Analytics

Peter Hoffman

February 21, 2026

Contents

1 General modeling

1.1 Maximum likelihood estimation

Maximum likelihood estimation involves first seeing data $x \in \mathcal{X}$ which we assume is generated by some model in a class of models $P_\theta \in \{P_\theta \mid \theta \in \Theta\}$. Then calculate likelihood, which is simply the joint pdf: $\mathcal{L}(x, \theta) = \prod_{i=1}^{n} P_\theta(x_i)$. Think of this likelihood as the probability we saw data conditioned on a theta. Because logarithms are strictly increasing functions, maximizing the likelihood is equivalent to maximizing the log-likelihood. However, maximizing the log likelihood is often easier because since most common probability distributions—notably the exponential family—are only logarithmically concave. Therefore, we take maximum of the log likelihood to find the $\theta \in \Theta$ that maximizes the likelihood that we received our data.

1.2 Common Types of Bias in Data Analytics

Biases are systematic errors in data collection, analysis, or interpretation. Common biases include:

(1) **Selection Bias:** Sampling procedure that systematically favors certain groups.

Example: Conducting an online survey that only includes internet users.

(2) **Survivorship Bias:** Focusing only on "survivors" and ignoring those that failed.

Example: Studying successful companies without accounting for failed startups. Additionally, using data that is censored in some way (e.g. data on loan repayments is only available from loans that were funded at the outset)

(3) **Population Bias:** Sample not representative of target population.

Example: Using data only from urban hospitals to generalize about national health outcomes. Note that Population Bias can be *created* by *both* Selection bias and Survivorship bias, which both cause a sample to not be representative of a population.

NOTE: when population bias exists, we must refine the thing we seek to predict/understand. For example, we cannot predict the probability that an applicant will default on a loan by evaluating default rates of past loans, since we only have data from loans that were approved. However, we *can* predict whether a successful loan applicant will default.

(4) **Confirmation Bias:** Interpreting data to support pre-existing beliefs.

Example: Highlighting correlations that fit a hypothesis while ignoring contradictions.

(5) **Measurement Bias:** Systematic error in how data is collected or recorded.

Example: Faulty sensors consistently under-reporting temperature.

(6) **Framing bias:** occurs when people's decisions are influenced by how information is presented, rather than by the information itself. *Example:* People prefer 80% fat free over 20% fat.

1.3 Occam's razor

The simplest model that fits sample data will probably generalize to the rest of the population the best

1.4 Common methods in predictive modeling

Predictive modeling involves variables → model → predictive output. Common methods include

- logistic regression
- linear regression
- classification and regression trees
- random forests
- gradient boosted machines
- support vector machines
- deep neural networks

1.5 Standardizing a dataset

Suppose we have a dataset $\{x_i\}$ with empirical mean μ and empirical variance σ^2. We use the following linear map to *standardize* the dataset.

$$x \leftarrow \frac{x - \mu}{\sigma}$$

Note: we use σ because standard deviation has the same units as the original data, hence it is more interpretable.

1.6 Overfitting

Let $f : x \mapsto y$ be a predictive function and let $\ell : (y, \hat{y}) \mapsto \mathbb{R}$ be a loss function. Then the *empirical risk* and *true risk* are defined as

$$R_{\text{empirical}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i) \quad \text{(empirical risk)}, \qquad R_{\text{true}}(f) = \mathbb{E}_{(x,y)\sim\mathcal{P}}[\ell(f(x), y)] \quad \text{(true risk)}.$$

$$\boxed{\text{Overfitting occurs when:} \quad R_{\text{empirical}}(f) \ll R_{\text{true}}(f)}$$

- Model fits the training data well but generalizes to the population poorly.
- Overfitting is caused by excessive model complexity relative to data (e.g. too many parameters)

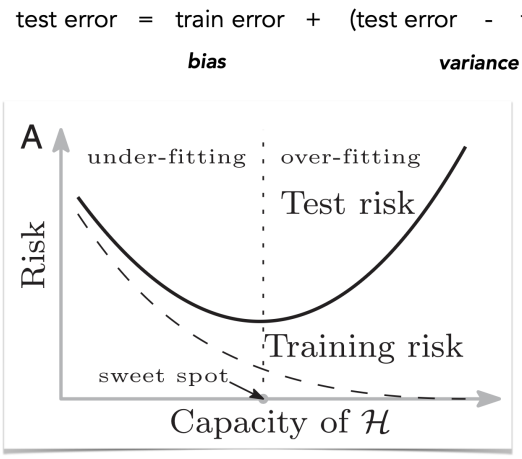


Figure 1: Overfitting from observing risk

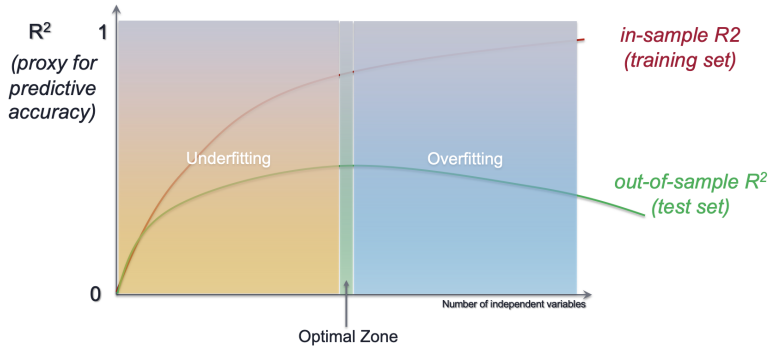


Figure 2: Overfitting from observing R^2

One hypothesis to describe overfitting:

$$\text{learned model} = \underbrace{\text{“simple”}}_{\text{predictive component}} + \underbrace{\text{“spiky”}}_{\text{overfitting component}}$$

1.7 Generalization error: Vapnik-Chervonenkis theory

$$\text{Generalization error} = \text{population error - training error}$$

Vapnik-Chervonenkis theorem: IF the size of training set $\gg$ number of functions in our function class $\implies$ the training error matches population error with high probability.

This means that if all data is IID and

1. the hypothesis class is small enough
2. there's a function in your hypothesis class that works well
3. and if we have an optimization algorithm to find that function

then training error matches population error with high probability.

Intuition:

- Our hypothesis class contains a bunch of functions.

- Maybe we just got too lucky and found a function that "happens to work on the training set" but is totally bad on the population
- But how often can we get that lucky? Chance of a false positive is:
  - higher if we have more candidate functions (more change of getting "lucky" on training data only)
  - lower if we have more data (each additional datapoint rules out more candidate functions)

1.8 The bias variance tradeoff

Reducing model bias (e.g. fitting the data better with a more complex model) often increases variance (e.g. our the model will be more sensitivity to training data across different datasets due to overfitting)

- High bias leads to underfitting (missing patterns), while high variance leads to overfitting.
- Complex models often have low bias but high variance.
- Simpler models often have high bias but low variance.

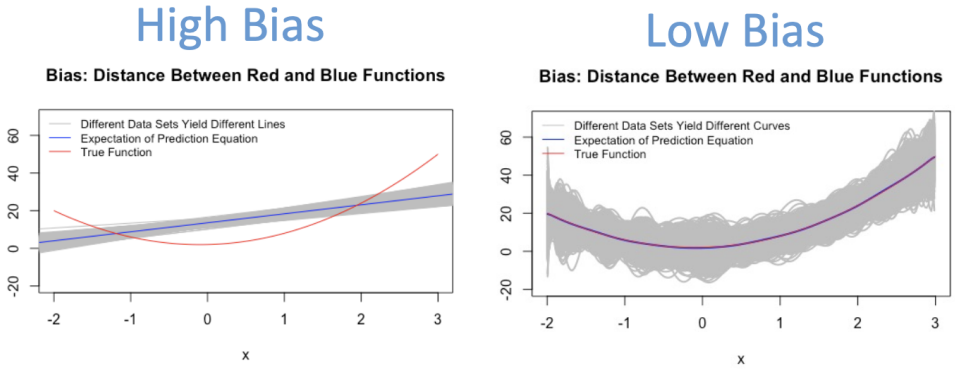


Figure 3: High bias occurs when the model does not capture the trend in the data

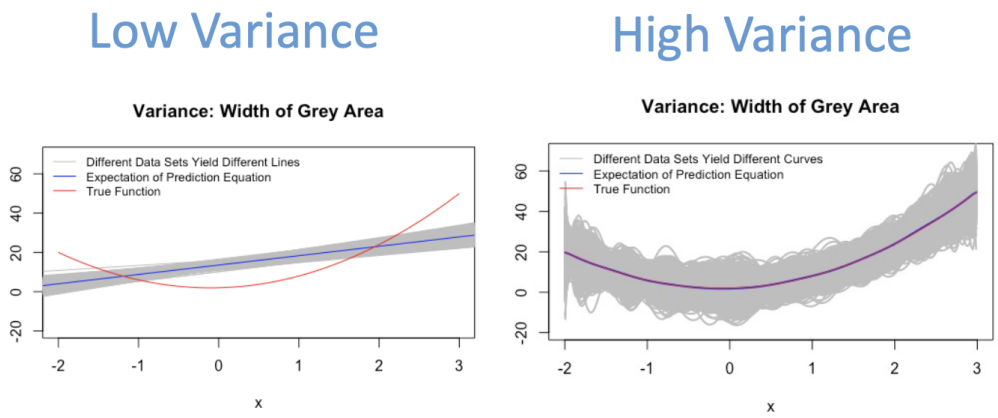


Figure 4: High variance occurs when the model is overly sensitive to changes in the training data

1.9 Bias variance tradeoff

How far off do we expect y_i to be from our prediction $\hat{f}(x_i)$? By expanding the prediction error

below and using that $\mathbb{E}[y - f(x)] = 0$ since the true model is assumed to be $y = f(x) + \epsilon$, then

$$\underbrace{\mathbb{E}\Big(y_i - \hat{f}(x_i)\Big)^2}_{\text{Prediction Error}} = \underbrace{\mathbb{E}(y_i - f(x_i))^2}_{\text{Irreducible Error}} + \underbrace{\mathbb{E}\Big(\hat{f}(x_i) - f(x_i)\Big)^2}_{\text{Reducible Error}}.$$

Irreducible error is:

$$\text{Var}(y_i \mid x_i) = \text{Var}(\varepsilon).$$

Reducible error is

$$\underbrace{\mathbb{E}\Big(\hat{f}(x_i) - f(x_i)\Big)^2}_{\text{Reducible Error}} = \underbrace{\big(\mathbb{E}[\hat{f}(x_i)] - f(x_i)\big)^2}_{\text{Squared Estimator Bias}} + \underbrace{\mathbb{E}\Big(\hat{f}(x_i) - \mathbb{E}[\hat{f}(x_i)]\Big)^2}_{\text{Estimator Variance}}.$$

Estimator Bias: how well we fit the relationship in the data.

