But for $K > 1$, the K-NN may not be able to reproduce any pattern of ± 1 in the labels. Hence its bias is larger than the bias of the 1-NN classifier. With the variance, the opposite happens: as K the number of neighbors increases, the decision regions of the K-NN classifier become more stable to the random sampling effects. Thus, the variance decreases with K.

Example (Linear vs quadratic vs cubic ... predictors)

The quadratic functions include all linear functions, the cubics include all quadratics, and so on. Linear classifiers will have more bias (less flexibility) than quadratic classifiers. On the other hand, the variance of the linear classifier will be lower than that of the quadratic. The case of regression is even more straightforward: if we fit the data with a higher degree polynomial, the fit will be more accurate, but the variation of the polynomial $f(x)$ for x values not in the training set will be higher too.

Example (Kernel regression)

Examples, examples... (2)

The bias-variance tradeoff can be observed on a continuous range for **kernel regression**. When the kernel width h is near 0, $f(x)$ from Lecture 1, equation (25) will fit the data in the training set exactly [Exercise: prove this], but will have high variance. When h is large, $f(x^i)$ will be smoothed between x^i and the other data points nearby, so it may be some distance from y^i. However, precisely because $f(x)$ is supported by a larger neighborhood, it will have low variance. [Exercise: find some intuitive explanations for why this is true] Hence, the smoothness parameter h controls the trade-off between bias and variance.

Example (Regularization)

The same can be observed if one considers equation (**??**). For $\lambda = 0$, one choses f that best fits the data (minimizes $\hat{L}$. For $\lambda \to \infty$, f is chosen to minimize the penalty J, disregarding the data completely. The latter case has 0 variance, but very large bias. Between these extreme cases, the parameter λ controls the amount in which we balance fitting the data (variance) with pulling f towards an a-priori "good" (bias).

Overfitting and Underfitting

- Bias and variance are properties of the model class $\mathcal{F}$ (sometimes toghether with the learning algorithm – more about this later). They are not properties of the parameters of f (e.g β), and not of a particular $f \in \mathcal{F}$.
- Variance decreases to 0 with n, but bias may not. This implies that for larger sample sizes n, the trade-off between variance and bias changes, and typically the “best” trade-off, aka the best model, will have larger complexity.
- **Overfitting**= is the situation of small bias and too much variance (i.e. $\mathcal{F}$ is too complex). In practice, if a learned predictor f has low $\hat{L}(f)$ but significantly higher $L(f)$, we say that the model has *overfit* the data $\mathcal{D}$. (Of course we cannot know $L(f)$ directly, and a significant amount of work in statistics is dedicated to predicting $L(f)$ for the purpose of chosing the best model.)
- **Underfitting**=bias is too high, or the model is too simple (a.k.a has too few degrees of freedom). [Exercise: what do you expect to see w.r.t. $\hat{L}(f)$ v.s. $L(f)$ for an underfitted model?]

Complexity, even though there are variations in its definition, and although it is not known exactly for most model classes, is at the core of learning theory, the part of statistical theory that gives provable results about the expected loss of a predictor.

