Week 5: Cross-validation

MATH-517 Statistical Computation and Visualization

Linda Mhalla

2024-10-11

Motivation

Over the last two lectures, we've covered **KDE** and **non-parametric regression methods**

Both required the choice of a certain tuning parameter

- KDE, $h > 0$

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)$$

- Local Polynomial Regression (with a fixed degree p), $h > 0$

$$\underset{\beta \in \mathbb{R}^{p+1}}{\arg\min} \sum_{i=1}^{n} \{Y_i - \beta_0 - \beta_1(X_i - x) - \ldots - \beta_p(X_i - x)^p\}^2 K\left(\frac{X_i - x}{h}\right)$$

Motivation

Many other modern methods for regression can be expressed as

- penalized regression

$$\arg\min_{\beta} \sum_{n=1}^{N} \left(y_n - x_n^\top \beta\right)^2 + \lambda R(\beta),$$

where R is a penalty, e.g., $\|\cdot\|_2^2$ for ridge regression or $\|\cdot\|_1$ for lasso

$\Rightarrow$ regularize to get sparsity or reduce variability (multicollinearity), or

- smoothing splines

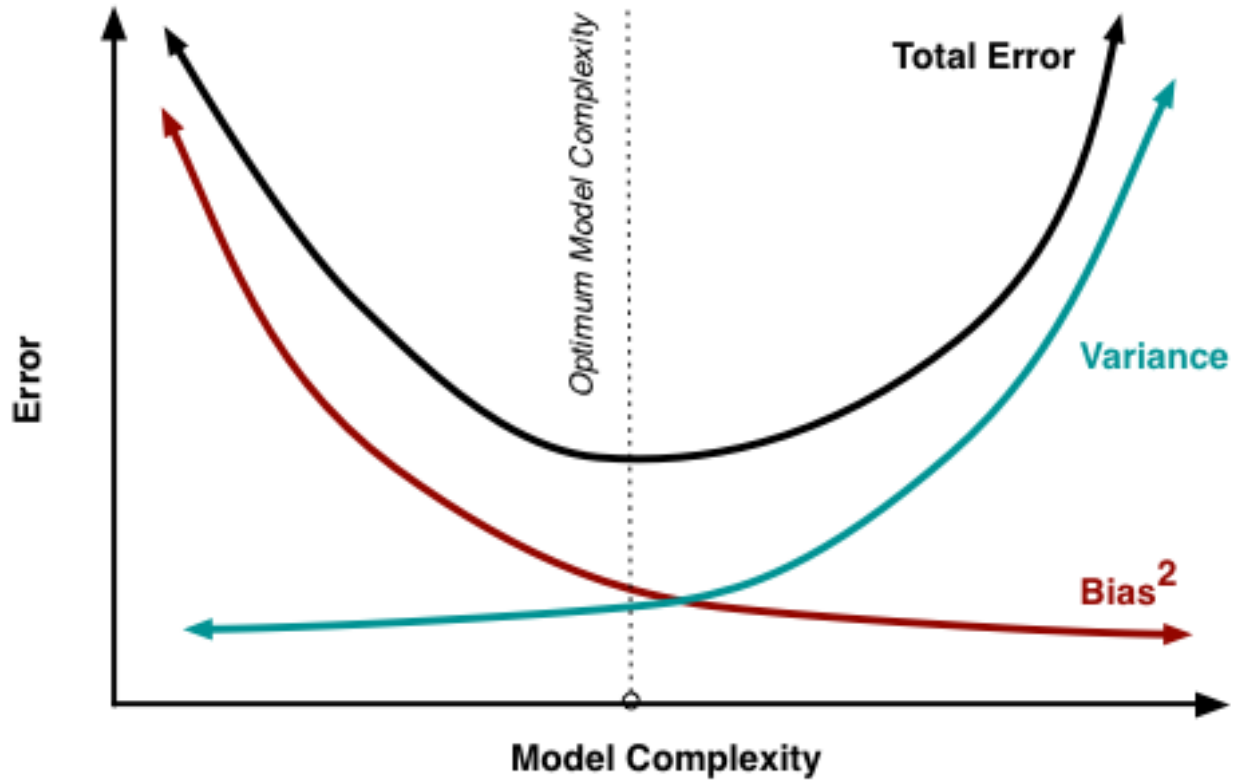
$$\arg\min_{\beta} \sum_{n=1}^{N} \{y_n - f(x_n)\}^2 + \lambda \int \{f''(x)\}^2 dx$$

