

Machine Learning Review

John Semerdjian

Model Assessment

2x2 Table	Correct (Truth)	Not Correct (Truth)
Predict Outcome Positive	TP	FP
Predict Negative	FN	TN

Precision: ability of the classifier not to label as positive a sample that is negative:

$$\frac{TP}{TP + FP} = P(\text{really positive} \mid \text{label positive})$$

Recall (Sensitivity, True Positive Rate): ability of the classifier to find all the positive samples:

$$\frac{TP}{TP + FN} = P(\text{label positive} \mid \text{really positive})$$

Specificity (True Negative Rate): ability to exclude a condition correctly:

$$\frac{TN}{TN + FP} = P(\text{label positive} \mid \text{really negative})$$

F1 Measure: weighted harmonic mean, very conservative average, but most people used the balanced

$$F_1 = 2 \frac{PR}{P + R}$$

ROC Curve: rates performance of binary classifier system; false positive rate (x-axis) vs. true positive rate (y-axis)

(Root) Mean Squared Error: measure of estimation error for linear regression.

Regression

	Solving for \mathbf{w}	Gradient, $w^{n+1} = w^n - \eta \nabla_w$	Loss Function
OLS	$\mathbf{w}^T = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$	$\frac{1}{n} \mathbf{X}^T (\mathbf{y} - \mathbf{X} \mathbf{w}^T)$	$\ \mathbf{y} - \mathbf{X} \mathbf{w}^T\ _2^2$
Linear Ridge	$\mathbf{w}^T = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$	$\frac{1}{n} (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) \mathbf{w}^T - \mathbf{X}^T \mathbf{y}$	$\ \mathbf{y} - \mathbf{X} \mathbf{w}^T\ _2^2 + \lambda \ \mathbf{w}\ _2^2$
Linear Lasso	Quadratic programming		$\ \mathbf{y} - \mathbf{X} \mathbf{w}^T\ _2^2 + \lambda \ \mathbf{w}\ _1$
Logistic	Newton-Raphson/GD	$\frac{1}{n} \mathbf{X}^T (\mathbf{y} - \mathbf{p})$	$-\sum_{i=1}^N y_i \log p_i + (1 - y_i) \log(1 - p_i)$