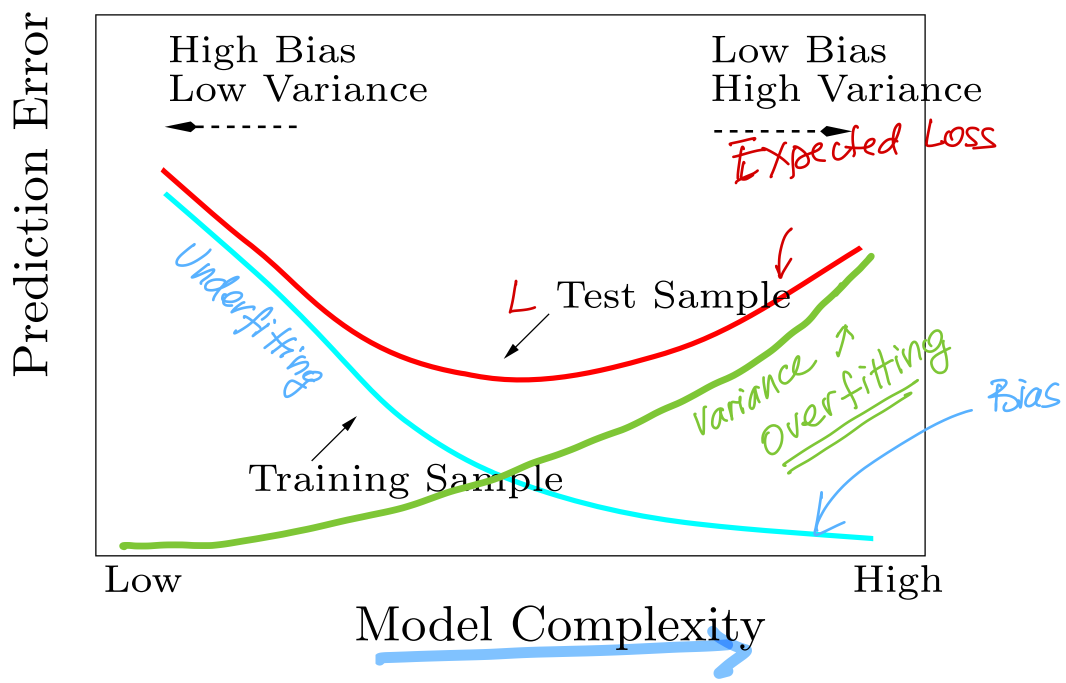


FIGURE 2.11. *Test and training error as a function of model complexity.*

Knn : K $\rightarrow$

Kernel r: h $\rightarrow$

Decision trees: # splits ↗

Polynomial regression: linear ⊂ quadratic ⊂ cubic ⊂

degree ↗

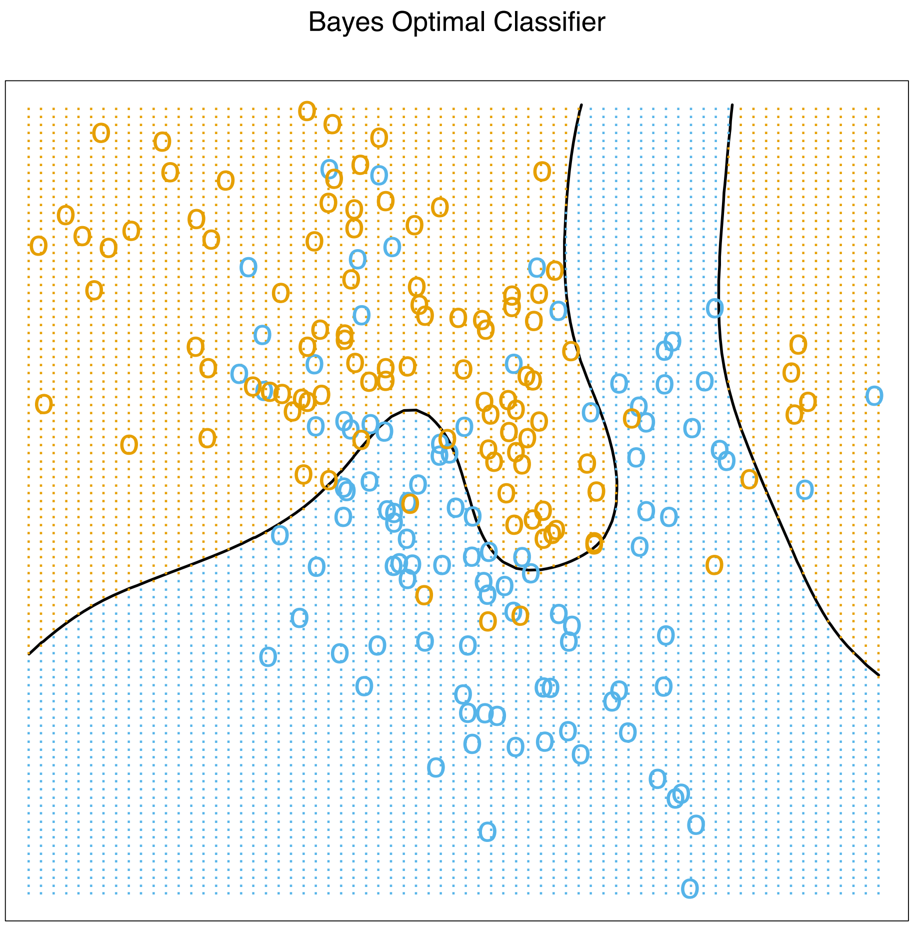


