Simulation: kNN vs Linear Regression

- Review two simple approaches for supervised learning:
  - k-Nearest Neighbors (kNN), and
  - Linear regression
- Then examine their performance on two simulated experiments to highlight the trade-off between bias and variance.

k-Nearest Neighbors

$$\hat{y} = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i,$$

where $N_k(x)$ denotes k samples from the training data, which (in terms of their x values) are closest to x.

Regression: kNN predicts y by a local average.

Classification: kNN can return the majority vote in $N_k(x)$, e.g., $\hat{y} = 1$ if $\frac{1}{k} \sum_{x_i \in N_k(x)} y_i > 0.5$ assuming $y \in \{1, 0\}$, or return a probability vector calculated based on the frequencies in $N_k(x)$.

What are the input parameters for kNN?

One input parameter is k, the neighborhood size.

- For 1NN ($k = 1$), the prediction at x_i (the i-th training sample) is exactly y_i, i.e., zero training error.
- For nNN ($k = n$), every neighborhood contains all the n training samples, so the prediction is the same no matter what value x takes.

The complexity or the dimension of kNN is roughly equal to n/k.

No magic value for k. It is a tuning parameter of the algorithm and is usually chosen by cross-validation.[a]

[a] If you don't know what cross-validation is, read chap 5.1 in ISLR.

The other input parameter is the metric, which we use to define the neighborhood.

The default is the Euclidean distance on the p-dimension feature vector $x \in \mathbb{R}^p$. However, it could be the weighted Euclidean, e.g.,

$$d(x, \tilde{x}) = \sum_{j=1}^{p} w_j (x_j - \tilde{x}_j)^2,$$

and we would like to learn the weights w_j's from the data.

It does not need to be Euclidean, as long as it is a similarity measure for any two samples, e.g., in image classification (from Flickr), we can measure the similarity of two images by their physical similarity, or by the similarity of their tags, or by the percentage of people who like both images.

Linear Regression

- In linear regression models, we approximate Y by a linear function of X:

$$f(\mathbf{x}) \approx \beta_0 + x_1\beta_1 + \cdots x_p\beta_p,$$

and estimate $\hat{\beta}_j$'s using the so-called Least Squares (LS) principle

$$\min_{\beta_0,\ldots,\beta_p} \sum_{i=1}^{n} (y_i - \beta_0 - x_{i1}\beta_1 \cdots - x_{ip}\beta_p)^2.$$

The solution is easy to compute – call command `lm` in `R`.

- We can also apply linear regression on classification problems with $Y = 0/1$, and predict Y to be 1 if the LS prediction $f(x)$ is bigger than 0.5 and 0 otherwise.

- There are some drawbacks with LS for classification:

  1) The squared difference $(y_i - f(\mathbf{x}_i))^2$ is not a good evaluation metric for classification;

  2) Ideally we would like to estimate the $\mathbb{P}(Y = 1 \mid X = \mathbf{x})$, however the linear function $f(\mathbf{x})$ could return us values outside $[0, 1]$.

  Later we'll learn a generalization of LS, called logistic regression model, where we assume the logit of the probability is a linear function:

$$\log \frac{p(\mathbf{x})}{1 - p(\mathbf{x})} \approx \beta_0 + x_1\beta_1 + \cdots x_p\beta_p.$$

- Despite some of the drawbacks, the LS approach for classification works reasonably well in practice; plus its computation is very fast. So we'll apply LS on the two toy examples.

Two Toy Examples

- Let's look at the performance of these two approaches on two simulated binary classification examples from chap 2 (ESL).
- Example I: The data in each class are generated from a Gaussian distribution and the two Gaussians have different means.
- Example II: The data in each class are generated from a mixture of 10 Gaussians in each class.
- Check the `R` code and lec_Introduction_kNN_vs_LinearReg_figs.pdf posted on the course website.

Compute the Bayes Rule: Example 1

$$\begin{aligned} Y &\sim \text{Bern}(p), \\ X \mid Y = 0 &\sim \text{N}(\mu_0, \sigma^2 \mathbf{I}_2), \\ X \mid Y = 1 &\sim \text{N}(\mu_1, \sigma^2 \mathbf{I}_2). \end{aligned}$$

The joint dist can be factorized as $P(Y, X) = P(Y) \times P(X \mid Y)$. All the calculation on the next slide is to use Bayes' theorem to compute $P(Y \mid X)$.

Note that $P(Y, X)$ is neither a pmf (for discrete r.v.) nor a pdf (for continuous r.v.). It will get a little technical if I tend to rigorously justify my calculation on the next slide. Let's ignore the technicality, and just treat X as discrete with pmf same as its density function. Trust me, the result is correct.

$$
\begin{aligned}
P(Y=1 \mid X=x) &= \frac{P(Y=1, X=x)}{P(X=x)} \\
&= \frac{P(Y=1, X=x)}{P(Y=1, X=x) + P(Y=0, X=x)} \\
&= \frac{P(Y=1)P(X=x|Y=1)}{P(Y=1)P(X=x|Y=1) + P(Y=0)P(X=x|Y=0)} \\
&= \frac{(p)\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^2 \exp\left\{-\frac{\|x-\mu_1\|^2}{2\sigma^2}\right\}}{(p)\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^2 \exp\left\{-\frac{\|x-\mu_1\|^2}{2\sigma^2}\right\} + (1-p)\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^2 \exp\left\{-\frac{\|x-\mu_0\|^2}{2\sigma^2}\right\}} \\
&= \left[1 + \exp\left\{\frac{1}{2\sigma^2}(\|x-\mu_1\|^2 - \|x-\mu_0\|^2) - \log\frac{p}{1-p}\right\}\right]^{-1}
\end{aligned}
$$

$$P(Y = 1 \mid X = x) > 0.5 \Longleftrightarrow \frac{1}{2\sigma^2}(\|x - \mu_1\|^2 - \|x - \mu_0\|^2) < \log \frac{p}{1-p}$$

- If $p = 0.5$, then we basically predict $Y = 1$ if x is closer to μ_1, and 0 if closer to μ_0.
- The formula on the previous slide needs to be updated if the two normal components have different variances.
- The Bayes error for Example II can be computed similarly, in which $P(X \mid Y)$ is a weighted sum of 10 normal densities.

kNN *vs.* Linear Regression

- Linear regression: f is linear
  - low variance: need to estimate $p = 3$ parameters
  - high bias (underfit): linear assumption is very restrictive
- kNN: no assumption on f, except local some smoothness.
  - low bias (overfit): flexible and adaptive. It can be shown that as $k, n \to \infty$ such that $k/n \to 0$, kNN is consistent.
  - high variance: num of parameters for kNN is roughly n/k, which goes to ∞ in order to achieve consistency.