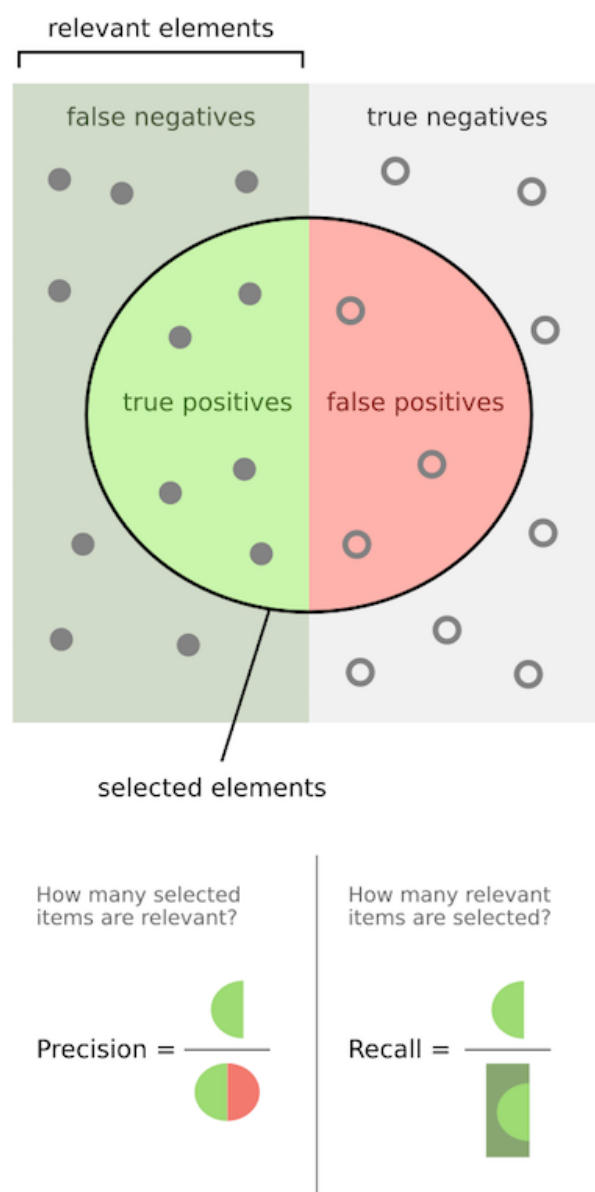


Figure 1: Precision & Recall

Linear Regression

Assumptions:

1. ϵ_i are IID with from $\sim N(0, \sigma^2)$
 - **Homoscedasticity:** error terms must have constant variance
 - Breaking this assumption means that the Gauss-Markov theorem does not apply
 - Heteroscedasticity can cause standard errors of coefficients to be biased, though coefficient estimates won't be biased
 - **Exogeneity assumption:** X_i were chosen by random by nature, independent from ϵ_i ("Endogeneity bias" = $\epsilon \not\perp X$)

$$E(\hat{\beta} | X) = \beta + \underbrace{(X^T X)^{-1} X^T E(\epsilon | X)}_{\text{if } X \perp \epsilon, \text{ then } E(\hat{\beta} | X) = \beta}$$

2. Y_i was determined by the response schedule: $Y_{i,x} = x\beta + \epsilon_i$
3. No **multicollinearity** (i.e., no correlated regressors)
 - Multicollinearity can cause inflated coefficient standard errors, nonsensical coefficients, and a singular sample covariance matrix.
 - Increased standard errors in turn means that coefficients for some independent variables may be found not to be significantly different from 0. In other words, by overinflating the standard errors, multicollinearity makes some variables statistically insignificant when they should be significant.
 - Without multicollinearity, those coefficients might be significant.
 - Use partial least squares or principal component regression.

Least Squares derivation:

$$\begin{aligned} RSS &= \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 = \sum_{i=1}^N (y_i - x_i^T \mathbf{w})^2 \\ &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T \mathbf{y} - \underbrace{\mathbf{y}^T \mathbf{X}\mathbf{w} - \mathbf{w}^T \mathbf{X}^T \mathbf{y}}_{\substack{\text{dim } 1 \times 1, = \text{to its transpose}}} + \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} \\ &= \mathbf{y}^T \mathbf{y} - 2\mathbf{w}^T \mathbf{X}^T \mathbf{y} + \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} \\ \frac{\partial RSS}{\partial \mathbf{w}} &= -2\mathbf{X}^T \mathbf{y} + 2\mathbf{X}^T \mathbf{X} \mathbf{w} = 0 \\ &= \underbrace{-2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w})}_{\text{derive from RSS w/ chain rule}} \\ \mathbf{w} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \\ \frac{\partial^2 RSS}{\partial \mathbf{w} \partial \mathbf{w}^T} &= 2\mathbf{X}^T \mathbf{X} \end{aligned}$$

Note: \mathbf{X} must have full column rank, and hence $\mathbf{X}^T \mathbf{X}$ is positive definite.

R^2 , **coefficient of determination:** measure of goodness of fit of linear model. The fraction (between 0 and 1) of MSE the model eliminates.

$$R^2 = \frac{\text{var}(X\hat{\beta})}{\text{var}(Y)} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

Feature Transformations

- **Log:** Performing a \log_e transformation on x (rather than \log_{10}) is directly interpretable as approximate proportional differences: with a coefficient of 0.06, a difference of 1 in x corresponds to an approximate 6% difference in y .
- **Square Root:** useful for compressing high values more mildly than by taking the log, but lack a clean interpretation; better for prediction tasks.

Log-Log Models: If we apply a log transformation to both features and outputs, we can interpret the coefficient as the expected proportional change in y per proportional change in x .