FIGURE 2.5. *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

Ex. of overfitting

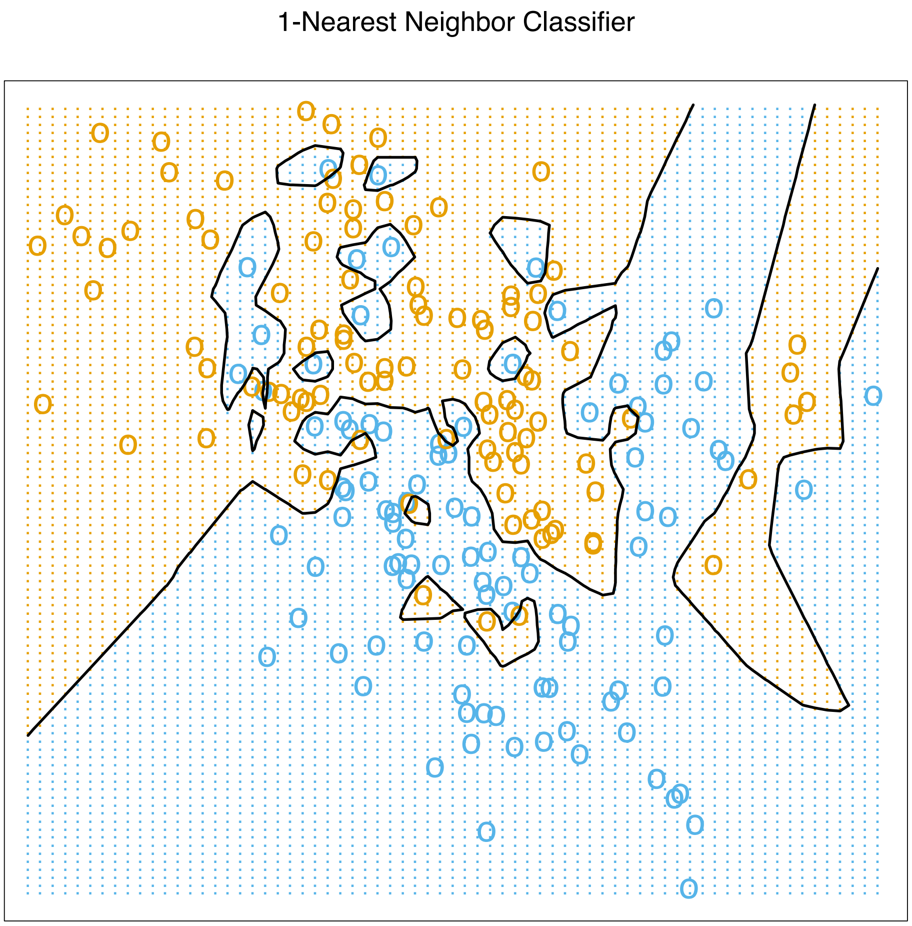


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

Underfitting

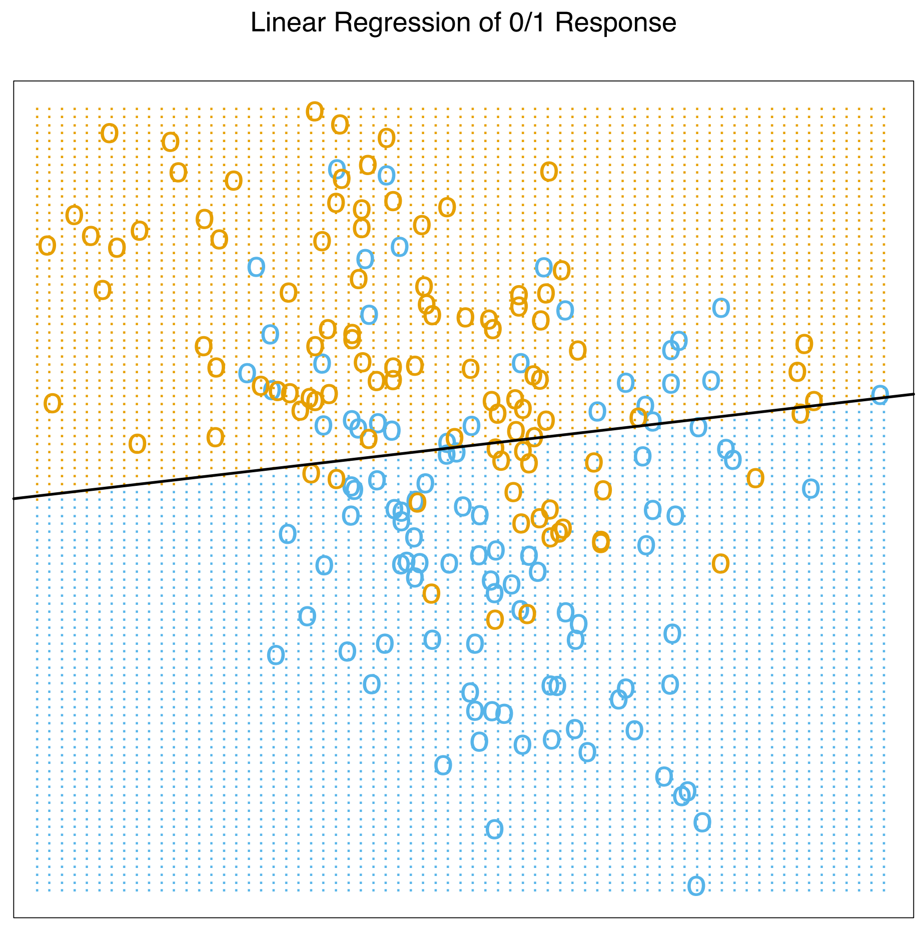


FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (*BLUE *= 0,* ORANGE *= 1), and then fit by linear regression. The line is the decision boundary defined by $x^T\hat{\beta} = 0.5$. The orange shaded region denotes that part of input space classified as* ORANGE*, while the blue region is classified as* BLUE.