Estimator Variance. How much does the predicted value $\hat{f}(x_i)$ vary from one dataset to the next.

1.10 How to avoid overfitting

- Simpler models with few parameters
- Early stopping during training
- Dropout (for Neural Networks): Randomly drop neurons during training to prevent over-reliance on specific nodes.

2 Covariance and correlation

2.1 Sum of square residuals

The unexplained variability of some model's outputted predictions $\{\hat{y}_i\}_{i\in[n]}$ is the SSR

$$\text{SSR}(\texttt{model}) = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Similarly,

$$\text{SST}(\bar{y}) = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

2.2 Covariance and correlation in linear regresssion

For random variables X, Y with finite variance, the *correlation*

$$\rho_{X,Y} = \frac{\text{Cov}(X,Y)}{\sigma_X\,\sigma_Y} \qquad \in [-1,1]$$

If $\{(y_i,\ x_i)\}_{i=1}^{n}$ is a dataset with n observations and means $\bar{y},\ \bar{x}$, then the *empirical correlation* $r_{x,y}$ is

$$r_{x,y} = \frac{S_{x,y}}{\sigma_x\sigma_y} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\,\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \qquad \in [-1,1].$$

Correlation measures the *strength and direction of a linear relationship* between two real-valued variables. When $r \approx 1$, there is a strong positive linear association; when $r \approx 0$ there is no linear association; and when $r \approx -1$: there is a negative linear association.

Properties

- Dimensionless; invariant to affine rescaling $(ax + b)$ with $a > 0$.
- *Linear* only: nonlinear relationships can have $r \approx 0$.
- Sensitive to outliers; inspect scatterplots.

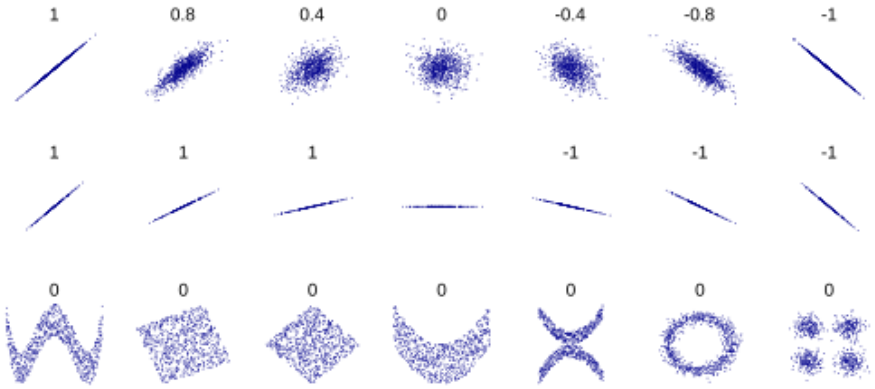


Figure 5: Correlation of data

2.3 Interpretation of R^2

R^2 is the proportion of the variation that is explained by the the regression model relative to the baseline model:

$$R^2 = \frac{SST - SSR}{SST}$$

In simple regression,

$$R^2 = (\text{cor}(y, x))^2$$

In multiple linear regression,

$$R^2 = (\text{cor}(y, \hat{y}))^2$$

Can calculate in-sample and out-of-sample, but note the out-of-sample SST uses the baseline and model calculated on the training dataset.

3 Linear regression

Assume the following *true* model:

$$Y_i \;=\; \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} + \varepsilon_i.$$

where the errors satisfy $\varepsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$ and $\beta_0,\ \beta_1, \ldots, \beta_k$.

3.1 Our proposed linear model

Our proposed linear model is

$$\hat{y}_i = \beta_0 + \beta_1 x_{i,1} + \ldots + \beta_K x_{i,K}$$

We can interpret these slopes as the rate of change of the output as a function of a given input when all other input coordinates are held constant.

Implicit assumptions:

1. **Linearity.** The true model is indeed linear between y_i and x_i.
2. **Homoskedasticity.** The dispersion of $\text{error}_i = y_i - \hat{y}_i$ is not systematically smaller or larger for large x_i values compared to small x_i values.
3. **Normality.** The residuals are approximately normal.

Note: what matters for linear regression is the linearity of the coefficients, not the input variables. E.g.

$$\log(sales) = b_0 + b_1(\text{price})^2 + b_2(\text{ads}) \cdot (\text{price})$$

is a linear model!

3.2 Ordinary least squares optimization and connections to likelihood

In least squares, we solve the following to find the coefficients b_i:

$$\min_{\substack{b_0,\ldots,b_k \\ e_1,\ldots,e_n}} \quad \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{1}$$

$$\text{s.t.} \qquad \hat{y}_i = b_0 + \sum_{j=1}^{k} b_j x_{i,j}, \qquad\qquad i = 1, \ldots, n \tag{2}$$

It's worth asking, "where does this loss function come from?" Answer: this is the loss function from maximizing the likelihood of our linear-gaussian model. Given a model

$$y_i \;=\; b_0 + b_1 x_i + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma^2), \qquad i = 1, \ldots, n.$$

and observed data

$$\mathcal{D} = \{(y_1, x_1), (y_2, x_2), \ldots, (y_n, x_n)\}.$$

Then the likelihood of the data given parameters (b_0, b_1, σ).

$$L(b_0, b_1, \sigma \mid \mathcal{D}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\,\sigma}\ \exp\!\left(-\frac{1}{2\sigma^2}(y_i - b_0 - b_1 x_i)^2\right).$$

We then use that finding the (b_0^*, b_1^*) that maximize the likelihood of our dataset under our model is equivalent to minimizing the $-\log$ likelihood.

$$(b_0^*, b_1^*) = \arg\max_{b_0,b_1} L(b_0, b_1, \sigma \mid \mathcal{D}) \iff \arg\min_{b_0,b_1} \big[-\log L(b_0, b_1, \sigma \mid \mathcal{D})\big].$$

where we can drop any dependence on parameters not b_0, b_1.

$$\begin{aligned} -\log L(b_0, b_1, \sigma \mid \mathcal{D}) &= n\log(\sqrt{2\pi}\sigma) + \frac{1}{2\sigma^2}\sum_{i=1}^{n}\big(y_i - b_0 - b_1 x_i\big)^2 \\ &\propto \sum_{i=1}^{n}\big(y_i - b_0 - b_1 x_i\big)^2. \end{aligned}$$

Therefore the optimization problem we solve in ordinary least squares is

$$\begin{aligned} (b_0^*, b_1^*) &= \arg\min_{b_0,b_1} \sum_{i=1}^{n}\big(y_i - b_0 - b_1 x_i\big)^2 \\ &= \arg\min_{b_0,b_1} \sum_{i=1}^{n}\big(y_i - \hat{y}_i\big)^2 \end{aligned}$$

Interpretation: assuming i.i.d. Gaussian errors turns MLE into minimizing the sum of squared errors, the familiar OLS loss.

3.3 Closed form solution: Ordinary least squares solution for a linear model

The ordinary least-squares regression solution finds the line that minimizes the residual sum of squares, SSR. First consider the linear data-generating model with no intercept and two regressors:

$$y_i = b_1 x_{1i} + b_2 x_{2i} + \varepsilon_i, \qquad i = 1, \ldots, n.$$

In matrix form $X \in \mathbb{R}^{n\times p}$ if there are n datapoints and k features:

$$X = \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \\ \vdots & \vdots \\ x_{1n} & x_{2n} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

Then the OLS estimator is

$$\hat{\mathbf{b}} = (X^\top X)^{-1} X^\top \mathbf{y}.$$

where for the inverse to exist we require, if $X \in \mathbb{R}^{n\times p}$, that $n \geq p$ (we have more data points than features, where features are not jus the input variables, but the *combinations/interactions* of them that we decide to use in our modeling process).

If we are in the case of simple linear linear regression, $\hat{y} = b_0 + b_1 x$ then the ordinary least squares solution is

$$\textbf{Slope:} \qquad b_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \mathrm{CORR}(X, Y) \cdot \frac{\mathrm{SD}(Y)}{\mathrm{SD}(X)} = \frac{\mathrm{COV}(X, Y)}{\mathrm{Var(X)}}$$

$$\textbf{Intercept:} \qquad b_0 = \bar{y} - b_1\,\bar{x}$$

where those are empirical correlations and standard deviations above.

3.4 Linear models with categorical input variables

We can also have categorical input variables (e.g. day of the week). In this setting, a categorical input variable with K "levels" uses $K-1$ independent variables. The omitted variable is the "reference" category, and the coefficients are to be interpreted as changes in output relative to the reference category.

$$\hat{y}_i = \beta_0 + \beta_1 \mathbb{I}(x_i = \text{ Class 1}) + \ldots + \beta_{K-1}\mathbb{I}(x_i = \text{ Class } K-1)$$

Basically think of this as a way to have an input variable that is a category, for which we essentially learn $K-1$ different slopes.

The coefficient β_k is interpreted as the predicted change in $\hat{y}_i$ from switching x_i from the reference category to category k.

3.5 Interactions between input variables

Linear models allow interaction terms $\beta_{jl} x_{ij} x_{il}$.

This allows the slope of x_{il} to *depend* on x_{ik}, and vice versa. Concretely, "the coefficient B_{jl} is the change in output per one-unit increase in x_{il} for a fixed x_{ik}, and vice versa."
If either input variable is categorical, the interaction allows us to have different coefficients for x_{il} depending on which category x_{il} takes.

3.6 Reasons for linear model failure

The model's accuracy and interpretability can be impaired by

- The presence of irrelevant input variables
- The presence of highly correlated input variables
- The presence of "too many" variables relative to the size of the dataset

3.7 Irrelevant input variables: p-values and confidence intervals

We use p-values and confidence intervals to tell if a variable is insignificant to the model.

The p-value quantifies how likely an input variable in linear regression is to get a non-zero coefficient purely by chance.

If a regression coefficient has $\mathbb{P}(> |t|) \geq 0.05$, then we say that "we are 95% confident that the true coefficient is different from zero. Therefore, we consider the input variable significant tat the 5% significance level)".

3.8 Dependent/co-linear input variables

Coefficients of highly correlated input variables divide the rate of change across multiple input variable dimensions. For example, if $x_{i,1} = x_{i,2}$ for all data i, then

$$\hat{y}_i = \beta_0 + \beta_2 x_{i,1} + \beta_1 x_{i,2} = \beta_0 + x_{i,1}(\beta_1 + \beta_2)$$

so $(\beta_1 + \beta_2)$ might make sense, but not β_1, β_2 individually. In particular, the signs of coefficients may be misleading. Inspect correlation matrix to identify co-linear input variables (look for $|\text{Cor}(x, y)| \approx 1$.

3.9 Distribution of predictions

- Since the assumed model has gaussian noise, our outputted coefficient β_i is actually gaussian with inferred mean and variance. (This is what we use to find p-values).
- From the normality of coefficients, the point prediction will be normal too with a mean of `point_prediction` and a standard deviation of `residual_std_error`.
- Therefore, we can also ask cumulative distribution questions on our point predictions.

4 Two-class logistic regression

Logistic regression is used to predict categorical probabilities. The model takes both continuous and categorical input variables as input, and outputs

$$\mathbb{P}(Y_i \text{ belongs to category } j)$$

for all categories $j \in [K]$, if there are K categories.

Everything here is for two-class prediction, where it suffices to only predict $\mathbb{P}(Y_i = 1) = p$. We discuss multinomial logistic regression later on.

4.1 Why does linear regression fail for classification?

Using linear regression to predict probabilities fails because linear regression asymptotes as $\pm\infty$, while clearly the probability must satisfy $p \in [0, 1]$. Linear regression also assumes normality of noise, which will not hold.

4.2 Logistic regression uses the sigmoid function

Recall the sigmoid function $\sigma : \mathbb{R} \to (0, 1)$ defined as

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

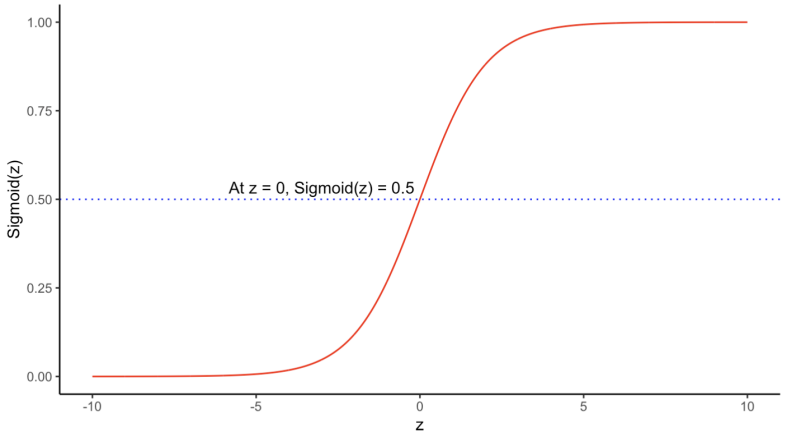


Figure 6: The shape of the logistic (sigmoid) function $\sigma(z)$

4.3 Logistic regression model definition

Logistic regression assumes the *true model*:

$$\mathbb{P}(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p)}}$$

and we learn the *estimated model*

$$\mathbb{P}(Y = 1) = \frac{1}{1 + e^{-(b_0 + b_1 X_1 + \ldots + b_p X_p)}}$$

by learning the model's coefficients

$$b_0,\ b_1,\ \ldots,\ b_p.$$

4.4 Odds in terms of probabilities and coefficients

We define odds in terms of probabilities:

$$\text{Odds}(Y_i = 1) = \frac{\mathbb{P}(Y_i = 1)}{\mathbb{P}(Y_i = 0)}$$

The coefficients of logistic regression models have a natural interpretation using odds

$$\log(\text{Odds}(Y_i = 1)) = (b_0 + b_1 x_1 + \ldots + b_p x_p)$$

Therefore,

$$\mathbb{P}(Y_i = 1) = \frac{1}{1 + e^{-\text{Odds}(Y_i=1)}}$$

4.5 From probabilities to classes using a cutoff

The output of the logistic regression model will be a probability for each of the categories, so we use a user-inputted probability cutoff to obtain predicted classes from probabilities. For example, we might predict the class with the highest probability, or set a hard threshold.

4.6 Assessing classification model quality

The first step of assesing the quality of a classification model is to use a confusion matrix to evaluate predictions.

		In Reality	
		Defaulted Y = 1	Repaid Y = 0
Model Prediction	Will default Pr(Y = 1) >= cutoff	A	B
	Will repay Pr(Y = 1) < cutoff	C	D

$$\text{Accuracy} = \frac{\text{A} + \text{D}}{\text{A} + \text{B} + \text{C} + \text{D}}$$

$$\text{Error rate} = 1 - \text{Accuracy}$$

Now consider the following rates:

- True Positive Rate (TPR) = correctly predicted positive ÷ all actual positives.
- False Negative Rate (FNR) = incorrectly said negative when actually positive ÷ all actual positives.
- TPR + FNR = 1.0
- False Positive Rate (FPR) = incorrectly said positive when actually negative ÷ all actual negatives.
- True Negative Rate (TNR) = correctly predicted negative ÷ all actual negatives.
- FPR + TNR = 1.0

We can use cutoffs to skew this one way or another (false negative is often much worse than false positive).

4.7 Precision, Recall, F1 score

Precision measures how many of the predicted positive examples are actually true positives. It is a measure of the model's ability to avoid false positives and so that every positive is actually positive.

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} \ + \ \text{False positives}}$$

Recall measures how many of the actual positive examples are correctly identified by the model. It is a measure of the model's ability to avoid false negatives and identify all positive examples correctly.

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} \ + \ \text{False negatives}}$$

The *F1 score* is used when both precision and recall matter

$$\text{Precision} = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

5 Multinomial logistic regression

5.1 Multiple logistic regression uses the softmax function

, Recall that the *softmax function* is a generalization of the sigmoid function: Given scores (logits) $z_1, \ldots, z_K \in \mathbb{R}$, the softmax maps them to a probability vector

$$\text{softmax} : \mathbb{R}^n \to [0,1]^n$$

where, for $z \in \mathbb{R}^n$

$$\text{softmax}(z)_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}, \qquad k = 1, \ldots, K,$$

which lies on the probability simplex (nonnegative and sums to 1) and is invariant to adding the same constant to all logits in the sense that $\text{softmax}(z) = \text{softmax}(z + c\mathbf{1})$.

5.2 Softmax reduces to sigmoid when $K = 2$

For $K = 2$, softmax reduces to the logistic sigmoid on the logit difference.

$$\text{softmax}(z_1, z_2)_1 = \sigma(z_1 - z_2), \qquad \text{softmax}(z_1, z_2)_2 = 1 - \text{softmax}(z_1, z_2)_1.$$

5.3 Definition of a multinomial logistic regression model

With K categories, set category K as the baseline. For non-baseline classes $k = 1, 2, \ldots, K-1$,

$$\Pr(Y = k) = \frac{e^{\beta_{k0} + \beta_{k1} x_1 + \cdots + \beta_{kp} x_p}}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0} + \beta_{l1} x_1 + \cdots + \beta_{lp} x_p}}.$$

and for the baseline Class K

$$\Pr(Y = K) = \frac{1}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0} + \beta_{l1} x_1 + \cdots + \beta_{lp} x_p}} \qquad \text{(class } K \text{ is the baseline).}$$

Since there are p features and K classes, the number of coefficients is $(p+1) \cdot (K-1)$.

5.4 Log odds = linear model

As before, multinomial logistic regression allows us to express the odds (ratio of probabilities with respect to the baseline category K) in terms of a linear model

$$\log\left(\frac{\Pr(Y = k)}{\Pr(Y = K)}\right) = \beta_{k0} + \beta_{k1} x_1 + \cdots + \beta_{kp} x_p, \qquad k = 1, 2, \ldots, K-1.$$

6 Classification and regression trees (CART)

Tree based models work well for prediction on continuous and categorical tabular data, but take different terms based upon the type of output variable:

- categorical output variable→ classification tree
- continuous output variable→ regression tree

6.1 Tree based methods intuition

In both classification trees and regression trees, the main method involves segmenting the feature space into a number of "simple" regions. A tree can be used to summarize these splitting rules.

- Post training, the outputted prediction for a given datapoint is the mean (if regression) *or* the most common class (if classification) of the training data belonging to the same region as that data point.
- Hence, essentially all we are doing is using baseline models, but on partitioned regions of the feature space.

6.2 The CART algorithm

How do we decide how to partition the feature space?