Regularization

Lasso, L_1 -norm: similar to siBayesian regression with Laplace priors on the slopes

Ridge, L_2 -norm: similar to Bayesian regression where slope priors are normally distributed with means equal to 0

Logistic Regression

$$\text{logit}(p) = \log\left(\frac{1}{1-p}\right) = \beta_0 + \beta^T x$$

$$\text{logistic}(z) = \text{logit}(z)^{-1} = \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z}$$

$$\max(\text{Log Likelihood}) = \min(\text{Cross-Entropy} / \text{Negative Log Likelihood})$$

Log Likelihood / Cross-Entropy

Let $y_i \in \{0, 1\}$

$$p_i = \frac{1}{1 + e^{-X_i \beta}}$$

$$\begin{aligned} \text{Likelihood, } L(\beta) &= \prod_{i=1}^n p_i^{y_i} (1 - p_i)^{1-y_i} \\ \text{Log Likelihood, } l(\beta) &= \sum_{i=1}^N \{y_i \log p_i + (1 - y_i) \log(1 - p_i)\} \\ &= \sum_{i=1}^N \log(1 - p_i) + \sum_{i=1}^N y_i \underbrace{(\log p_i - \log(1 - p_i))}_{\log\left(\frac{p_i}{1-p_i}\right) = \log(x) - \log(y)} \\ &= \sum_{i=1}^N \log(1 - p_i) + \sum_{i=1}^N y_i \underbrace{\log\left(\frac{p_i}{1 - p_i}\right)}_{X_i \beta} \\ &= \sum_{i=1}^N -\log(1 + e^{X_i \beta}) + \sum_{i=1}^N y_i (X_i \beta) \end{aligned}$$

Derivative of Negative Log Likelihood

$$\begin{aligned}\frac{\partial l(\beta)}{\partial \beta} &= - \left(- \sum_{i=1}^N \frac{1}{1 + e^{-X_i \beta}} X_i + \sum_{i=1}^N y_i X_i \right) \\ &= - \sum_{i=1}^N (y_i - p_i) X_i\end{aligned}$$

$$\begin{aligned}\text{In matrix form } \rightarrow & -\mathbf{X}^T(\mathbf{y} - \mathbf{p}) \\ &= -\mathbf{X}^T(\mathbf{y} - \mathbf{p}) + \underbrace{2\lambda\beta}_{L_2}\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} &= \mathbf{X}^T \mathbf{W} \mathbf{X} \\ &= \mathbf{X}^T \mathbf{W} \mathbf{X} + \underbrace{2\lambda \mathbf{I}}_{L_2}\end{aligned}$$

where $\mathbf{W} = \text{diag}(p_i(1 - p_i))$

Newton-Raphson / Iteratively Reweighted Least Squares (IRLS)

$$\beta^{n+1} = \beta^n - \left(\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} \right)^{-1} \frac{\partial l(\beta)}{\partial \beta}$$

Latent-Variable Formulation

There's no error term in the formulation, but we set one up: x_i has a latent variable U_i that has a logistic distribution that is IID. In Probit Regression, the latent variable $U_i \sim N(0, 1)$. Difference between linear and logistic/probit is that we can't estimate latent variables like we can measure the residuals in linear regression.

$$\begin{aligned}Y_i &= 1 \text{ if } X_i \beta + U_i > 0 \\ Y_i &= 0 \text{ if } X_i \beta + U_i < 0\end{aligned}$$

Bias / Variance Tradeoff

Mean Squared Error Decomposition:

$$\begin{aligned}&= \mathbb{E}(Y - \hat{Y})^2 \\ &= \text{Var}(Y) + \mathbb{E}[\text{Bias}^2(\hat{Y}) + \text{Var}(\hat{Y})] \\ &= \underbrace{\sigma^2}_{\text{irreducible error}} + \text{MSE}(\hat{Y})\end{aligned}$$

Kernels

- a similarity measure
- a dot product in some feature space, but we don't need to know the ϕ representation