Best Bias-Var Trade-off for $n = 200$

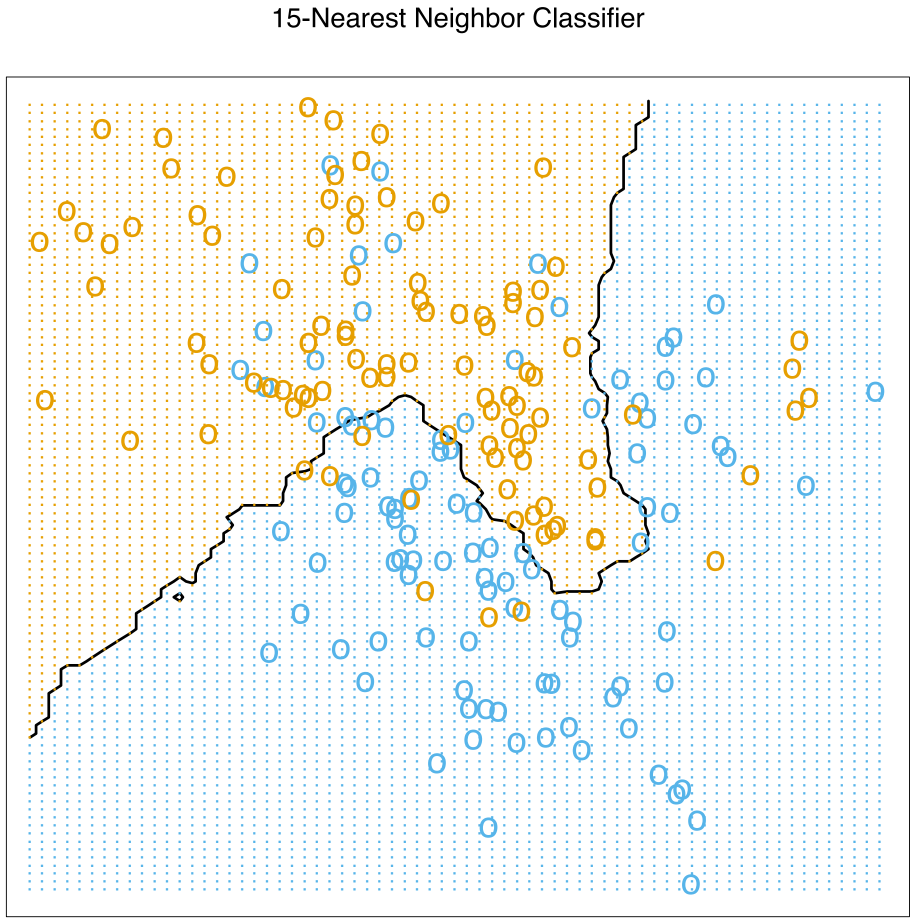


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

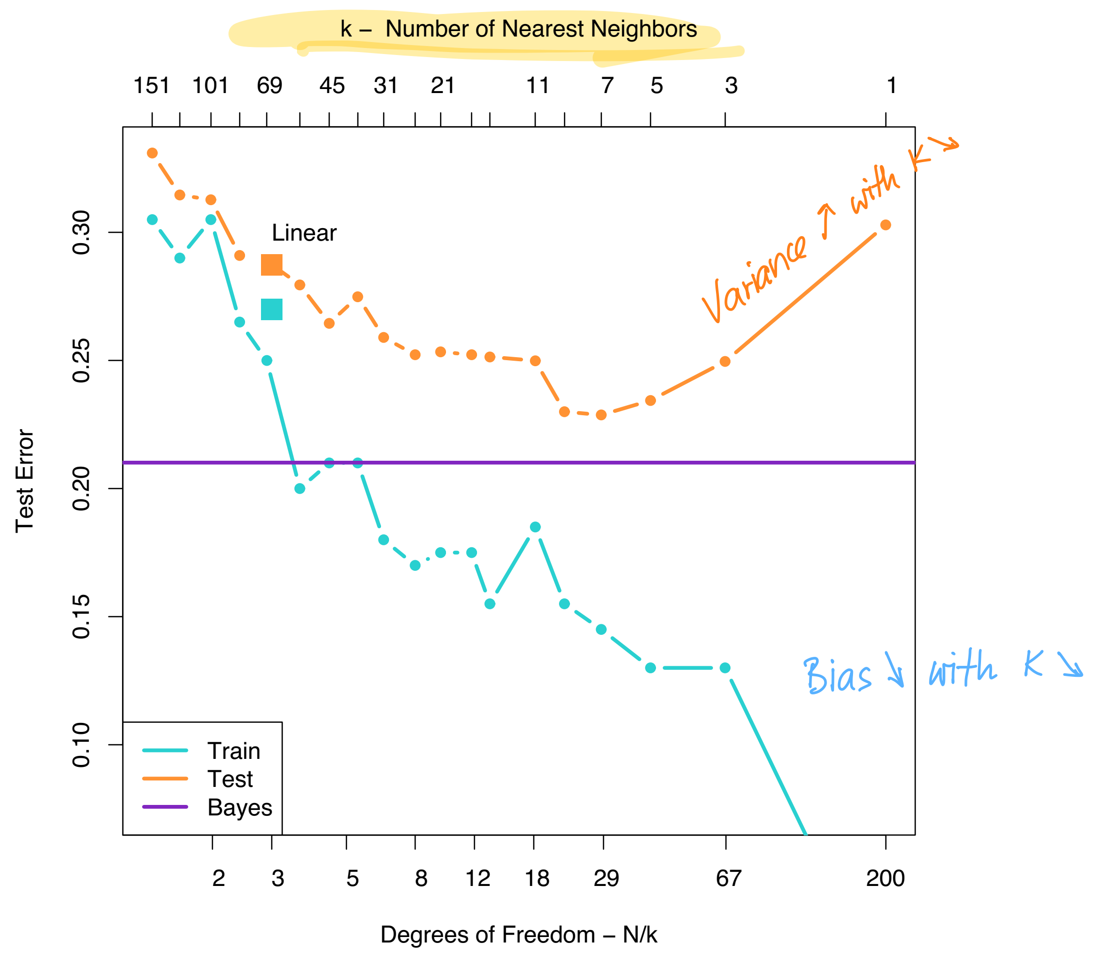


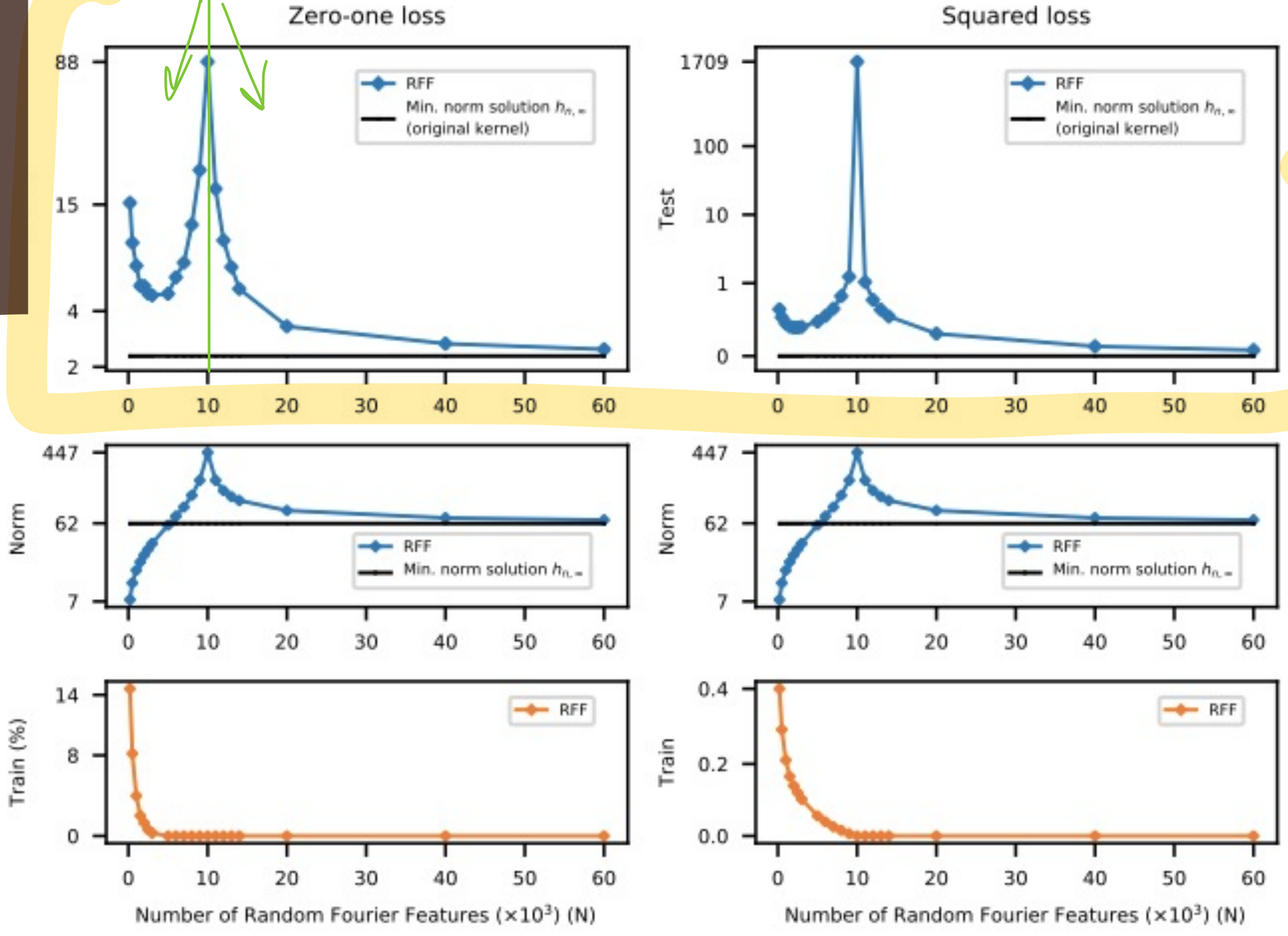
FIGURE 2.4. *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size* 200 *was used, and a test sample of size* 10,000*. The orange curves are test and the blue are training error for k-nearest-neighbor classification. The results for linear regression are the bigger orange and blue squares at three degrees of freedom. The purple line is the optimal Bayes error rate.*

Double descent: beyond the bia—variance tradeoff

interpolation limit

Gradient Descent

$\uparrow$ f smooth



complexity

Lecture Notes III – Neural Networks

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

October, 2020

Two-layer Neural Networks $\leftarrow$

Multi-layer neural networks

A zoo of multilayer networks

Reading HTF Ch.: 11.3 Neural networks, Murphy Ch.: (16.5 neural nets) and Dive Into Deep Learning 4.1-4.3

Two-layer Neural Networks

- The **activation function** (a term borrowed from neuroscience) is any continuous, bounded and strictly increasing function on $\mathbb{R}$. ~~Almost universa~~lly, the activation function is the **logistic** (or **sigmoid**)

$$\phi(z) = \frac{1}{1+e^{-z}} \tag{1}$$

because of its nice additional computational and statistical properties.