$\Rightarrow$ regularize to induce smoothness

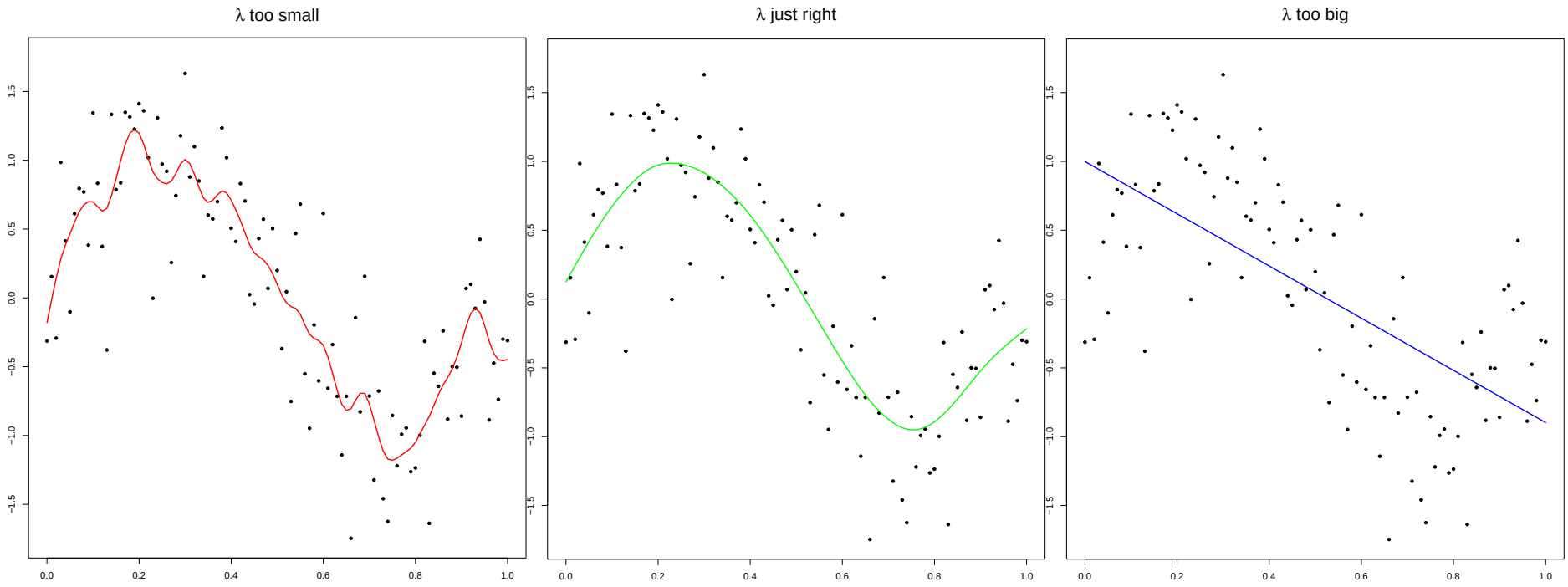
In all cases, **cross-validation (CV)** can be used to select the tuning parameters

- not always straightforward!

Bias-variance Trade-off



Bias-variance Trade-off: Smoothing splines



Purpose of Cross-validation

Cross-validation (CV) is a very general method for

- tuning the **regularization parameter** of a method
- estimating the **predictive** power of a method

Since training an algorithm and evaluating its performance on the same data yields an overoptimistic result, CV fixes the issue by testing the output of a method on (independent) "new data"

CV estimates the prediction error of an algorithm (e.g., regression or classification)

$$\text{TestError}(\hat{f}_\lambda) = \frac{1}{N}\sum_{n=1}^{N} \ell\{y'_n, \hat{f}_\lambda(x'_n)\}$$

that is different from

$$\text{TrainError}(\hat{f}_\lambda) = \frac{1}{N}\sum_{n=1}^{N} \ell\{y_n, \hat{f}_\lambda(x_n)\}$$

Purpose of Cross-validation

CV involves

- splitting a data set into a training data set and a test data set
- fitting the model using the training data set
- using the test data set to evaluate how the model performs (according to a measure of error/risk)
- computing the average over several splits (several splitting strategies exist!)

Section 1

CV for Supervised Problems

Local Polynomial Regression

Setup: A sample $(x_1, y_1)^\top, \ldots, (x_N, y_N)^\top \in \mathbb{R}^2$ from a population $Y = m(X) + \epsilon$ with $X \perp\!\!\!\perp \epsilon$. For a fixed bandwidth h, we estimate $m(x) = \mathbb{E}(Y|X = x)$ as $\widehat{m}_h(x)$ by, e.g., local linear regression.

Question: How to choose h? (i.e., how to obtain a good bias-variance trade-off?)