1. For each independent variable and each possible way to "split" the dataset based on that variable, calculate the reduction in *total* SSR

$$SSR(\texttt{model}) = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

   where in the baseline model we use $\hat{y}_i = \bar{y}_i$

2. Choose the split that generates the maximum reduction in *total* SSR.

3. Repeat.

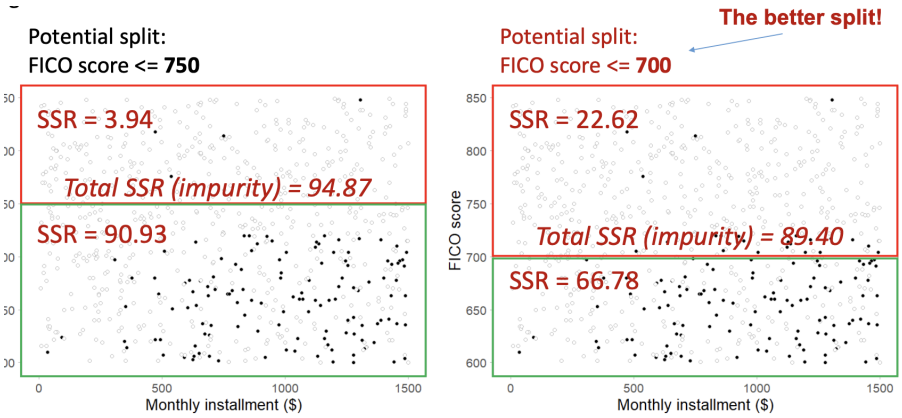


Figure 7: Select the partition that results in the lowest *total* SSR across all regions

6.3 Recursive splits

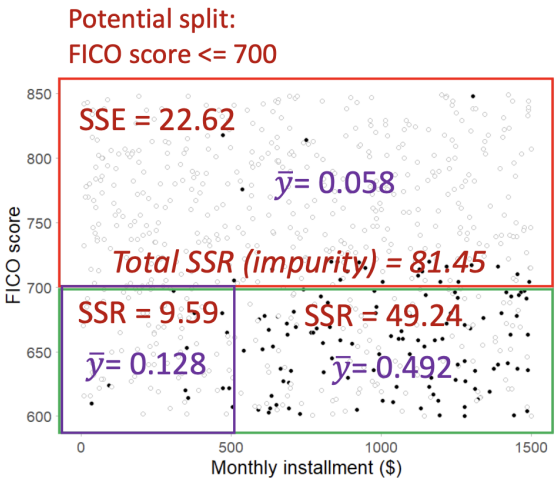


Figure 8: Recursive splits by selecting the partition that results in the lowest *total* SSR across all regions

6.4 Prediction with CART

Given a new data point, just "follow the splits" to the bottom of the tree. The meaning of the terminal node differs depending on whether it's a classification tree or a regression tree:

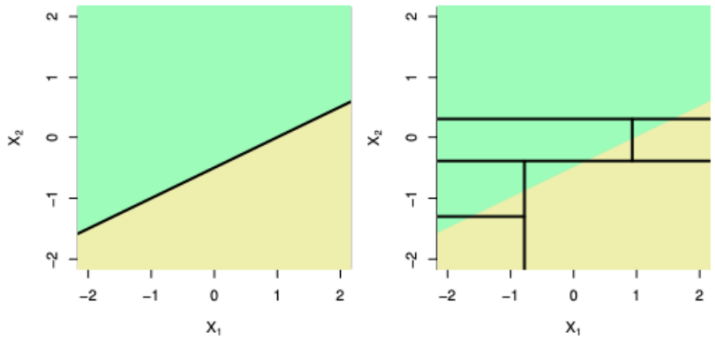
- For classification trees, each terminal node will be a probability distribution across the classes using the frequency of points in that region. Then use cutoffs to obtain probabilities from categorical predictions. (Or simply reply with the most common class of all data points in that region)
- For regression trees, the output is the mean value of the training dataset in the partition you end up in.

6.5 When to stop splitting

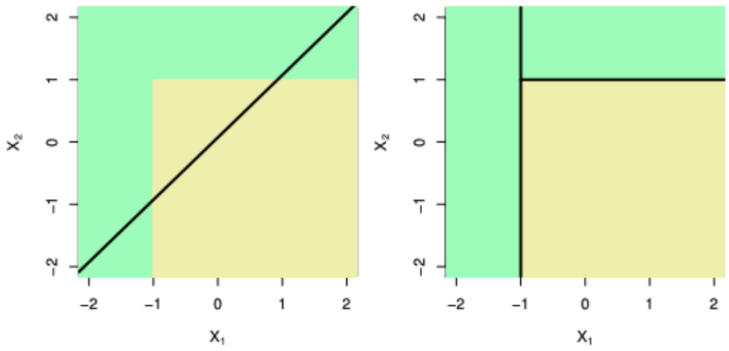
Too much splitting can result in overfitting. The splitting process is controlled by two key parameters:

- `minsplit` parameter is the minimum number of observations that must exist in a region in order for a split to be attempted
- `minbucket` parameter is a lower limit on the number of observations in each region. The smaller this parameter, more splits are possible
- `cp` parameter is known as the *(complexity parameter)*. Any split that does not decrease the overall lack of fit by a factor of `cp` is not attempted. The main role of this parameter is to save computing time by pruning off splits that are obviously not worthwhile.

6.6 CART vs logistic regression



Logistic regression is a better classifier



CART is a better classifier

6.7 For classification trees: the Classification Error Rate

In a classification tree, each terminal region m predicts the majority class in that region. The classification error rate for region m is the fraction of training points in m that are not in that majority class. For a given region m in a classification tree with K categories,

$$E(\text{region } m) = 1 - \max_{k \in [K]} \hat{p}_{mk}$$

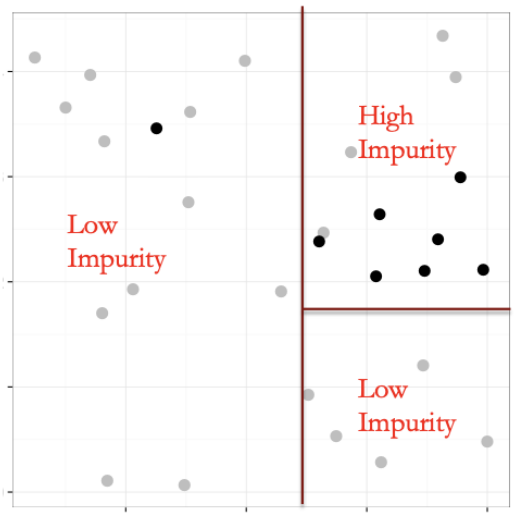
where

$$\hat{p}_{mk} = \frac{n_{mk}}{n_m}$$

is the proportion of training observations in the m-th region in the k-th class.

6.8 For classification trees: the Gini coefficient

In classification tree, regions that contain points belonging to multiples classes are likely to contribute to higher classification error. We say that partitions containing data with different class labels have *high impurity*.



High impurity regions = regions containing data with different labels

Gini impurity (for classification trees). For region R_m and class $k \in \{1, \ldots, K\}$, let $\hat{p}_{mk}$ be the proportion of observations in region R_m belonging to class k. The Gini impurity for region R_m is

$$G(R_m) \;=\; \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \;=\; 1 - \sum_{k=1}^{K} \hat{p}_{mk}^2.$$

Lower is better (purer). For two classes $K = 2$ with class proportion p_m,

$$G(R_m) = 2p_m(1 - p_m) = SSR(R_m) \qquad \in [0,\, \tfrac{1}{2}],$$

In general, $G(R_m) \in [0,\, 1 - \frac{1}{K}]$, with the maximum at the uniform distribution.

6.9 Benefits and downsides of CART

- Intuitive and interpretable if the tree is small
- CART can learn non-linear relationships and can discover interactions between variables without explicitly incorporating interaction terms
- However, CART models have many hyperparameters and risk overfitting to training data.

7 k-fold cross-validation

TLDR: k-fold cross validation is a method used for automated hyperparameter selection.

7.1 Cross validation: "divide, train, evaluate, average, select"

Divide the training set into k folds, then train a model on $k-1$ out of the k folds, and evaluate on the k^{th} folds. Do this k times over all of the $k = \binom{k}{k-1}$ options, then average the loss over these k instances. Do this procedure for each possible hyperparameter value, then select the hyperparameter value which had the lowest averaged loss. For example, with $k = 5$, the following shows a single trial for a given hyperparameter value

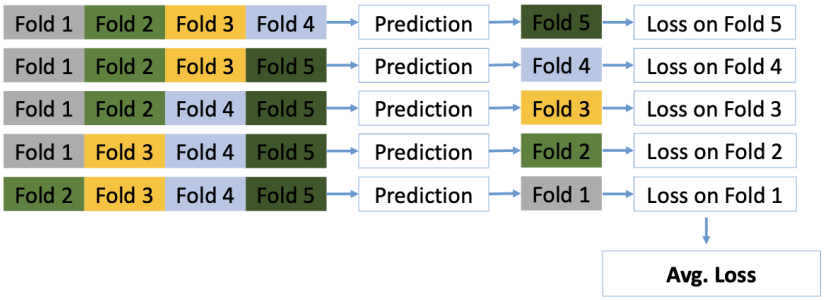


Figure 9: A single step of k-fold cross-validation. Do this for each possible hyperparameter value and select whichever had the lowest averaged loss on the k folds.

Plotting this averaged k^{th} fold loss as a function of hyperparameter value can help us understand when the model is overfitting

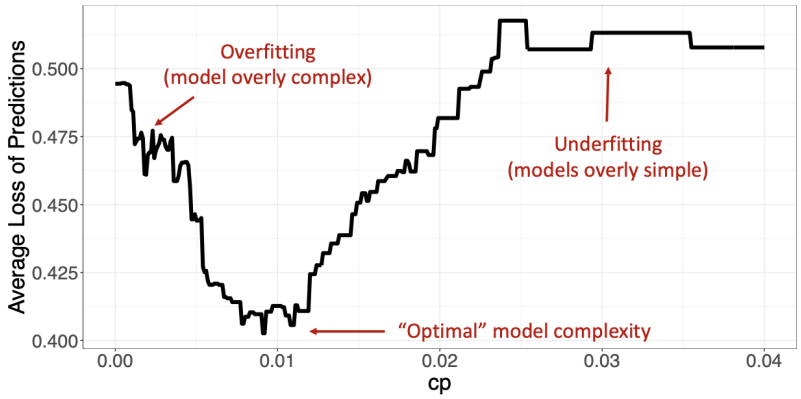


Figure 10: k-fold cross-validation helps us identify underfitting and overfitting

8 Random forests

A Random Forest is an ensemble of CART trees.