Hebb's Rule

- update weights whether the prediction was right or wrong

$$\mathbf{w} = \sum_k \mathbf{y}^k \cdot \mathbf{x}^k$$

Perceptron

- loss function: $\max(0, -z)$
- update weights only when we misclassify

Large Margin SVM

- Hinge loss: $\max(0, 1 - z)$
- update weights when misclassify or within margin
- weights must be normalized to give the scale

Generative Models

Naive Bayes

All naive Bayes classifiers assume that the value of a particular feature is independent of any other feature, given the class variable. Advantages: very fast; robust to irrelevant features; very good in domains with many equally important features

Gaussian NB

- calculate mean and standard deviation of each feature for each class
- calculate the probability for each feature using the Gaussian PDF and multiply all probabilities together
- assign label with largest value as prediction

Multinomial NB

1. calculate priors for each class, $\hat{P}(c) = \frac{N_c}{N}$
2. calculate likelihood of a word occurring in a class (using add 1 smoothing): $\hat{P}(w|c) = \frac{\text{count}(w, c) + 1}{\text{count}(c) + |V|}$, where V is vocabulary size
3. Since the data are proportional, we don't need to divide by $P(\text{document})$


Gaussian classifier

- = centroid method
- generating model: draw y first, then draw x given y
- GC independence of features in a given class

Linear Discriminant Analysis

- a generalization of Gaussian classifier for cases where input variables are NOT independent, but all classes have same covariance matrix
- Ridge regression applied to classification is similar to LDA

Dan Jurafsky



$$\hat{P}(c) = \frac{N_c}{N}$$

$$\hat{P}(w|c) = \frac{\text{count}(w,c)+1}{\text{count}(c)+|V|}$$

Priors:

$$P(c) = \frac{3}{4}$$

$$P(j) = \frac{1}{4}$$

Conditional Probabilities:

P(Chinese | c) = (5+1) / (8+6) = 6/14 = 3/7

P(Tokyo | c) = (0+1) / (8+6) = 1/14

P(Japan | c) = (0+1) / (8+6) = 1/14

P(Chinese | j) = (1+1) / (3+6) = 2/9

P(Tokyo | j) = (1+1) / (3+6) = 2/9

45 P(Japan | j) = (1+1) / (3+6) = 2/9

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Choosing a class:

$$P(c|d5) \propto 3/4 * (3/7)^3 * 1/14 * 1/14 \approx 0.0003$$

$$P(j|d5) \propto 1/4 * (2/9)^3 * 2/9 * 2/9 \approx 0.0001$$

Figure 2: Multinomial Naive Bayes

Gaussian mixtures

- what if the single cluster per class hypothesis is violated? We can introduce multiple clusters per class (similar to RBF)

TF-IDF

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

- Term Frequency: the number of times that term t occurs in document d
- Inverse Document Frequency: $\log \left(\frac{N}{1 + |\{d \in D : t \in d\}|} \right)$
 - Numerator: N = number of docs
 - Denominator: number of documents where the term t appears, adjusted for division-by-zero errors

KNN

- curse of dimensionality: get more data, make better features, reduce dimensions
- $O(N)$ complexity, infeasible for large datasets
- K-d tree is a binary tree data structure for organizing a set of points in a K-dimensional space
- Hashing by Random Projections

Distance Measures

Mahalanobis: using covar matrix has the effect of decorrelating and normalizing features

$$(\mathbf{x}, \mathbf{y} \mid \Sigma) = \sqrt{(\mathbf{x} - \mathbf{y})\Sigma^{-1}(\mathbf{x} - \mathbf{y})}$$

Euclidean: special case of Mahalanobis with the identity matrix \mathbf{I} as the covariance matrix

$$(\mathbf{x}, \mathbf{y} \mid \Sigma) = \sqrt{(\mathbf{x} - \mathbf{y}) \cdot (\mathbf{x} - \mathbf{y})}$$

```
sqrt((x - y) %*% (x - y))
```