What is the measure of how good our estimator $\widehat{m}_h(x)$ for a given bandwidth is?

$$MISE(\widehat{m}_h) = \int \mathbb{E}\{\widehat{m}_h(x) - m(x)\}^2 f_X(x) dx$$

- let's choose h that minimizes the (density-weighted) MISE

Here, what matters is to minimize the estimation error on the regions where the density of X is higher

Local Polynomial Regression

But we don't know m. How about using the average RSS

$$\frac{1}{N}\sum_{n=1}^{N}\left\{Y_n - \widehat{m}_h(X_n)\right\}^2$$

as a proxy for the MISE?

That's a *bad idea*, because $\left\{Y_n - \widehat{m}_h(X_n)\right\}^2 \to 0$ for $h \to 0$

- this is called *overfitting* (useless interpolation)
- the problem lies in validating on data used to fit the model (favours estimates too well-adapted to data and unreasonable for new obs.)

Instead, consider this to approximate the MISE:

$$CV(h) = \frac{1}{N}\sum_{n=1}^{N}\left\{Y_n - \widehat{m}_h^{(-n)}(X_n)\right\}^2,$$

where $\widehat{m}_h^{(-n)}(X_n)$ is the model fitted without the n-th observation

CV for Local Polynomial Regression

$$CV(h) = \frac{1}{N}\sum_{n=1}^{N}\left\{Y_n - \widehat{m}_h^{(-n)}(X_n)\right\}^2$$

Since $Y = m(X) + \epsilon$, we can write

$$\begin{aligned} CV(h) &= \frac{1}{N}\sum_{n=1}^{N}\left\{Y_n - m(X_n) + m(X_n) - \widehat{m}_h^{(-n)}(X_n)\right\}^2 \\ &= \frac{1}{N}\sum_{n=1}^{N}\epsilon_n^2 + \frac{2}{N}\sum_{n=1}^{N}\epsilon_n\left\{m(X_n) - \widehat{m}_h^{(-n)}(X_n)\right\} \\ &\qquad + \underbrace{\frac{1}{N}\sum_{n=1}^{N}\left\{m(X_n) - \widehat{m}_h^{(-n)}(X_n)\right\}^2}_{\mathbb{E}_{X,Y}(\star) = MISE\left(\widehat{m}_h\right)}, \end{aligned}$$

$$MISE(\widehat{m}_h) = \mathbb{E}_{X,Y}\left\{\widehat{m}_h(X) - m(X)\right\}^2$$

CV for Prediction

More generally: $(x_1, y_1)^\top, \ldots, (x_N, y_N)^\top \in \mathbb{R}^{p+1}$

Model for prediction: $\widehat{Y} = \widehat{m}(X)$

How good is the model: measured by a loss function, e.g., $\mathbb{E}\{Y - \widehat{m}(X)\}^2$

- other losses possible, e.g., if undershooting better than overshooting

If another data set $(x_1^\star, y_1^\star)^\top, \ldots, (x_M^\star, y_M^\star)^\top$ available (generated by the same process as the original data set), we can approximate loss empirically

$$\frac{1}{M} \sum_{k=1}^{M} \{y_k^\star - \widehat{m}(x_k^\star)\}^2$$

CV is the alternative when no other data set is available:

$$CV(\widehat{m}) := \frac{1}{N} \sum_{n=1}^{N} \{y_n - \widehat{m}^{(-n)}(x_n)\}^2,$$

where $\widehat{m}^{(-n)}$ is the model fitted without the n-th observation

CV for Model Selection

CV can also be used to compare candidate models $\widehat{m}_1, \ldots, \widehat{m}_j$

- can be completely different models
  - typically it is the same model with different tuning parameter values
- select the model for which the CV criterion is minimized
- beware: when not in the "vanilla" iid case (e.g. times series, stratified data, etc.), things are not so straightforward...

But there are computational costs. The model has to be re-fitted for

- all the tuning parameter values considered
- every data point left out
  - actually, this might not be necessary...

Computational Shortcut for Linear Smoothers

If $\widehat{m}$ is a linear smoother, i.e., the predictions $\hat{y}_n = \widehat{m}(x_n)$ are given all together as

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$$

where $\mathbf{S} \in \mathbb{R}^{N \times N}$ depends on x's, then re-fitting (leaving out data points one by one) may not be necessary!