Due to their tree-like structure, CART models are non-robust in the sense that small changes in the data may lead to large changes in the final predicted model. Instead of a using single tree model that is subject to this high instability, we build a large number of different tree models (grow a forest), then then combine the trees' by averaging their predictions (regression) or selecting to most common class (classification) to create a more robust model

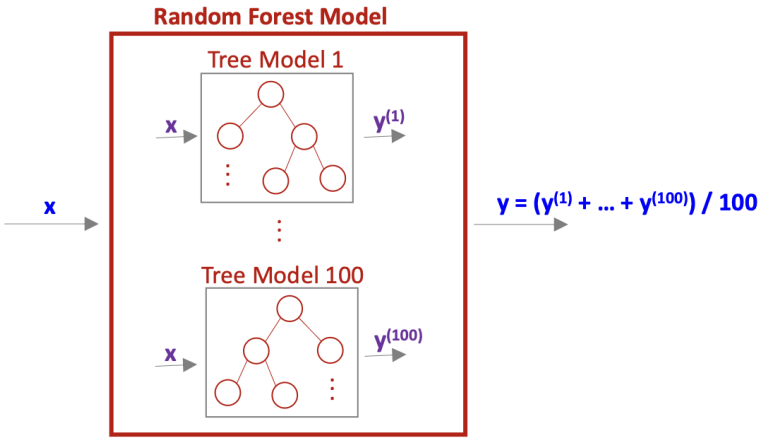


Figure 11: Random forests take the average (regression) or most common class (classification) across all predictions of the trees in the forest.

8.1 Why do random forests work

- Extreme errors cancel each other in the final prediction.
- Works best when, or because, errors are uncorrelated (weakly correlated)
- "wisdom of the crowd"
- However, we need methods to ensure that the trees in our forest are diverse: Bootstrapping, Bagging

8.2 How to create random forests (idea 1): Bootstrapping

"How to create diverse trees from the same dataset"

Bootstrapping trains each tree in the forest on random subset of the training dataset.

- Crucially, bootstrapping creates these subsets of the dataset by sampling with replacement from the training set, selecting the same number of observations as in the training set. Hence samples in the training set may appear more than once
- This sampling is done for every tree in the forest.
- (Also known as bagging)

8.3 How to create random forests (idea 2): Subset of variables

"How to create diverse trees from the same dataset"

The subset of variables method builds a tree by only considering a random subset of variables at each level of the tree, which is then resampled at each level. This is then done for every tree

in the forest.

Branching using subsets of variables explicitly forces trees in random forests to be diverse by forcing us to branch in diverse ways, as opposed to every model in the forest simply only branching on the strong predictors.

8.4 Random forest parameters

- `ntree`: number of CART trees in the forest (default value = 500) this is for both bootstrapping and the random subset method
- for the random subset method, `mtry` is the number of variables considered at each split of the trees (default value is $k/3$ for regression and $\sqrt{(}k)$ for classification)

9 Gradient boosting machines

Gradient boosting machines (GBMs) are another form of ensemble learning (aside from random forests) which typically use trees as base models. But while random forests use large trees that are trained independently then combined, GBMs use shallow trees that are trained *adaptively* then combined.

9.1 Boosted trees algorithm

At each iteration, the next CART tree in the ensemble is built to perform well on the data points where the existing trees have done poorly.

We observe a dataset $\{(y_i, \mathbf{x}_i) : i = 1, \ldots, n\}$ and build a model of the form $y \sim x$ and denote its prediction function by $\hat{f}(\mathbf{x})$. However, residuals will remain: for each observation,

$$y_i \;=\; \hat{f}(\mathbf{x}_i) + r_i.$$

The boosting step is to consider the new dataset $\{(r_i, \mathbf{x}_i) : i = 1, \ldots, n\}$ and build a *boosting* model of the form $r \sim x$ using the function $\hat{f}^{\,b}(\mathbf{x})$. We then update the original model $\hat{f}$ via a small *shrinkage/learning-rate* parameter $\eta > 0$:

$$\hat{f}(\mathbf{x}) \;\leftarrow\; \hat{f}(\mathbf{x}) + \eta\, \hat{f}^{(b)}(\mathbf{x}).$$

Equivalently, after b rounds,

$$\hat{f}(\mathbf{x}) \;=\; \hat{f}(\mathbf{x}) + \eta \sum_{j=1}^{b} \hat{f}^{(b)}(\mathbf{x}).$$

In the b^{th} round, the new tree $\hat{f}^{(b)}(\mathbf{x}_i)$ is fit to the $(b)^{th}$ order residuals

$$r_i^{(b)} \;=\; y_i - \hat{f}(\mathbf{x}_i) - \hat{f}^{(1)}(\mathbf{x}_i) - \ldots - \hat{f}^{(b-1)}(\mathbf{x}_i) \quad \forall \text{ observations } i.$$

- If the residual at $\mathbf{x}_i$ is positive ($r_i^{(b)} > 0$), we want the next tree to *bump up* our prediction.
- If the residual at $\mathbf{x}_i$ is negative ($r_i^{(b)} < 0$), we want the next tree to *push down* our prediction.

This is effected by the update

$$\hat{f}(\mathbf{x}) \;\leftarrow\; \hat{f}(\mathbf{x}) + \eta\, \hat{f}_b(\mathbf{x}).$$

Why do we need η? Because it helps avoid overfitting to the residuals.

9.2 Parameters for boosting

- shrinkage (η) controls the effect to which we fit to residuals
- `n.trees` controls the total number of boosting iterations
- `interaction.depth`: maximum number of variable interactions, i.e. the number of splits for a given residual
- `n.minobsinnode`: same as minbucket in CART

9.3 Boosting vs random forests

Conventional wisdom is that boosting typically dominates random forests

- Random forests are easier to tune since it has fewer parameters and is generally more robust by the consequence of averaging.
- Boosting uses an adaptive design of trees, compared to random forests' independant design of trees.

- Random forests' trees are deep so each tree typically over-fits the data but they are averaged to reduce overfitting
- Boosting's trees are shallow so each tree typically under-fits the data but they are added together in such a way that the overall model fits the data

10 Regularization

Regularization helps avoid overfitting by which penalizing model complexity. It is an explicit departure from he "best fit to the observed data".

Regularization incorporates a complexity penalty directly into the optimization problem so that we have something that looks like

$$\min\Big(\ \texttt{SSR}\ \ +\ \ \text{Penalty}(\texttt{model\_complexity})\Big)$$

.

We have three types of regularization

- L0/Best-subset
- L1/Lasso
- L2/Ridge

Here is the model that we assume:

$$\begin{aligned} \hat{y}_i &= b_0 + b_1 x_{i,1} + b_2 x_{i,2} + \cdots + b_k x_{i,k}, && i = 1, \ldots, n, \\ e_i &= y_i - \hat{y}_i, && i = 1, \ldots, n. \end{aligned}$$

where e_i is the residual for every datapoint i, and b_j for $j \in [k]$ are our (linear) coefficients. Recall that the sum of squared residuals is

$$\text{SSR}(\texttt{model}) = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

10.1 L0/best-subset regularization

$$\min_{b_0,\ldots,b_k}\ \text{SSR}(b_0, \ldots, b_k)\ +\ \lambda \sum_{j=1}^{k} \mathbf{1}\{\, b_j \neq 0 \,\}$$

The penalty is the number of variables with non-zero coefficient values b_j)

10.2 L1/lasso regularization

$$\min_{b_0,\ldots,b_k}\ \text{SSR}(b_0, \ldots, b_k)\ +\ \lambda \sum_{j=1}^{k} |b_j|$$

Almost as easy to compute as L2/ ridge regression. Tends to set $b_j = 0$ for a reasonable number of variables (more than ridge, less than best subset). NOTE: this penalty only makes sense if the input variables $\{x_i\}$ are on comparable scales.

10.3 L2/ridge regularization

$$\min_{b_0,\ldots,b_k}\ \text{SSR}(b_0, \ldots, b_k)\ +\ \lambda \sum_{j=1}^{k} b_j^2$$

Tends to produce lots of b_j values with annoyingly small coefficient values, hence no sparsity. NOTE: this penalty only makes sense if the input variables $\{x_i\}$ are on comparable scales.

10.4 Why L2 results in coefficients $\beta_j = 0$ but not L1

The ℓ_2 unit ball is $B_2 = \{x \in \mathbb{R}^n \mid \|x\|_2 \leq 1\}$ Geometrically this looks like a circle.

The ℓ_1 unit ball is $B_1 = \{x \in \mathbb{R}^n \mid |x| \leq 1\}$. geometrically this looks like a diamond.

In the diagram below, the blue region is the constrain set, and the red ovals are equal–loss contours of SSR. The place where these SSR contours touch the constraint set is optimal. In Lasso this happens at a "sharp" point, while in Ridge this happens at a "smooth" point. Hence we Lasso is usually sparse, while Ridge usually has lots of small but non-zero coeffecients.

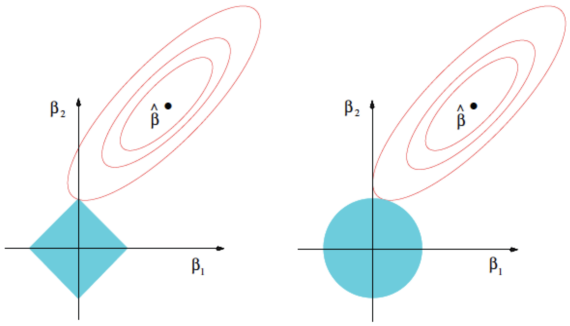


Figure 12: Geometric intuition on why Lasso results in zero coefficients, but not Ridge

11 Clustering using the k-means algorithm

In unsupervised learning, we have *unlabeled data* and aim to find natural groups in the data (assign it into groups). Evaluating the quality of clustering is hard since there is no ground truth for simple questions like "how many clusters there are in the data?". The figure below shows the difference between classification and clustering.

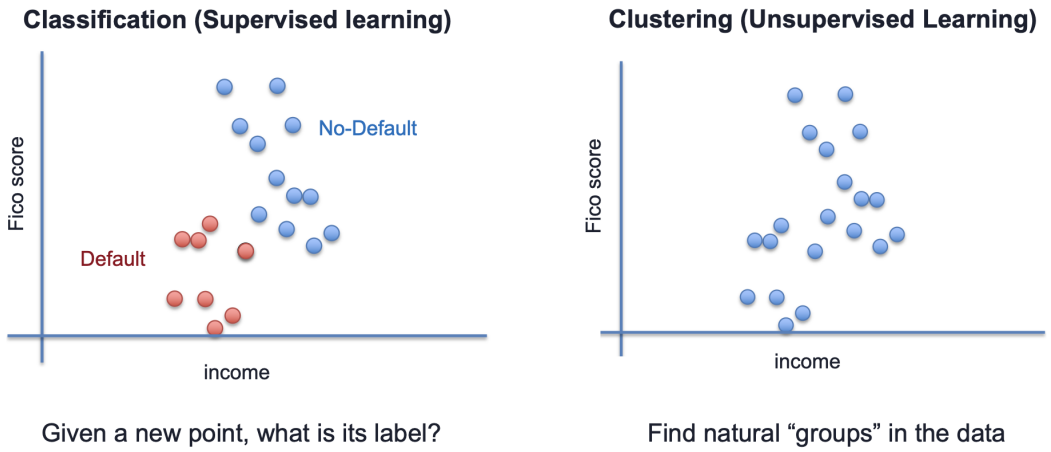


Figure 13: Classification uses labeled data to predict. Clustering uses unlabeled data to find natural groups in the data.

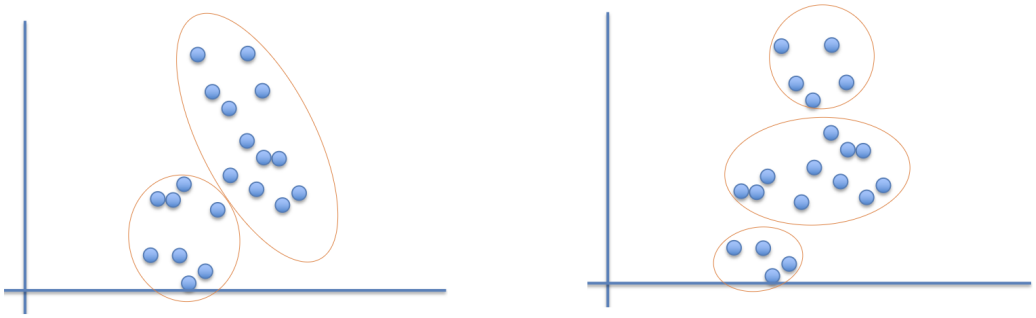


Figure 14: For a given dataset, there exist many plausible "groupings" of the data.

11.1 The goal of clustering

The goal of clustering is to divide the data into groups that share relevant characteristics and are interpretable.

11.2 Measure of distance = measure of dissimilarity

For two points $x, y \in \mathbb{R}^n$

$$\text{Distance}(x, y) = (x_1 - y_1)^2 + \ldots + (x_n - y_n)^2$$

ding by the standard deviation

11.3 Decomposition of sum of squared distances

Total sum of squares = "Within-cluster" sum of squares + "Between-cluster"sum of squares

For example if we have only two clusters $\mathcal{G}_1$ and $\mathcal{G}_2$, then

$$\sum_{i\neq j}(x_i - x_j)^2 = \sum_{i,j\in\mathcal{G}_1}(x_i - x_j)^2 + \sum_{i,j\in\mathcal{G}_2}(x_i - x_j)^2 + \sum_{i\in\mathcal{G}_1, j\in\mathcal{G}_2}(x_i - x_j)^2$$

Clustering algorithms pick $\mathcal{G}_1$ and $\mathcal{G}_2$ in an attempt to minimize "within-cluster" sum of squares, or equivalently maximize "between- cluster" sum of squares.

11.4 The Centroid of a Set of Data Points

The centroid of multiple data points is the average, and has the additional property that it minimizes the sum of *squared* distances over all points

$$\begin{aligned}\text{Centroid}(x_1, \ldots, x_n) &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ &= \arg\min_{\mu} \sum_{i=1}^{n}(x_i - \mu)^2\end{aligned}$$

11.5 Baseline model for clustering

The baseline model is the sum of squares of the entire dataset using one cluster $\mathcal{G}_1 = \mathcal{D}$. Therefore, the Total sum of squares is just the "within cluster" sum of squares.

11.6 The k-means algorithm

The $k-$means algorithm is a way to select K clusters $\mathcal{G}_i$ for $k \in [K]$.

1. Select a number of clusters K
2. Randomly choose K cluster centroid locations
3. Assign observations to the closest centroid
4. Recalculate centroids as the average of the assigned observations
5. Repeat steps 3 and 4 until no observations get reassigned

The choice of starting centroids matters, so often specify an `nstart` parameter and take the average of the outputted centroids, then assign groups accordingly.

11.7 k-means as optimization

Given observations $x_1, \ldots, x_n \in \mathbb{R}^p$, partition the indices into K disjoint clusters $C_1, \ldots, C_K$ (with $\bigcup_{k=1}^{K} C_k = \{1, \ldots, n\}$) to minimize some function $W(C_k)$ of the points in set C_k.

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_j).$$

A common choice of $W(\cdot)$ is

$$W(C_k) = \frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij} - x_{i'j})^2, \qquad |C_k| = \text{size of cluster } C_k.$$

Therefore the combined objective is

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}.$$

11.8 Scree plot

A scree plot displays the tradeoff between number of clusters k and the within-cluster sum of squared distances

- As we increase k, this value should decrease.
- But, hopefully, the curve will flatten out at some point suggesting diminishing returns from adding more clusters
- If the line is flat, then this suggests that the data is not "cluster-able"

Figure 15: Scree plots show the diminishing marginal returns from adding more clusters

11.9 Parallel plots

Parallel plots show the mean of clusters across the various input variables.

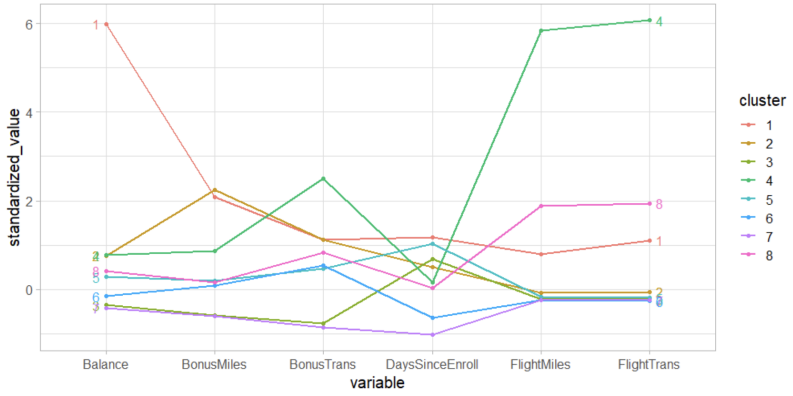


Figure 16: Use a Parallel plots to understand the difference between clusters.

12 Gaussian mixture models for clustering

12.1 Definition of a GMM

Our goal in Gaussian mixture models is to describe a set of data over $x \in \mathbb{R}^d$ using a mixture of K Gaussian components. Let $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ denote the parameters:

- mixing weights $\pi_k \geq 0$, $\sum_{k=1}^{K} \pi_k = 1$,
- means $\mu_k \in \mathbb{R}^d$,
- (symmetric positive definite) covariances $\Sigma_k \in \mathbb{R}^{d\times d}$ with $\Sigma_k \succ 0$.

Therefore, the mixture density for a single data point x_n is

$$p(x_n \mid \Theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}\big(x_n \mid \mu_k, \Sigma_k\big),$$

where $\mathcal{N}(x \mid \mu, \Sigma)$ denotes the Gaussian density $\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\!\big(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\big)$. Furthermore, if we introduce a categorical latent variable $z_n \in \{1, \ldots, K\}$ indicating which component generated x_n, then *responsibility* (posterior soft assignment) of component k for x_n is

$$P(z_n{=}k \mid x_n, \Theta) = \frac{\pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n \mid \mu_j, \Sigma_j)} = \gamma_{nk}, \qquad \text{where} \qquad \sum_{k=1}^{K} \gamma_{nk} = 1 \ \forall n.$$

12.2 GMM likelihood

Given data $X = \{x_n\}_{n=1}^{N}$, the (incomplete-data) likelihood of the data under the parameters is

$$\mathcal{L}(X \mid \Theta) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k),$$

therefore log likelihood is

$$\begin{aligned}
\ell(X \mid \Theta) &= \log\Big(\mathcal{L}(X \mid \Theta)\Big) \\
&= \log\Big(\prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k)\Big) \\
&= \sum_{n=1}^{N} \log\Big(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k)\Big).
\end{aligned}$$

However, maximizing this log-likelihood directly is because of the inner sum over components. EM addresses this by introducing *latent one-hot indicators*

12.3 Latent-variable view of GMMs

We assume the following generative model for each datapoint $x_n \in \mathbb{R}^d$:

1. Draw a latent component index $z_n \in \{1, \ldots, K\}$ with prior $P(z_n = k) = \pi_k$ where $\sum_k \pi_k = 1$.
2. Given this $z_n = k$ chosen above, draw $x_n \sim \mathcal{N}(\mu_k, \Sigma_k)$. Across datapoints we assume i.i.d. sampling.

We emphasize that we *first* draw the discrete component index $z_n \in \{1, \ldots, K\}$ from a categorical distribution with probabilities $\{\pi_k\}_{k=1}^{K}$ (these are part of Θ), and *then* draw the continuous observation $x_n \in \mathbb{R}^d$ from the Gaussian associated with that component:

$$P(z_n = k \mid \Theta) = \pi_k, \qquad p(x_n \mid z_n = k, \Theta) = \mathcal{N}(x_n \mid \mu_k, \Sigma_k).$$

Given this ordering, then by the chain rule

$$p(x_n, z_n \mid \Theta) = P(z_n \mid \Theta)\, p(x_n \mid z_n, \Theta).$$

Conditioning on Θ fixes the parameters (so they are not random here). The factor $P(z_n \mid \Theta)$ says which component was chosen; the factor $p(x_n \mid z_n, \Theta)$ then gives the density of x_n for that chosen component. Multiplying them gives the joint probability of seeing *both* the component choice and the resulting data point under the model.

12.4 One-hot encodings

Next, define let $Z = \{z_{nk}\}$ be the one-hot encoding of z_n:

$$z_{nk} \in \{0, 1\}, \qquad \sum_{k=1}^{K} z_{nk} = 1$$

and

$$z_{nk} = 1 \iff z_n = k.$$

Using the one-hot encoding,

$$p(x_n, z_n \mid \Theta) = P(z_n \mid \Theta)\, p(x_n \mid z_n, \Theta).$$

can be written as products that *select* the active component via exponents z_{nk}:

$$P(z_n \mid \Theta) = \prod_{k=1}^{K} \pi_k^{z_{nk}}, \qquad p(x_n \mid z_n, \Theta) = \prod_{k=1}^{K} \Big[\mathcal{N}(x_n \mid \mu_k, \Sigma_k)\Big]^{z_{nk}}.$$

Multiplying these gives the joint for one datapoint:

$$\begin{aligned} p(x_n, z_n \mid \Theta) &= \prod_{k=1}^{K} \Big[\pi_k^{z_{nk}}\, \mathcal{N}(x_n \mid \mu_k, \Sigma_k)^{z_{nk}}\Big] \\ &= \prod_{k=1}^{K} \Big[\pi_k\, \mathcal{N}(x_n \mid \mu_k, \Sigma_k)\Big]^{z_{nk}}, \end{aligned}$$

Independence across the N samples implies

$$p(X, Z \mid \Theta) = \prod_{n=1}^{N} p(x_n, z_n \mid \Theta) = \prod_{n=1}^{N} \prod_{k=1}^{K} \Big[\pi_k\, \mathcal{N}(x_n \mid \mu_k, \Sigma_k)\Big]^{z_{nk}}.$$

Taking logs yields the complete-data log-likelihood

$$\log p(X, Z \mid \Theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \Big(\log \pi_k + \log \mathcal{N}(x_n \mid \mu_k, \Sigma_k)\Big).$$

12.5 Expectation maximization principle

The EM algorithm alternates between the E-step and M-step

12.6 The E-step

The E-step compute the posterior over Z under current parameters Θ^{old}. In particular, *given* the current parameter values $\Theta^{\text{old}} = \{\pi_k^{\text{old}}, \mu_k^{\text{old}}, \Sigma_k^{\text{old}}\}_{k=1}^{K}$, compute the posterior distribution

over the latent assignment variables $Z = \{z_{nk}\}$ conditioned on the observed data $X = \{x_n\}_{n=1}^N$. Formally, this is the distribution $p(Z \mid X, \Theta^{\text{old}})$. These weights γ_{nk} act like soft counts and will drive the M-step updates. The E-step posterior responsibilities is

$$\gamma_{nk} \leftarrow \frac{\pi_k^{\text{old}}\,\mathcal{N}(x_n \mid \mu_k^{\text{old}}, \Sigma_k^{\text{old}})}{\sum_{j=1}^K \pi_j^{\text{old}}\,\mathcal{N}(x_n \mid \mu_j^{\text{old}}, \Sigma_j^{\text{old}})} \qquad \text{for all } n, k.$$

12.7 The M-step

The M-step maximize the expected complete-data log-likelihood

$$\mathcal{Q}(\Theta \mid \Theta^{\text{old}}) \;=\; \mathbb{E}_{Z|X,\Theta^{\text{old}}}\big[\log p(X, Z \mid \Theta)\big] \;=\; \sum_{n=1}^N \sum_{k=1}^K \gamma_{nk}\Big(\log \pi_k + \log \mathcal{N}(x_n \mid \mu_k, \Sigma_k)\Big),$$

where $\gamma_{nk} = P(z_n{=}k \mid x_n, \Theta^{\text{old}})$ which was calculated during the E-step.

For parameter update, let $N_k = \sum_{n=1}^N \gamma_{nk}$ denote the *effective membership* of component k. Maximizing $\mathcal{Q}(\,\cdot \mid \Theta^{\text{old}})$ subject to $\sum_k \pi_k = 1$ yields

$$\pi_k \leftarrow \frac{N_k}{N}, \qquad \mu_k \leftarrow \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}\, x_n,$$

$$\Sigma_k \leftarrow \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}\,(x_n - \mu_k)(x_n - \mu_k)^\top, \qquad \Sigma_k \succ 0.$$

12.8 Algorithm summary

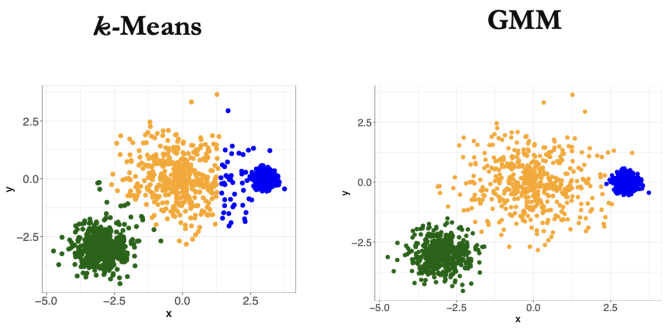
Initialize $\{\pi_k, \mu_k, \Sigma_k\}$ (e.g. k-means centroids for μ_k, uniform π_k, empirical Σ_k). Repeat until convergence:

1. **E-step:** compute γ_{nk} for all n, k.
2. **M-step:** update (π_k, μ_k, Σ_k) via the formulas above.

Monitor $\ell(\Theta; X)$; it increases monotonically and EM stops at a local optimum.

Benefits and tradeoffs to GMMs

- We get soft clustering use responsibilities γ_{nk} as membership scores, but can still obtain hard assignment by using $\hat{z}_n = \arg\max_k \gamma_{nk}$.
- We get density estimation for free using $p(x) = \sum_k \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)$.
- Must choose K, Gaussianity assumption, potential slow convergence



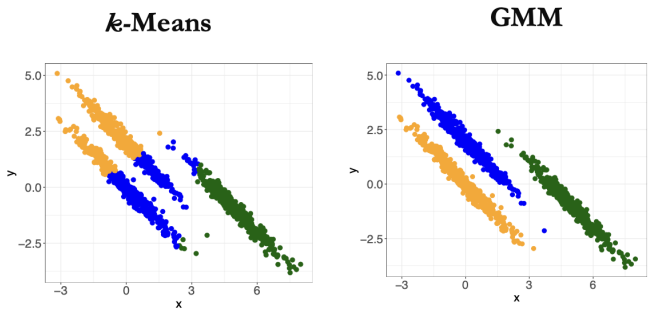


Figure 17: k-means vs GMM

13 Principal Component Analysis (PCA)

PCA is an orthonormal change of coordinates for centered data. Concretely, it finds a $V \in \mathbb{R}^{d\times d}$ with orthonormal columns ($V^\top V = I_d$, therefore the transformation is length and angle preserving) and maps a centered vector $x \in \mathbb{R}^d$ to $z = V^\top x \in \mathbb{R}^d$. Formally, with the PCA choice of $V \in \mathbb{R}^{d\times d}$:

(i) The score $z = V^\top x$ has uncorrelated components, $\mathrm{Cov}(z_i, z_j) = 0$ for all $i \neq j \implies \mathrm{Cov}(z) = \mathrm{diag}$.

(ii) The first coordinate z_1 has the largest variance among all z_i, z_2 has the second largest variance among all z_i, and so on.

(iii) The r-dimensional truncation of V yields the best rank-r approximation of the centered data matrix measured in the Frobenius norm.

13.1 PCA uses centered data

Let $X \in \mathbb{R}^{n\times d}$ be a data matrix whose ith row $x_i^\top$ is an observation in $\mathbb{R}^d$. Define the sample mean

$$\bar{x} \;=\; \frac{1}{n}\sum_{i=1}^{n} x_i \in \mathbb{R}^d, \qquad \text{and the centering operator } \; H \;=\; I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top.$$

The centered data matrix is

$$\tilde{X} \;=\; HX \in \mathbb{R}^{n\times d}, \quad \text{whose rows are } (x_i - \bar{x})^\top.$$

Unless otherwise stated, PCA is performed on $\tilde{X}$; centering is *essential* for standard PCA.

13.2 Empirical covariance matrix

Let $x \in \mathbb{R}^d$ be a zero-mean random vector with population covariance $\Sigma_X = \mathbb{E}[xx^\top]$.

From data iid data, the (biased) sample covariance is

$$S_{\text{bias}} \;=\; \frac{1}{n}\tilde{X}^\top\tilde{X} \in \mathbb{R}^{d\times d}, \qquad \text{which satisfies} \qquad \mathbb{E}[S_{\text{bias}}] = \frac{n-1}{n}\,\Sigma_X.$$

The unbiased version is

$$S_{\text{unbiased}} \;=\; \frac{1}{n-1}\tilde{X}^\top\tilde{X}, \qquad \text{which satisfies} \qquad \mathbb{E}[S] = \Sigma_X.$$

The scalar factor between S_{unbiased} and S_{bias} does not affect eigenvectors and only rescales eigenvalues. In what follows, any choice proportional to $\tilde{X}^\top\tilde{X}$ is acceptable; for definiteness we use

$$S \;=\; \frac{1}{n}\tilde{X}^\top\tilde{X}.$$

13.3 Finding principal directions via KKT (no Lagrangian)

Our goals are to find the d directions v_k so that (i) v_k are orthonormal, (ii) their associated score variances are in nonincreasing order, (iii) the score coordinates are pairwise uncorrelated.