- We build a **two-layer neural network** in the following way:

$x \in \mathbb{R}^d$

Inputs	x_k	$k = 1 : d$
hidden Bottom layer[1]	$z_j = \phi(w_j^T x)$	$j = 1 : m,\ w_j \in \mathbb{R}^d$
Top layer	$f = \phi(\beta^T z)$	$\beta \in \mathbb{R}^m$
Output	f	$\in [0, 1]$

parameters
$\beta \in \mathbb{R}^m$
$W \in \mathbb{R}^{d \times m}$

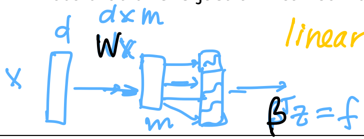
In other words, the neural network implements the function

$$f(x) = \underbrace{\sum_{j=1}^{m} \beta_j z_j}_{\text{linear}} = \sum_{j=1}^{m} \beta_j \phi\Big(\underbrace{\sum_{k=1}^{d} w_{kj} x_k}_{\text{linear}}\Big) \in (-\infty, \infty) \tag{2}$$

Note that this is just a linear combination of logistic functions.

m "units" in hidden layer

$z_j \quad j = 1:m$



[1] In neural net terminology, each variable z_j is a **unit**, the bottom layer is **hidden**, while top one is **visible**, and the units in this layer are called hidden/visible units as well. Sometimes the inputs are called **input units**; imagine neurons or individual circuits in place of each x, y, z variable.

Output layer options

- **linear** layer as in (2) $f = \sum_j \beta_j z_j$
- **logistic** layer: in classification $f(x) \in [0,1]$ is interpreted as the probability of the $+$ class.

$$f(x) \;=\; \phi\left(\sum_{j=1}^{m} \beta_j z_j\right) \;=\; \phi\left(\sum_{j=1}^{m} \beta_j \phi(\sum_j w_{kj} x_k)\right) \tag{3}$$

- **softmax** layer in multiway classification

  The softmax function $\phi(z) : \mathbb{R}^m \to (0,1)^m$

$$\phi_k(z) \;=\; \frac{e^{z_k}}{\sum_{j=1}^{m} e^{z_j}} \tag{4}$$

- Properties
  - $\sum_{j=1}^{m} \phi_j(z) = 1$ for all z
  - for $z_k \gg z_j$, $j \neq k$ $\phi_k(z) \to 1$.
  - derivatives $\frac{\partial \phi_j}{\partial z_k} \;=\; \phi_k \delta_{jk} - \phi_j \phi_k$

Generalized Linear Models (GLM)

A GLM is a regression where the "noise" distribution is in the exponential fami ly.

- $y \in \mathbb{R}$, $y \sim P_\theta$ with
$$P_\theta(y) \;=\; e^{\theta y - \ln \psi(\theta)} \tag{5}$$
- the parameter θ is a linear function of $x \in \mathbb{R}^d$
$$\theta \;=\; \beta^T x \tag{6}$$
- We denote $E_\theta[y] = \mu$. The function $g(\mu) = \theta$ that relates the mean parameter to the natural parameter is called the **link function**.

The log-likelihood (w.r.t. β) is
$$l(\beta) \;=\; \ln P_\theta(y|x) \;=\; \theta y - \psi(\theta) \quad \text{where } \theta = \beta^T x \tag{7}$$
and the gradient w.r.t. β is therefore
$$\nabla_\beta l \;=\; \nabla_\theta l \nabla_\beta(\beta^T x) \;=\; (y-\mu)x \tag{8}$$
This simple expression for the gradient is the generalization of the gradient expression you obtained for the two layer neural network in the homework. [Exercise: This means that the sigmoid function is the *inverse link function* defined above. Find what is the link function that corresponds to the neural network.]

Hidden layer options

- sigmoidal functions ϕ, *tanh*
- hinge functions RELU $= max(z, 0)$, softplus $= \ln(1 + e^z)$

Rectified Linear Unit