Example: Ridge regression is a linear smoother

$$\arg\min_{\beta} \sum_{n=1}^{N} \left(y_n - x_n^\top \beta\right)^2 + \lambda \|\beta\|_2^2.$$

- $\hat{\beta} = (\mathbf{X}^\top \mathbf{X} + \lambda I)^{-1} \mathbf{X}^\top \mathbf{y}$
- $\hat{\mathbf{y}} = \underbrace{\mathbf{X}(\mathbf{X}^\top \mathbf{X} + \lambda I)^{-1} \mathbf{X}^\top}_{=:\mathbf{S}} \mathbf{y}$

$$CV(\lambda) = \frac{1}{N} \sum_{n=1}^{N} \left\{ y_n - \mathbf{x}_n^\top \hat{\beta}^{(-n)} \right\}^2 = \frac{1}{N} \sum_{n=1}^{N} \left\{ \frac{y_n - \widehat{m}(x_n)}{1 - s_{nn}} \right\}^2$$

Example: Ridge Regression

Noticing $\hat{\beta}^{(-n)} = (\mathbf{X}^\top \mathbf{X} + \lambda I - \mathbf{x}_n \mathbf{x}_n^\top)^{-1}(\mathbf{X}^\top \mathbf{y} - \mathbf{x}_n y_n)$, we can use Sherman-Morrison formula:

- denoting $\mathbf{A} := \mathbf{X}^\top \mathbf{X} + \lambda I$
- $\alpha_n := 1 - \mathbf{x}_n^\top \mathbf{A}^{-1} \mathbf{x}_n$

$$\begin{aligned}\hat{\beta}^{(-n)} &= \left(\mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{x}_n \mathbf{x}_n^\top \mathbf{A}^{-1}}{1 - \mathbf{x}_n^\top \mathbf{A}^{-1} \mathbf{x}_n}\right)(\mathbf{X}^\top \mathbf{y} - \mathbf{x}_n y_n) \\ &= \hat{\beta} - \frac{1}{\alpha_n}(\mathbf{A}^{-1}\mathbf{x}_n \mathbf{x}_n^\top \hat{\beta} - \mathbf{A}^{-1}\mathbf{x}_n y_n).\end{aligned}$$

Plug this back into the general CV formula and do some simple algebra to obtain the last formula on the previous slide

- check out lecture notes for details, if interested

Computational Shortcut for Linear Smoothers

A similar computational shortcut is possible for the linear smoothers

- linear models
- local constant regression
  - what about other polynomial orders?
- ridge regression
- smoothing splines
- KDE (when working on a grid),

as long as the LOOCV is built on the squared error loss

On the other hand, such shortcuts are not possible for

- lasso
- many other penalized or otherwise complicated estimators

When a computational shortcut is impossible, perform K-fold CV instead!

K-fold CV

Divide the set $\{1, \dots, N\}$ into K subsets (*folds*) of approximately equal size, $J_1, \dots, J_K$, such that

- fold $J_k \subset \{1, \dots, N\}$ for $k = 1, \dots, K$ such that $J_k \cap J_{k'} = \emptyset$ for $k \neq k'$ and $\bigcup_{k=1}^{K} J_k = \{1, \dots, N\}$

For $k = 1, \dots, K$:

- Consider training on (x_i, y_i), $i \notin J_k$, and validating on (x_i, y_i), $i \in J_k$
- Fit the model on the training set and compute the error on the validation set

$$e_k = \sum_{n \in J_k} \{y_n - \widehat{m}^{(-J_k)}(x_n)\}^2$$

  where $m^{(-J_k)}$ is the model fitted without the data in the k-th fold J_k
- Compute the average error over all folds

In practice, choose $K = 5$ or $K = 10$, perform random permutation of indices and split the data

K-fold CV

Instead of the (leave-one-out) CV criterion

$$CV(\widehat{m}) := \frac{1}{N}\sum_{n=1}^{N}\left\{y_n - \widehat{m}^{(-n)}(x_n)\right\}^2,$$

use the K-fold CV criterion:

$$CV_K(\widehat{m}) = K^{-1}\sum_{k=1}^{K}|J_k|^{-1}\sum_{n\in J_k}\left\{Y_n - \widehat{m}^{(-J_k)}(X_n)\right\}^2.$$

- requires every candidate model to be fitted K-times
- it is difficult to study properties of $CV_K(\widehat{m})$ properly. One usually examines whether leave-one-out CV works and, if yes and if no computational shortcuts available, resorts to K-fold CV for computational reasons

Properties of the CV schemes

- LOOCV is approximately unbiased for the true prediction error (bias comes from using $N - 1$ observations)
- LOOCV has typically high variance (doesn't shake up the data enough)
- K-fold CV has higher bias than LOOCV (training sets have smaller size)
- $K = 5$ or 10 provides a good compromise for the bias-variance trade-off
- Validation approach estimates the prediction error with high variance (depends on the unique split)

Section 2

CV for Unsupervised Problems

Bandwidth Selection for KDE

Sample $X_1, \ldots, X_N$ from f, goal is to estimate $f(x)$ by

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{N} K\left(\frac{X_i - x}{h}\right)$$

- no response here!