Cosine similarity: differs from Euclidean in that it doesn't depend on the length of the vectors \mathbf{x} and \mathbf{y} . Turned in a distance metric by taking $1 - \cos(\theta)$

$$\cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{\mathbf{x} \cdot \mathbf{y}}{\sqrt{(\mathbf{x} \cdot \mathbf{x})(\mathbf{y} \cdot \mathbf{y})}}$$

```
x %*% y / sqrt(x%*%x * y%*%y)
```

Pearson Correlation:

$$\rho_{X,Y} = \frac{(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{y} - \bar{\mathbf{y}})}{\|\mathbf{x} - \bar{\mathbf{x}}\| \|\mathbf{y} - \bar{\mathbf{y}}\|} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2} \sqrt{\sum (y_i - \bar{y})^2}} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

```
sum(x_diff*y_diff) / (sqrt(sum(x_diff^2)) * sqrt(sum(y_diff^2)))
# or
mean(x_diff*y_diff) / (sd(x) * sd(y))
```

String Distance

Hamming Distance: Measures the minimum number of substitutions required to change one string into the other between two strings of equal length.

Levenshtein Distance: Counts the minimum number of operations needed to transform one string into the other, where an operation is defined as an insertion, deletion, or substitution of a single character, or a transposition of two adjacent characters.

Jacard Similarity: The size of the intersection of two sample sets divided by the size of the union.

Decision Tree

Measure best split by:

$$\text{Information Gain} = \text{Entropy}(\text{parent}) - [\text{Weighted Average Entropy}(\text{children})]$$

Entropy is a measure of uncertainty:

$$\text{Entropy} = - \sum_{i=1}^d p_i \log p_i$$

- Stopping criteria: max tree depth, min # of points per node, tree complexity, validation error monitoring
- Pruning can improve performance: prune tree back to where it gives more accurate results on the test data
- Deep trees have high variance (pruning controls this), low bias

Random Forest

- Averaging: randomize each model
- Bagging (Bootstrap aggregating): randomize the dataset fed to a tree
- More trees reduce variance

K-means Clustering

Initialize k cluster centers (e.g., by randomly choosing k points among our datapoints). Then we repeatedly:

1. E-step (in EM algorithm): Assign each datapoint to the nearest cluster center (we pretend that we know the cluster centers, and based off this pretense, we guess which cluster each point belongs to)
2. M-step: Update all the cluster centers: for each cluster i , take the mean over all points currently in the cluster, and update cluster center i to be this mean (we do the reverse: we pretend that we know which cluster each point belongs to, and then try to guess the cluster centers)
3. Repeat steps 1 and 2 until cluster assignments stop changing

Finding K

- [The Elbow Method](#): looks at the percentage of variance explained as a function of the number of clusters. Choose a number of clusters so that adding another cluster doesn't give much better modeling of the data. Percentage of variance explained is the ratio of the between-group variance to the total variance, also known as an F-test. A slight variation of this method plots the curvature of the within group variance.
- [The Silhouette Method](#): measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from -1 to 1. Can be calculated with any distance metric, such as the Euclidean distance or the Manhattan distance.
- Gap statistic:

K-medoids

Cluster centers are existing coordinates; can be used with a dissimilarity matrix

Gaussian Mixture Models

k-means has a hard notion of clustering: point X either belongs to cluster C or it doesn't. But sometimes we want a soft notion instead: point X belongs to cluster C with probability p (according to a Gaussian kernel). This is exactly like k-means in the EM formulation, except we replace the binary clustering formula with Gaussian kernels.

Linear Algebra

Definitions

Rank: The rank of A is the number of linearly independent rows in A , which is equal to the number of linearly independent columns in A . A matrix is said to have *full rank* if its rank equals the largest possible for a matrix of the same dimensions, which is the min of the number of rows and columns. If A is a square matrix, then A is invertible if and only if A has rank n .

