

Problem set 1: Approximation, estimation, and shrinkage

Foundations of Machine Learning, Hilary term 2027

Maximilian Kasy

What this problem set is about

In lectures, we derived two sets of theoretical results that this problem set lets you see “in action”:

- From **PAC learning theory**: the risk of a learned predictor can be decomposed into an *approximation error* (how well the best predictor in the class can do) and an *estimation error* (how much we lose because we only have a finite sample). Richer classes reduce the first and increase the second. In Problem 1, you will compute both, using a real data set.
- From our discussion of **shrinkage in the normal means model**: the James-Stein estimator has lower risk than the maximum likelihood estimator, no matter what the true parameter is, and Stein’s unbiased risk estimate (SURE) gives an unbiased estimate of this risk. In Problem 2, you will verify these results by simulation.

In both problems, the goal is not just to produce the figures, but to understand why they look the way they do. That understanding is what the exam will test.

How this problem set works

1. **Software.** Problem 1 has to be solved in Python, using scikit-learn. Problem 2 can be solved in Python or R. If you are new to scikit-learn, our lecture notebook *Supervised learning using Python*, Chapter 5 of the Python Data Science Handbook, and the scikit-learn documentation are good places to start.
2. **Your report.** Your code should run from start to end in one execution, producing all the output. Code, output, and discussion should be integrated in a single report (a Jupyter notebook, or R-Markdown/Quarto), which you submit as a pdf. Please label all figures and tables clearly, so that someone who has not read your code can understand them. For reference: my solution runs in about five minutes on a laptop.
3. **Predictions.** Throughout the problem set, you will find blue boxes labelled *Predict* asking you to **predict** what you will find, *before* you run the corresponding code. Please take these seriously: they are the most useful part of the exercise. Write down your prediction in your report, in a few sentences, together with a brief reason (“because the class gets larger, ...”). Keep your prediction in the report even if it turns out to be wrong; your predictions are not assessed on whether they are correct. At the end of each problem, a short section asks you to look back at your predictions: Were they

right? If not, what did you overlook? This is the discussion part of your report; there is no separate discussion to write. Predictions that turned out wrong are usually where you learn the most, and they are good preparation for the exam.

4. **Checkpoints.** The grey boxes list numbered *checkpoints*, [C1.1], [C1.2], and so on: specific numbers that your code should produce. For many of them, the theory from class tells you what the value should be; in that case, the box says so. If your number differs from the theoretical prediction by more than Monte Carlo error, say so in your report and try to explain why. At the end of your report, print all checkpoints in one block, one per line, in exactly this format:

```
CHECKPOINT C1.1 = 0.4096
```

for instance using

```
print(f"CHECKPOINT C1.1 = {value:.4f}")          # Python
cat(sprintf("CHECKPOINT C1.1 = %.4f\n", value))    # R
```

5. **Preparing for the exam.** Each problem ends with a green box of *questions for the exam*. You do not need to answer these in your report. The exam will contain questions similar to these, about figures like the ones you produce here, and about short pieces of code from my solution. If you have done the problem set yourself and thought about your predictions, you should find them straightforward.

Problem 1: Approximation error and estimation error (Python)

The idea

In practice, we never know the risk of a predictor, because we do not know the population distribution of the data. To make the theory from class visible, we will use a trick: we take a real data set, and *pretend that it is the entire population*. We then draw small random samples from this “population”, fit predictors on the samples, and evaluate them on the rest of the population. Because we know the population, we can compute exactly the quantities that appear in the theory, such as the approximation error.

Data and setup

We use the California Housing data, which contains $N = 20,640$ districts. The outcome y is the median house value in a district (in units of \$100,000), and there are 8 covariates (median income, house age, average number of rooms, location, and so on). Load the data, and replace each covariate by its rank in the population, scaled to lie between 0 and 1:

```
import numpy as np
from scipy.stats import rankdata
from sklearn.datasets import fetch_california_housing

data = fetch_california_housing()
X, y = data.data, data.target          # 8 covariates; median house value
N = len(y)                             # N = 20,640 districts
U = (rankdata(X, axis=0) - 0.5) / N     # rank-transformed covariates in (0,1)
```

(The rank transformation tames some extreme values in the covariates, which would otherwise cause trouble for the splines below.)

We call the full data set the **population** $\mathcal{D}$. We use squared error loss, so that the risk of a predictor h is its mean squared error in the population,

$$L(h, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^N (y_i - h(x_i))^2.$$

Hypothesis classes. We consider a sequence of increasingly flexible classes of predictors. For $K \in \{2, 3, 5, 9, 17\}$, the class $\mathcal{H}_K$ contains all additive cubic splines in the 8 covariates, with K equally spaced knots for each covariate. In practice, this means: $\mathcal{H}_K$ is the set of linear predictors $h(x) = a + F_K(x) \cdot \beta$, in the spline features

```
from sklearn.preprocessing import SplineTransformer
F = SplineTransformer(n_knots=K, degree=3).fit_transform(U)
```

Larger K means more features ($8 \cdot (K + 2)$ of them), and hence a richer class.

(a) Approximation error

The approximation error is the risk of the *best* predictor in the class,

$$\epsilon_{app}(K) = \min_{h \in \mathcal{H}_K} L(h, \mathcal{D}).$$

Since we know the population, we can compute it: the best linear predictor in the population is the population least squares fit.

PREDICT P1.1

before you run the code

How will $\epsilon_{app}(K)$ change as K increases from 2 to 17? Will it go up, go down, or could it do either?

For each K , fit `LinearRegression()` of y on the features F_K using the *entire population*, and compute the resulting mean squared error. This is $\epsilon_{app}(K)$.

WHAT TO REPORT

[C1.1]: $\epsilon_{app}(2)$. [C1.2]: $\epsilon_{app}(17)$.

These two numbers involve no randomness, so they should agree with my solution up to rounding.

(b) Training error, cross-validation, and true risk

Now we learn from a sample, as we would in practice. For each K , and for each of the two sample sizes $n = 300$ and $n = 1000$, repeat the following steps $R = 100$ times:

1. Draw a random sample of n districts from the population, without replacement.
2. Fit `LinearRegression()` on the sample. This gives a learned predictor $\hat{h} \in \mathcal{H}_K$.
3. Compute three numbers for $\hat{h}$:
 - the **training MSE**: the mean squared error of $\hat{h}$ on the sample that it was fit on;
 - the **10-fold cross-validation MSE** on the sample, which you can compute as
`-cross_val_score(LinearRegression(), F[idx], y[idx], cv=10, scoring="neg_mean_squared_error").mean()`
 - the **true risk** $L(\hat{h}, \mathcal{D})$, which we approximate by the mean squared error of $\hat{h}$ on the $N - n$ districts that are *not* in the sample.

Then average each of the three numbers over the R repetitions. The true risk is something we can only compute because we know the population; the training MSE and the CV MSE are what you could compute in a real application.

PREDICT P1.2

before you run the code

- Which of the three curves (training MSE, CV MSE, true risk) will lie *below* the approximation error $\epsilon_{app}(K)$, at least for some K ?
- Which of them will be close to each other?
- For $n = 300$, roughly where do you expect the true risk to be smallest? Will the best K be larger or smaller for $n = 1000$?

WHAT TO REPORT

Figure 1.1: Two panels, one for $n = 300$ and one for $n = 1000$. Horizontal axis: K (log scale). Vertical axis: mean squared error, from 0 to 0.8. Four lines in each panel: average training MSE, average CV MSE, average true risk, and $\epsilon_{app}(K)$.

[C1.3]: average training MSE, for $n = 300$ and $K = 9$.

[C1.4]: average CV MSE, for $n = 300$ and $K = 9$.

[C1.5]: average true risk, for $n = 300$ and $K = 9$.

[C1.6]: average true risk, for $n = 1000$ and $K = 9$.

(c) Regularization instead of a smaller class

Instead of choosing a small class, we can also use a rich class and penalize complex predictors. Fix the richest class, $K = 17$, and the small sample size, $n = 300$. Repeat part (b), replacing `LinearRegression()` by ridge regression, `Ridge(alpha=a)`, for each penalty value a in `np.logspace(-3, 2, 11)`.

PREDICT P1.3

before you run the code

How will the training MSE and the true risk change as the penalty α goes from very small (0.001) to very large (100)? Which end corresponds to overfitting, and which to underfitting?

WHAT TO REPORT

Figure 1.2: Horizontal axis: α (log scale). Vertical axis: mean squared error, from 0 to 0.8. Four lines: average training MSE, average CV MSE, average true risk, and $\epsilon_{app}(17)$.

[C1.7]: the smallest average true risk across the values of α .

(d) Looking back at your predictions

Return to your predictions P1.1–P1.3. For each one, say in a sentence or two whether it was right, and if not, what you overlooked. Where you can, use the terms from class: approximation error, estimation error, and the bias-complexity tradeoff.

QUESTIONS FOR THE EXAM

1. In Figure 1.1, the true risk equals $\epsilon_{app} + \epsilon_{est}$. Read off the approximation error and the (average) estimation error for $n = 300$ and $K = 17$.
2. Why does the approximation error decrease as K increases? Why does the true risk not decrease in the same way?
3. Why is the training MSE below the true risk? Why can it even be below the approximation error?
4. Why is the estimation error smaller for $n = 1000$ than for $n = 300$? Relate your answer to the sample size needed for uniform convergence in a finite hypothesis class, $n = \lceil \log(2|\mathcal{H}|/\delta)/(2\epsilon^2) \rceil$.

5. In a real application, we cannot compute the true risk. Which curve in Figure 1.1 would you use to choose K , and why does it work?
6. In Figure 1.2, which values of α lead to overfitting, and which to underfitting? How can you tell from the figure?

Problem 2: Shrinkage in the normal means model (Python or R)

The idea

In the normal means model, we observe $\mathbf{X} \sim N(\boldsymbol{\theta}, I_k)$ and want to estimate the k -vector $\boldsymbol{\theta}$. The obvious estimator is the MLE, $\hat{\boldsymbol{\theta}} = \mathbf{X}$. In class, we discussed several estimators that shrink $\mathbf{X}$ towards zero (or towards the average of its components), and we showed that the James-Stein estimator has lower risk than the MLE for every value of $\boldsymbol{\theta}$, as long as $k \geq 3$. In this problem, you will compute the risk functions of these estimators by simulation, and compare them.

Throughout, we use the loss $\frac{1}{k} \|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\|^2$, that is, the squared error averaged across the k components. (In the slides, we used the sum instead of the average; dividing by k makes it easier to compare different dimensions k .) With this normalization, the risk of the MLE equals 1.

Estimators

As in the slides, let $\bar{X}^2 = \frac{1}{k} \sum_i X_i^2$, $\bar{X} = \frac{1}{k} \sum_i X_i$, $s_X^2 = \frac{1}{k} \sum_i (X_i - \bar{X})^2$, and $\mathbf{1} = (1, \dots, 1)$. We consider five estimators:

1. The **MLE**, $\hat{\boldsymbol{\theta}} = \mathbf{X}$.
2. The **empirical Bayes** estimator, $\hat{\boldsymbol{\theta}} = \left(1 - \frac{1}{\bar{X}^2}\right) \cdot \mathbf{X}$. We derived this estimator in three different ways in class: as a feasible version of the best linear predictor, as the parametric empirical Bayes estimator, and as the shrinkage factor that minimizes SURE.
3. The **James-Stein** estimator, $\hat{\boldsymbol{\theta}} = \left(1 - \frac{(k-2)/k}{\bar{X}^2}\right) \cdot \mathbf{X}$, which differs from the empirical Bayes estimator only by a degrees of freedom correction.
4. The **positive part James-Stein** estimator, $\hat{\boldsymbol{\theta}} = \max\left(0, 1 - \frac{(k-2)/k}{\bar{X}^2}\right) \cdot \mathbf{X}$.
5. The **grand mean** estimator, which shrinks towards the average of the components instead of towards zero, $\hat{\boldsymbol{\theta}} = \bar{X} \cdot \mathbf{1} + \left(1 - \frac{(k-3)/k}{s_X^2}\right) \cdot (\mathbf{X} - \bar{X} \cdot \mathbf{1})$.

Simulation design

For each dimension $k \in \{3, 5, 10\}$, do the following:

1. Let $\boldsymbol{\theta}_1 = (1, 0, \dots, 0)$, and consider the parameter values $\boldsymbol{\theta} = r \cdot \boldsymbol{\theta}_1$, for $r \in \{0, 0.25, 0.5, \dots, 6\}$. Note that $r = \|\boldsymbol{\theta}\|$ measures how far $\boldsymbol{\theta}$ is from zero.
2. Draw $R = 10,000$ vectors $\mathbf{Z}_j \sim N(0, I_k)$, and use the *same* draws for all values of r , setting $\mathbf{X}_j = \boldsymbol{\theta} + \mathbf{Z}_j$. (Using the same draws for all r is called “common random numbers”; it makes the risk curves much smoother.)
3. For each r and each estimator, compute the loss $\frac{1}{k} \|\hat{\boldsymbol{\theta}}(\mathbf{X}_j) - \boldsymbol{\theta}\|^2$ for every draw j , and average across the R draws. This average is our estimate of the risk.

(a) Risk functions

PREDICT P1.4

before you run the code

- For $k = 3$, two of the five estimators are exactly identical. Which ones? (Look at the formulas.)
- Which estimator do you expect to have the lowest risk at $r = 0$? What happens to all the risk functions as r becomes large?
- Will any of the estimators have higher risk than the MLE, for some k and r ?

WHAT TO REPORT

Figure 1.3: Three panels, one for each k . Horizontal axis: r . Vertical axis: risk. One line for each of the five estimators, with a legend.

[C1.8]: risk of James-Stein, for $k = 10$ and $r = 0$. Theory: $2/k$.

[C1.9]: risk of empirical Bayes, for $k = 10$ and $r = 0$. Theory: $2/(k - 2)$.

[C1.10]: risk of the grand mean estimator, for $k = 10$ and $r = 0$. Theory: $3/k$.

[C1.11]: risk of the MLE, for $k = 5$, averaged over all values of r . Theory: 1.

[C1.12]: risk of James-Stein, for $k = 3$ and $r = 0$. Theory: $2/3$. (This one is noisy; do not worry if you are off by a few hundredths.)

(b) Stein's unbiased risk estimate

Recall from class that Stein's unbiased risk estimate (SURE) lets us estimate the risk of an estimator from a single draw of $\mathbf{X}$, without knowing $\boldsymbol{\theta}$. For the James-Stein estimator, and with our normalization of the loss, SURE equals

$$\hat{R} = 1 - \frac{(k - 2)^2}{k \cdot \sum_i X_i^2}.$$

For each r , compute $\hat{R}$ for each draw $\mathbf{X}_j$, and average across the R draws.

PREDICT P1.5

before you run the code

Will the average of $\hat{R}$ across draws lie above, below, or on top of the simulated risk of James-Stein? Will this depend on r ?

WHAT TO REPORT

Figure 1.4: Two panels, for $k = 5$ and $k = 10$. Horizontal axis: r . Two lines: the simulated risk of James-Stein, and the average of $\hat{R}$.

[C1.13]: for $k = 10$, the largest absolute difference (across the values of r) between the average of $\hat{R}$ and the simulated risk of James-Stein. Theory: 0, up to Monte Carlo error.

(c) Looking back at your predictions

Return to your predictions P1.4 and P1.5. For each one, say in a sentence or two whether it was right, and if not, what you overlooked. Where you can, relate your answer to the

theoretical results on James-Stein shrinkage from class.

QUESTIONS FOR THE EXAM

1. Why is the risk of the MLE equal to 1 for every value of r ?
2. In class, we derived $R(\hat{\boldsymbol{\theta}}^{JS}, \boldsymbol{\theta}) = k - E[(k-2)^2 / \sum_i X_i^2]$ (for the loss summed over components), and that $E[1 / \sum_i X_i^2] = 1/(k-2)$ when $\boldsymbol{\theta} = 0$. Use these to show that the risk in Figure 1.3, at $r = 0$, equals $2/k$.
3. Why do the risk functions of all shrinkage estimators approach the risk of the MLE as r becomes large?
4. Which result from class does Figure 1.3 confirm for James-Stein? Which estimator has the lowest risk everywhere in your figure, and is this consistent with what we said in class?
5. Why is the grand mean estimator identical to the MLE when $k = 3$? Why would James-Stein be identical to the MLE when $k = 2$?
6. The empirical Bayes estimator and James-Stein differ only by the degrees of freedom correction. In your figure, which one has lower risk? Does the difference matter more for $k = 3$ or for $k = 10$?
7. What does Figure 1.4 show? Why is SURE useful in practice?