A good estimator (a well-chosen h) minimizes

$$\begin{aligned} MISE(\hat{f}_h) &= \mathbb{E} \int \{\hat{f}_h(x) - f(x)\}^2 dx \\ &= \mathbb{E} \underbrace{\int \{\hat{f}_h(x)\}^2 dx}_{\|\hat{f}_h(x)\|_2^2} - 2 \underbrace{\mathbb{E} \int \hat{f}_h(x) f(x) dx}_{A(h)\text{: the CV idea?}} + \underbrace{\int \{f(x)\}^2 dx}_{\text{no } h \text{ here}}. \end{aligned}$$

Bandwidth Selection for KDE

Sample $X_1, \ldots, X_N$ from f, goal is to estimate $f(x)$ by

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{N} K\left(\frac{X_i - x}{h}\right)$$

- no response here!

A good estimator (a well-chosen h) minimizes

$$\begin{aligned} MISE(\hat{f}_h) &= \mathbb{E} \int \{\hat{f}_h(x) - f(x)\}^2 dx \\ &= \mathbb{E} \underbrace{\int \{\hat{f}_h(x)\}^2 dx}_{\|\hat{f}_h(x)\|_2^2} - 2 \underbrace{\mathbb{E} \int \hat{f}_h(x) f(x) dx}_{A(h)\text{: the CV idea?}} + \underbrace{\int \{f(x)\}^2 dx}_{\text{no } h \text{ here}}. \end{aligned}$$

Let's find an unbiased estimator of $A(h)$!

Bandwidth Selection for KDE

The CV idea: see how your estimator behaves on a left-out datum:

$$\mathbb{E}\hat{f}_h^{(-n)}(X_n) = \mathbb{E}\frac{1}{(n-1)h}\sum_{j\neq n} K\left(\frac{X_n - X_j}{h}\right) = \frac{1}{h}\mathbb{E}_{X_1,X_2}\left\{K\left(\frac{X_1 - X_2}{h}\right)\right\}$$

$$= \int \underbrace{\int \frac{1}{h}K\left(\frac{x-y}{h}\right) f(y)dy}_{\mathbb{E}\hat{f}_h(x)} f(x)dx = \mathbb{E}\int \hat{f}_h(x)f(x)dx.$$

$\Rightarrow N^{-1}\sum_{n=1}^{N} \hat{f}_h^{(-n)}(X_n)$ is an unbiased estimator of $\mathbb{E}\int \hat{f}_h(x)f(x)dx$

Thus, up to the constant (not depending on h), an unbiased estimator of

$$MISE(\hat{f}_h) = \mathbb{E}\int \left[\hat{f}_h(x)\right]^2 dx - 2\mathbb{E}\int \hat{f}_h(x)f(x)dx + \int \left[f(x)\right]^2 dx.$$

is given by the CV

$$CV(h) = \int \left[\hat{f}_h(x)\right]^2 dx - \frac{2}{N}\sum_{n=1}^{N} \hat{f}_h^{(-n)}(X_n)$$

Bandwidth Selection for KDE

The computational formula for $CV(h)$ is given by

$$\frac{1}{N^2 h}\sum_{i=1}^{N}\sum_{j=1}^{N}\int K(y)K\left(\frac{X_i - X_j}{h} - y\right)dy - \frac{2}{N(N-1)h}\sum_{j=1}^{N}\sum_{i\neq j}K\left(\frac{X_i - x_j}{h}\right)$$

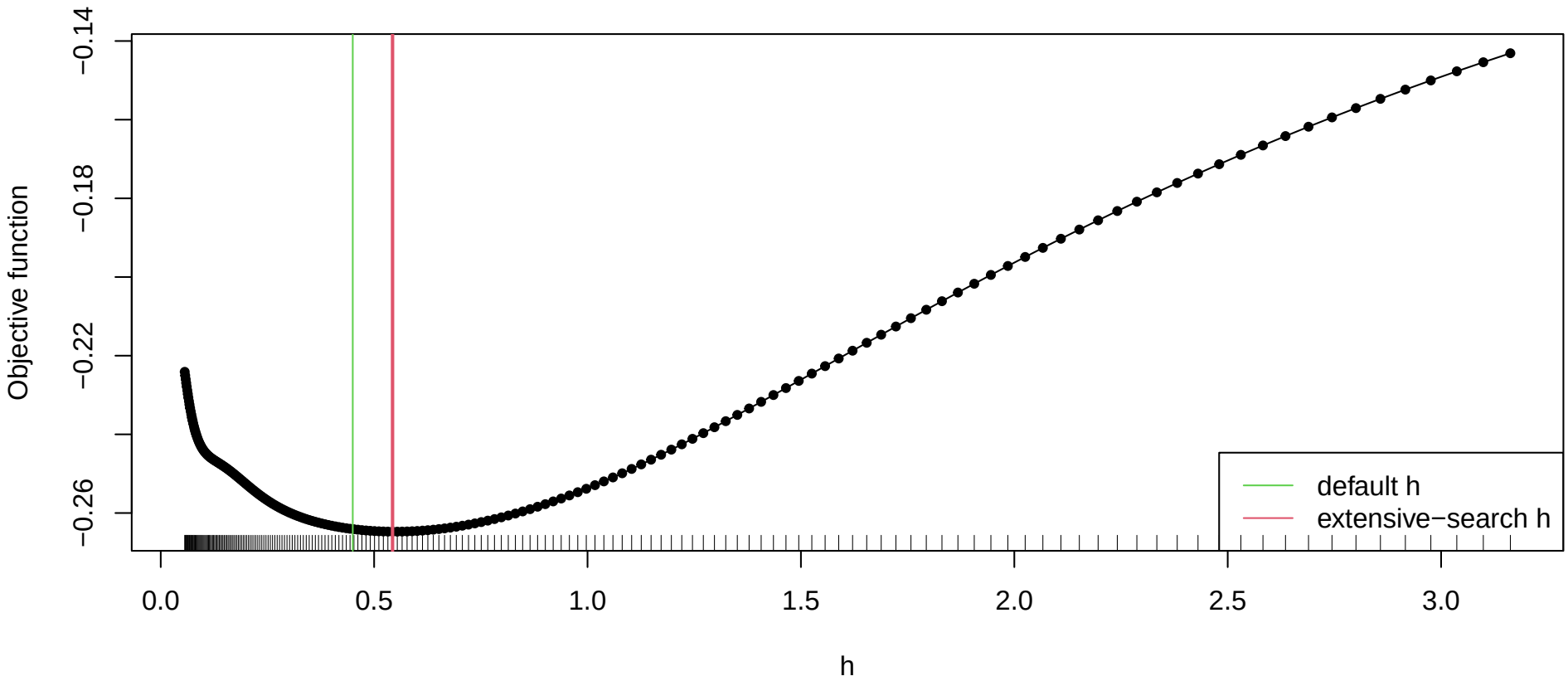
The optimal bandwidth is then given by

$$h_{opt} = \arg\min_{h>0} CV(h)$$

- Numerical optimisation is required
- The roughness of the objective function depends on n and f $\Rightarrow$ might have several local minima

$\Rightarrow$ Always check the solution by plotting $CV(h)$ for a range of h

Bandwidth Selection for KDE



CV for PCA

(1) linear combinations with maximal variance (Pearson, 1901)

$$\underset{v^\top v=1}{\arg\max}\, v^\top \widehat{\Sigma} v$$

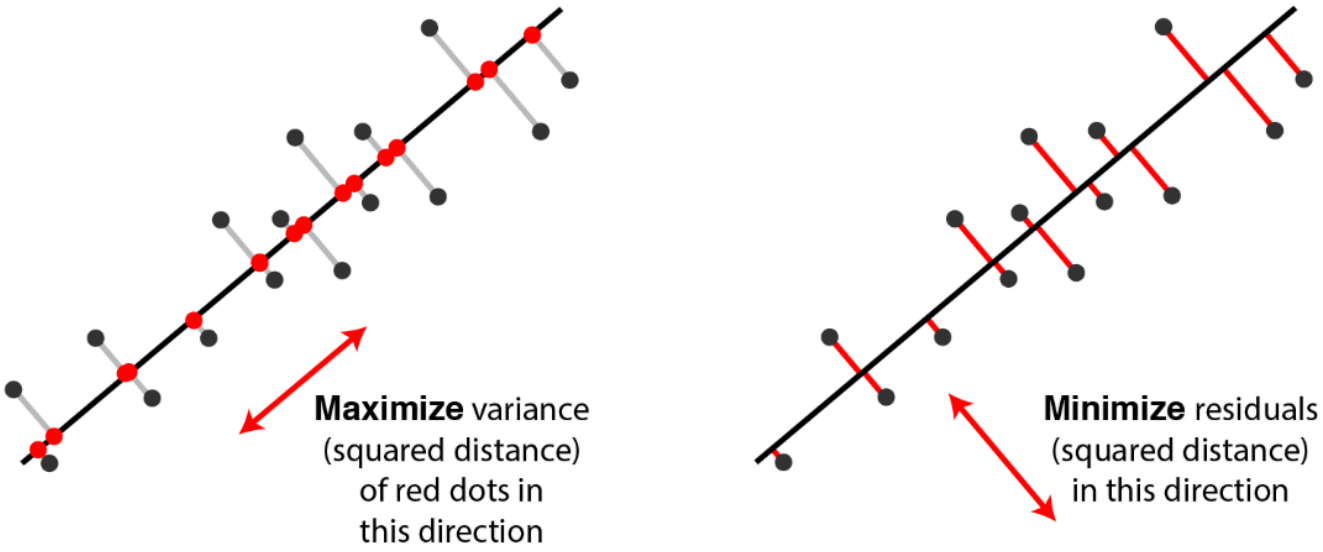
(2) minimum least square error projection into lower dimension (Hotelling, 1933)

$$\underset{V^\top V=I_r}{\arg\min} \sum_{i=1}^{n} \|x_i - \mathbf{V}\mathbf{V}^\top x_i\|_2^2$$

(3) best low-rank matrix approximation (Eckart & Young, 1936)

$$\underset{\text{rank}(\mathbf{L})=r}{\arg\min} \|\mathbf{X} - \mathbf{L}\|_2^2$$

(1)-(2) Optimisation problems

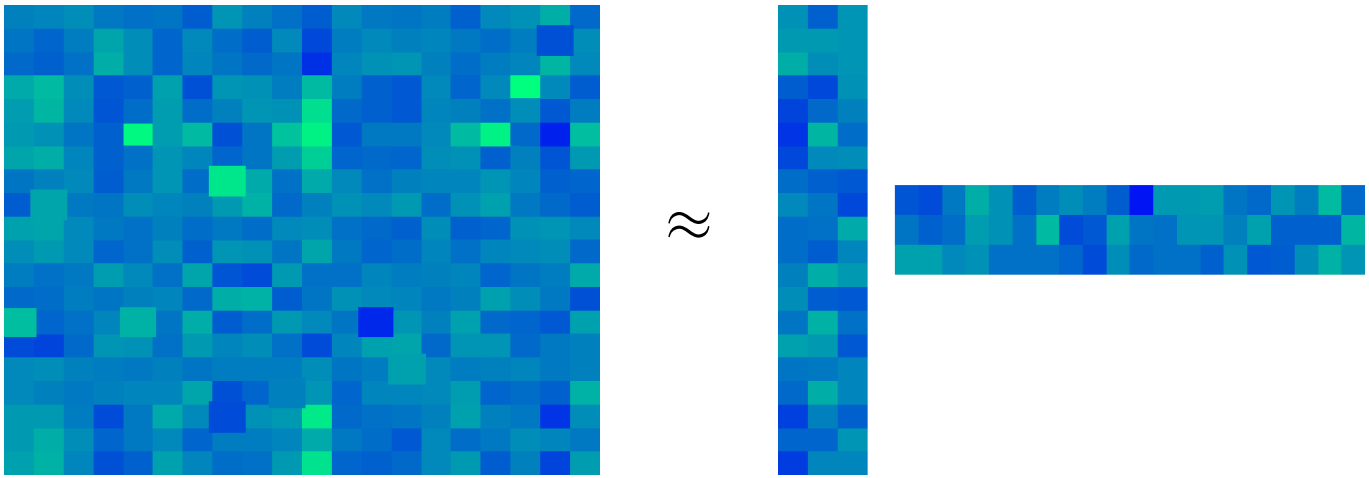


Two equivalent views of principal component analysis.

Source: this blog

(3) Low-rank Matrix Approximation

Visualization for $r = 3$:



$$\mathbf{X} \approx \mathbf{L} = \mathbf{A}\mathbf{B}^\top = \sum_{i=1}^{r} \mathbf{a}_i \mathbf{v}_i^\top$$

The tall and skinny matrix A and the short and fat matrix B are obtained by truncating the **SVD** decomposition: $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ to keep the r top singular values of X ($A = U_k D_k^{1/2}$ and $B = V_k D_k^{1/2}$)

CV for PCA

In all formulations, there is a hyperparameter $r(< p)$!

Let's focus on the third formulation of PCA

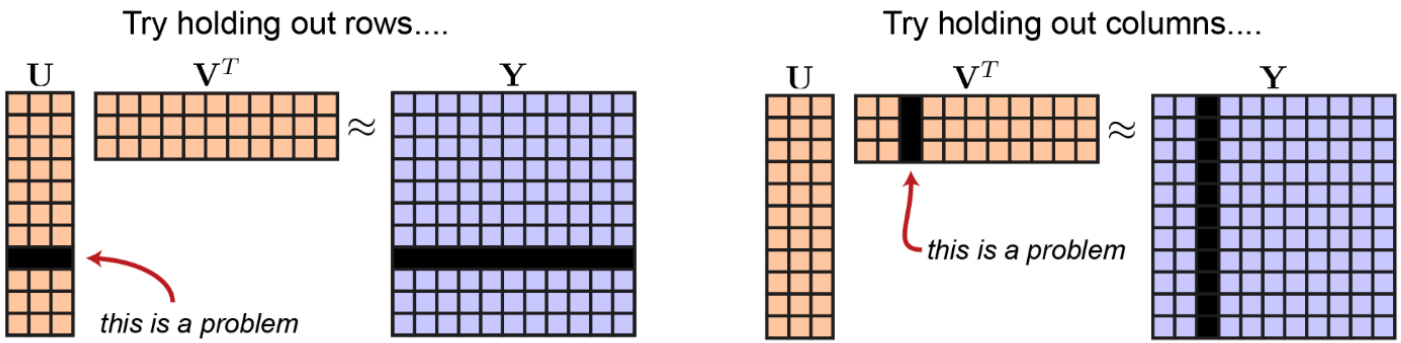
$$\underset{\text{rank}(\mathbf{L})=r}{\arg\min} \|\mathbf{X} - \mathbf{L}\|_2^2$$

How to choose the rank r? Some proposed the following K-fold CV scheme:

- split data into K folds $J_1, \dots, J_K$
- **for** $k = 1, \dots, K$
  - solve $\widehat{\mathbf{L}} = \underset{\text{rank}(\mathbf{L})=r}{\arg\min} \|\mathbf{X}[J_k^c,] - \mathbf{L}\|_2^2$
  - calculate $Err_k(r) = \sum_{n \in J_k} \|x_n - P_{\widehat{L}} x_n\|_2^2$
- **end for**
- choose $\hat{r} = \underset{r}{\arg\min} \sum_{k=1}^{K} |J_k|^{-1} Err_k(r)$

But this is wrong! (as $r \nearrow$ we have $\|x_j - P_{\widehat{L}} x_j\| \searrow$, so r is overestimated)

CV for PCA



Not so great ideas for cross-validating matrix factorization.

Source: this blog

Problems with holding out a whole column (or row) of the data matrix are discussed in more detail by Bro et al. (2008) and Owen & Perry (2009)

There are smarter holdout patterns

- Wold hold-out: requires an SVD decomposition with missing data as entries are held-out at random
- Gabriel hold-out: transforms the unsupervised learning problem into a supervised one by holding-out a block of the data matrix

Intermezzo: Linear Prediction for Gaussian Vectors

For $X \sim \mathcal{N}(\mu, \Sigma)$ split into

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}, \qquad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \qquad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{pmatrix},$$

the conditional expectation of X_1 given X_2 is given by

$$\mathbb{E}_{\mu,\Sigma}\left[X_1 \middle| X_2 = \mathbf{x}_2\right] = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}_2 - \mu_2)$$

Intermezzo: Linear Prediction for Gaussian Vectors

For $X \sim \mathcal{N}(\mu, \Sigma)$ split into

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}, \qquad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \qquad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{pmatrix},$$

the conditional expectation of X_1 given X_2 is given by

$$\mathbb{E}_{\mu,\Sigma}\left[X_1 \middle| X_2 = \mathbf{x}_2\right] = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}_2 - \mu_2)$$

Assume we have a sample $X_1, \ldots, X_N$ from which we obtain estimators $\hat{\mu}$ and $\widehat{\Sigma}$, and a new incomplete observation $X^\star = (X_1^\star, X_2^\star)^\top$, where only $X_2^\star$ is observed. We simply predict the missing part by

$$\widehat{X}_1^\star = \hat{\mu}_1 + \widehat{\Sigma}_{12}\widehat{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \mu_2)$$

Even without Gaussianity, this is the best linear unbiased predictor (BLUP)

- The quality of BLUP depends on that of the estimators $\hat{\mu}$ and $\widehat{\Sigma}$

CV for PCA Repaired

Assume that data $\mathbf{x}_n \in \mathbb{R}^p$ are i.i.d. realizations of $X \sim \mathcal{N}(\mu, \Sigma)$.

- split data into K folds $J_1, \dots, J_K$
- **for** $k = 1, \dots, K$
  - estimate μ and Σ empirically using all but the k-th fold J_k, but truncate Σ to be rank-r
  - **for** $n \in J_k$
    - split $\mathbf{x}_n$ into a "missing" part $\mathbf{x}^{miss}$ that will be used for validation and an "observed" part $\mathbf{x}^{obs}$
    - predict $\mathbf{x}_n^{miss}$ from $\mathbf{x}_n^{obs}$ as discussed on the previous slide
  - **end for**
  - calculate $Err_k(r) = \sum_{n \in J_k} \|(\mathbf{x}_n^{obs}, \mathbf{x}_n^{miss})^\top - (\mathbf{x}_n^{obs}, \hat{\mathbf{x}}_n^{miss})^\top\|_2^2$
- **end for**
- choose $\hat{r} = \arg\min_r \sum_{k=1}^K |J_k|^{-1} Err_k(r)$

Is there a bias-variance trade-off now?

Assignment 4 [5 %]

Go to Assignment 4 for details.