The *first* principal direction $v_1 \in \mathbb{R}^d$ solves

$$\max_{v\in\mathbb{R}^d} \; v^\top S v \quad \text{s.t.} \quad \|v\|_2^2 = 1. \tag{3}$$

The KKT conditions are necessary in this setting. Using

$$f(v) = v^\top S v \quad \Longrightarrow \quad \nabla f(v) = 2Sv$$

and

$$h(v) = \|v\|_2^2 - 1 = 0 \quad \Longrightarrow \quad \nabla h(v) = 2v,$$

it is *necessary* that there exist an $\lambda \in \mathbb{R}$ such that

$$-\nabla f(v) = \lambda \nabla h(v) \quad \Longrightarrow \quad \exists \lambda \in \mathbb{R} \quad s.t. \quad Sv = \lambda v,$$

thus any optimizer v is a unit eigenvector of S. Notice that the objective becomes

$$v^\top S v = v^\top \lambda v = \lambda v^\top v = \lambda,$$

therefore maximizing variance $v^\top S v$ is equivariant to selecting the eigenvector associated with the largest eigenvalue of S. More generally, the k^{th} principal component direction solves

$$\max_{v \in \mathbb{R}^d} v^\top S v \quad \text{s.t.} \quad \|v\|_2 = 1, \;\; v^\top v_j = 0, \; j = 1, \ldots, k-1. \tag{4}$$

The solution is v_k, the eigenvector of S with the k^{th} largest eigenvalue λ_k, with $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$. The principal axes $\{v_k\}_{k=1}^d$ are orthonormal.

13.4 SVD of the centered data

Let the (full) singular value decomposition (SVD) of $\tilde{X} \in \mathbb{R}^{n \times d}$ be

$$\tilde{X} \;=\; U \Sigma V^\top,$$

where $U \in \mathbb{R}^{n \times n}$, $V \in \mathbb{R}^{d \times d}$ are orthogonal, and $\Sigma \in \mathbb{R}^{n \times d}$ is rectangular diagonal with nonnegative entries

$$\sigma_1 \geq \cdots \geq \sigma_\rho > 0, \qquad \sigma_{\rho+1} = \cdots = \sigma_r = 0,$$

with $\rho = \text{rank}(\tilde{X}) \leq \min\{n, d\}$ and the number of eigenvalues $r = \min\{n, d\}$. Then

$$S \;=\; \frac{1}{n} \tilde{X}^\top \tilde{X} \;=\; \frac{1}{n} V \Sigma^2 V^\top \in \mathbb{R}^{d \times d}.$$

which is itself an eigen-decomposition of S. Hence the eigenvectors of S are the columns of V, and the eigenvalues are

$$\lambda_k \;=\; \frac{\sigma_k^2}{n} \quad (k = 1, \ldots, \rho), \qquad \lambda_k \;=\; 0 \quad (k = \rho + 1, \ldots, d).$$

where σ_k are the sigular values from the SVD of $\tilde{X}$. (To count the number of non-zero eigenvalues, we used $\text{rank}(\tilde{X}^\top \tilde{X}) = \text{rank}(\tilde{X}) = \rho$.)

13.5 "Thin" SVD provides loss-less representation of $\tilde{X}$

Keeping only the nonzero singular values gives the "thin" SVD

$$\tilde{X} \;=\; U_\rho \Sigma_\rho V_\rho^\top, \qquad U_\rho \in \mathbb{R}^{n \times \rho}, \;\; \Sigma_\rho \in \mathbb{R}^{\rho \times \rho}, \;\; V_\rho \in \mathbb{R}^{d \times \rho},$$

which is a loss-less representation of our data.

13.6 Principal component coordinates

For the full-dimensional $V \in \mathbb{R}^{d\times d}$ and $z = V^\top x \in \mathbb{R}^d$ with centered x,

$$\mathrm{Cov}(z) \;=\; \mathbb{E}[zz^\top] \;=\; \mathbb{E}[V^\top x x^\top V] \;=\; V^\top \Sigma_X V \;=\; \mathrm{diag}(\lambda_1, \ldots, \lambda_\rho, 0, \ldots, 0).$$

Given these above eigenpairs $\{(v_k, \lambda_k)\}_{k\in[d]}$ of S, the k^{th} score vector (the vector of the n samples projected along direction v_k) is

$$z^{(k)} \;=\; \tilde{X} v_k \in \mathbb{R}^n, \qquad Z \;=\; \tilde{X} V \;=\; [\, z^{(1)} \;\cdots\; z^{(d)} \,] \in \mathbb{R}^{n\times d}.$$

Since $\lambda_k = 0$ for $k > \rho$, the last $d - \rho$ columns of Z are identically zero. Restricting to the informative part,

$$Z_\rho \;=\; \tilde{X} V_\rho \in \mathbb{R}^{n\times\rho}, \qquad \frac{1}{n} Z_\rho^\top Z_\rho \;=\; \Lambda_\rho \;=\; \mathrm{diag}(\lambda_1, \ldots, \lambda_\rho).$$

13.7 Best rank p approximation of $\tilde{X}$

When we deliberately reduce dimension, PCA provides the optimal rank-p linear approximation (in Frobenius norm). Define the rank-p truncation

$$\tilde{X}_p \;=\; U_p \Sigma_p V_p^\top, \qquad U_p \in \mathbb{R}^{n\times p},\; \Sigma_p \in \mathbb{R}^{p\times p},\; V_p \in \mathbb{R}^{d\times p}.$$

Theorem (Eckart–Young–Mirsky): For any $1 \le p \le \rho$, the best rank-p approximation of $\tilde{X}$ in Frobenius norm is

$$\tilde{X}_p \;=\; U_p \Sigma_p V_p^\top,$$

obtained by truncating to the top p singular values/vectors. Moreover,

$$\|\tilde{X} - \tilde{X}_p\|_F^2 \;=\; \sum_{j>p} \sigma_j^2 \;=\; n \sum_{j>p} \lambda_j.$$

13.8 Explained variance (full and informative parts)

The total empirical variance of $\tilde{X}$

$$\begin{aligned} \mathrm{tr}(S) &= \frac{1}{n}\mathrm{tr}(\Sigma^2) \\ &= \frac{1}{n} \sum_{i=1}^{\rho} \sigma_i^2 \end{aligned}$$

since $S = \frac{1}{n} V \Sigma^2 V^\top$ and trace is invariant under orthogonal similarity. Furthermore, the fraction of empirical variance captured by the top p principal component directions ($1 \le p \le \rho$) is

$$\frac{\sum_{k=1}^{p} \lambda_k}{\sum_{k=1}^{\rho} \lambda_k} \;=\; \frac{\sum_{k=1}^{p} \sigma_k^2}{\sum_{k=1}^{\rho} \sigma_k^2}.$$

13.9 Geometric intuition

The data $x \in \mathbb{R}^d$ exists as a cloud of points in some $d-$dimensional subspace. PCA can be thought of as a projection of these points onto the p-dimensional PC subspace using the

projection $P_p = V_p V_p^\top$. The orthogonal residual is $(I - P_p)x$.

13.10 Properties and invariances

We now know that PCA is essentially just SVD on the data $\tilde{X}$. Therefore, any operation's effect on PCA is determined by its effect on the SVD of $\tilde{X}$

1. **Translation invariance.** Adding a constant vector to all samples changes only the mean; $\tilde{X}$, S, and the PCs are unchanged, (but we needed mean zero to find PCA in the first place).
2. **Equivariance to orthonormal transforms.** If features are rotated/reflected by $Q \in \mathbb{R}^{d\times d}$ with $Q^\top Q = I_d$, then $\tilde{X} \mapsto \tilde{X}Q$ implies $S \mapsto Q^\top S Q$ and $v_j \mapsto Q^\top v_j$, with eigenvalues unchanged.

13.11 Formulas to know

1. Empirical covariance: for $\tilde{X} \in \mathbb{R}^{n\times d}$

$$S_{\text{bias}} = \frac{1}{n}\tilde{X}^\top \tilde{X}, \quad S_{\text{unbiased}} = \frac{1}{n-1}\tilde{X}^\top \tilde{X}, \quad \text{use } S = \frac{1}{n}\tilde{X}^\top \tilde{X}.$$

2. Compute PCs via SVD: first decompose

$$\tilde{X} = U\Sigma V^\top, \qquad \sigma_1 \geq \cdots \geq \sigma_d > 0,$$

   so that

$$S = \frac{1}{n}\tilde{X}^\top \tilde{X} = \frac{1}{n} V \Sigma^2 V^\top,$$

   so the eigenvectors of S are the columns of V. If $\rho = \operatorname{rank}(\tilde{X})$, then the eigenvalues

$$\lambda_k = \frac{\sigma_k^2}{n} \ (k \leq \rho), \qquad \lambda_k = 0 \ (k > \rho).$$

3. Thin SVD is loss-less for dimension ρ:

$$\tilde{X} = U_\rho \Sigma_\rho V_\rho^\top, \qquad U_\rho \in \mathbb{R}^{n\times\rho}, \ \Sigma_\rho \in \mathbb{R}^{\rho\times\rho}, \ V_\rho \in \mathbb{R}^{d\times\rho}.$$

4. Score vectors from data and loadings: the k^{th} score is the vector obtained by projecting each datapoint along direction v_k

$$z^{(k)} = \tilde{X} v_k \in \mathbb{R}^n$$

   and stacking columns gives

$$Z = \tilde{X}V = \left[z^{(1)} \ \cdots \ z^{(d)}\right], \qquad \text{where} \qquad \frac{1}{n} Z^\top Z = \operatorname{diag}(\lambda_1, \ldots, \lambda_\rho, 0, \ldots, 0),$$

   hence the d scores are uncorrelated

13.12 Practical considerations

- Centering is mandatory
- Scaling matters: if variables have different units/scales, normalize columns before PCA
- Outliers: variance is sensitive to outliers

- Uncorrelated vs. independent. PC scores are pairwise uncorrelated; they are independent only under stronger assumptions (e.g., joint Gaussianity).