Matrix Inverse: can only invert a matrix if it is nonsingular square matrix of full rank.s

$$AA^{-1} = A^{-1}A = I$$

Singular Matrix: a square matrix that is not invertible; singular if and only if $\det(A) = 0$.

Transpose

$$\begin{aligned}(A^T)^T &= A \\ (AB)^T &= B^T A^T \\ (A + B)^T &= A^T + B^T\end{aligned}$$

Multiplication

Matrix · Matrix

$$B = AC \rightarrow b_{ij} = \sum_{k=1}^n a_{ik}c_{kj}$$

Vector · Vector

$$\begin{aligned}x^T y &= \sum_{i=1}^n x_i y_i \\ \text{Euclidean norm, } \|x\|_2 &= \sqrt{x^T x} = \left(\sum_{i=1}^n |x_i|^2 \right)^{1/2} \\ \text{1-norm, } \|x\|_1 &= \sum_{i=1}^n |x_i|\end{aligned}$$

Moore-Penrose Pseudoinverse

$$A^+ = (A^T A)^{-1} A^T$$

SVD

$$\underbrace{X}_{N \times p} = \underbrace{U}_{N \times p} \underbrace{\Sigma}_{p \times p} \underbrace{V^T}_{p \times p}$$

Reduced SVD ($m \geq n$)

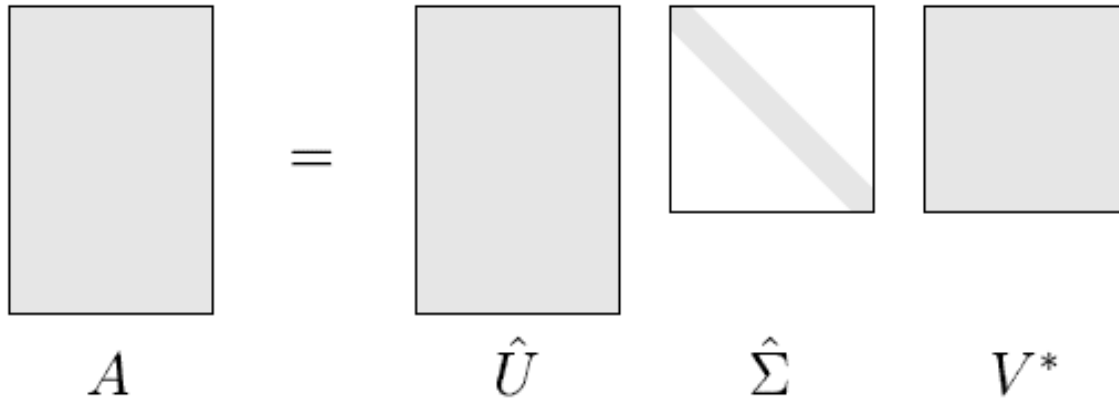


Figure 3: Singular Value Decomposition

- \mathbf{U} and \mathbf{V} are orthogonal matrices of left- and right-singular vectors along their columns
- $\mathbf{\Sigma}$ is matrix of singular values with diagonal elements $\Sigma_1 \geq \Sigma_2 \geq \dots \geq \Sigma_p \geq 0$
- The non-zero singular values of \mathbf{X} are the square roots of the eigenvalues $\mathbf{X}^T \mathbf{X}$ and $\mathbf{X} \mathbf{X}^T$
- The columns of $\mathbf{U} \mathbf{\Sigma}$ are called the principal components of \mathbf{X}

PCA

1. subtract mean from each column of \mathbf{X}
2. (sometimes) scale dimensions by its variance
3. compute covariance matrix, $\mathbf{\Sigma} = \mathbf{X}^T \mathbf{X}$
4. square roots of eigenvalues of $\mathbf{\Lambda}$ are the variance explained by the principal components (eigenvectors)

$$\underbrace{\mathbf{\Sigma}}_{p \times p} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^T$$

- \mathbf{V} is an orthogonal matrix composed of eigenvectors of \mathbf{X}
- $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues