Introduction to Kernel-Based Machine Learning

W. Evan Johnson, Ph.D.
Professor, Division of Infectious Disease
Director, Center for Data Science
Rutgers University – New Jersey Medical School

2024-05-21

GitHub Page

Here is a link to the GitHub Repository for this tutorial:

https://github.com/wevanjohnson/2024_05_RRDS_ML_Workshop/

Please download, clone, or fork this directory to get the materials!

Tools to install

Here is the list of R packages to install for this tutorial:

- Data management and visualization: `tidyverse`, `DT`, `gridExtra`, and `dslabs` (Rafa Irizarry's datasets for his Data Science book!).

```
install.packages(c(tidyverse, DT, gridExtra, dslabs))
```

- Machine learning: `caret`, `e1071`, `rpart`, `neuralnet`

```
install.packages(c(caret, e1071, rpart, neuralnet))
```

And download (using R) this dataset (from the Elements of Statistical Learning book) and put it in your working directory:

```
download.file("http://www-stat.stanford.edu/
              ~tibs/ElemStatLearn/datasets/ESL.mixture.rda",
              destfile='ESL.mixture.rda')
```

Agenda

- Introduction to Machine Learning Terminology
- Definition of features, labels, data preparation, input, output, etc.
- Overview of Machine Learning Methods
  - Data preparation
  - Supervised and unsupervised learning
  - Model training and testing
  - Model evaluation / performance metrics
- Kernel-based Methods
  - Conceptual understanding
- Decision Trees
- Neural Networks
- Code + case studies are integrated throughout

Formal definitions

Machine learning is a computer's way of learning from examples, and it is one of the most useful tools we have for the construction of *artificially intelligent* systems

Artificial intelligence is a term used when machines (or computers) can mimic functions that humans can do.

- Example: Learning and problem solving

Formal definitions

An **Algorithm** is a sequence of actions that are "self-contained"

- Effective method for calculation
  - Finite steps/instructions
  - When applied, it produces a correct answer
  - The instructions need to be followed
  - In principle, it can be done by a "human"

Formal definitions

A **Computer Algorithm** is a sequence of actions that are "self-contained"

- Effective method for calculation
  - Finite steps/instructions
  - When applied, it produces a correct answer
  - The instructions need to be followed
  - In principle, it can be done by a "computer"

Formal definitions

- Computer Code:
  - Human readable text
  - Fully executable description of a software system
- What's in the data?
  - Datum: "(thing) given"
  - Data is useful only when it has been analyzed

Supervised vs unsupervised learning

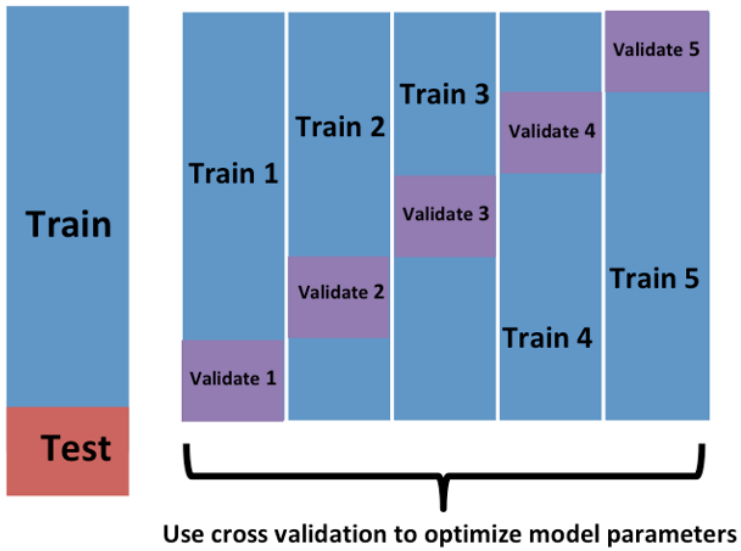
Machine learning algorithms are generally classified into two categories:

1. **Supervised:** Outcomes used to create the predictor, e.g., classification.
2. **Unsupervised:** Don't know outcomes, rather interestedin discovering groups, e.g., clustering.

Formal definitions

Training and Test sets: We usually split our dataset into two parts, one for model creation (train) and one for validation (test)

Cross-validation: Iterative retraining of the predictor on multiple partitions of training set



Formal definitions

Confusion matrix: tabulates each combination of predicted and actual values:

	Actually Positive	Actually Negative
Predicted positive	True positives (TP)	False positives (FP)
Predicted negative	False negatives (FN)	True negatives (TN)

Overfitting: Dangerously over-optimistic assessments—this is a *big problem* in machine learning

The caret package in R

The `caret` package in R has several useful functions for building and assessing machine learning methods.

- *Examples:*

Regularization in Machine Learning

In regression analysis, the features are estimated using coefficients while modeling. In small sample sizes or noisy data coefficient estimates could be anecdotally incorrect (e.g., overfitting) or innacurate.

If the estimates can be restricted, penalized, or shrunk towards zero, then the impact of insignificant features might be reduced and would prevent models from high variance with a stable fit.[1]

[1] Adapted from:
https://www.analyticssteps.com/blogs/l2-and-l1-regularization-machine-learning

Regularization in Machine Learning

Regularization is the most used technique to penalize complex models in machine learning, it is deployed for reducing overfitting (or, contracting generalization errors) by putting small network weights into the model (adding a small amount of biad). Also, it enhances the performance of models for new inputs.[2]

Examples of regularization in machine learning, include:

- Regression: Penalizing coefficients to create parsimonious models (variable selection)
- K-means: Restricting the segments for avoiding redundant groups.
- Neural networks: Confining the complexity (weights) of a model.
- Random forests: Reducing the depths of tree and branches (new features)
- Neural networks: Confining the complexity (weights) of a model.

[2]Source:
https://www.analyticssteps.com/blogs/l2-and-l1-regularization-machine-learning

Ridge regression

Ridge regression **regularizes** (shrinks) coefficients by imposing a penalty on their size. The ridge coefficients minimize a penalized sum of squared error:

$$\hat{\beta}^{ridge} = \inf_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\},$$

where $\lambda \geq 0$ is a parameter that controls the shrinkage. The larger the value of λ the more shrinkage (towards 0).

Ridge regression

Or in matrix form, ridge regression minimizes:

$$\hat{\beta}^{ridge} = \inf_{\beta} \left\{ (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) + \lambda \beta^T \beta \right\}.$$

With a little work, the ridge regression solution can be shown to be:

$$\hat{\beta}^{ridge} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_N)^{-1} \mathbf{X}^T \mathbf{y}$$

Ridge regression

The regularization for ridge regression, $\lambda\beta^T\beta$, is usually denoted as an **L2 regularization** or **L2 penalty**, as it adds a penalty which is equal to the square of the magnitude of coefficients. Both Ridge regression and Support Vector Machines (SVMs) implement this method.

L2 regularization can deal with multicollinearity problems (independent variables are highly correlated) through constricting the coefficient while keeping all the variables in a model.

However, L2 regularization is not an effective method for selecting relevant predictors (or removing redundant parameters). We will later use a **L1 regularization** for this purpose.

Ridge regression: a Bayesian perspective

Ridge regression also has a clear Bayesian interpretation. It can be shown that the Ridge penalty can be interpreted as a 'zero' prior (Normal prior with zero mean), and the λ is related to the variance of the prior.

Lasso regression

Lasso (Least Absolute Shrinkage and Selection Operator) regression also **regularizes** coefficients by imposing a penalty on their size, but it uses an **L1** penalty. The lasso coefficients minimize the following cost function:

$$\hat{\beta}^{lasso} = \inf_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \alpha \sum_{j=1}^{p} |\beta_j| \right\},$$

where $\alpha \geq 0$ is a parameter that controls the shrinkage. The larger the value of α the more shrinkage (towards 0).

Lasso regression

Notice the similarity to the ridge regression problem: the L2 ridge penalty $\sum_{j=1}^{p} \beta_j^2$ is replaced by the L1 lasso penalty $\sum_{j=1}^{p} |\beta_j|$.

This latter constraint makes the solutions nonlinear in the y_i, and there is no closed form expression for the lasso as was the case in ridge regression.

Because of the nature of the constraint, making α sufficiently small will cause some of the coefficients to be exactly zero. Thus the lasso does a kind of continuous subset selection, or conducts a **variable selection**.

Lasso vs Ridge regression

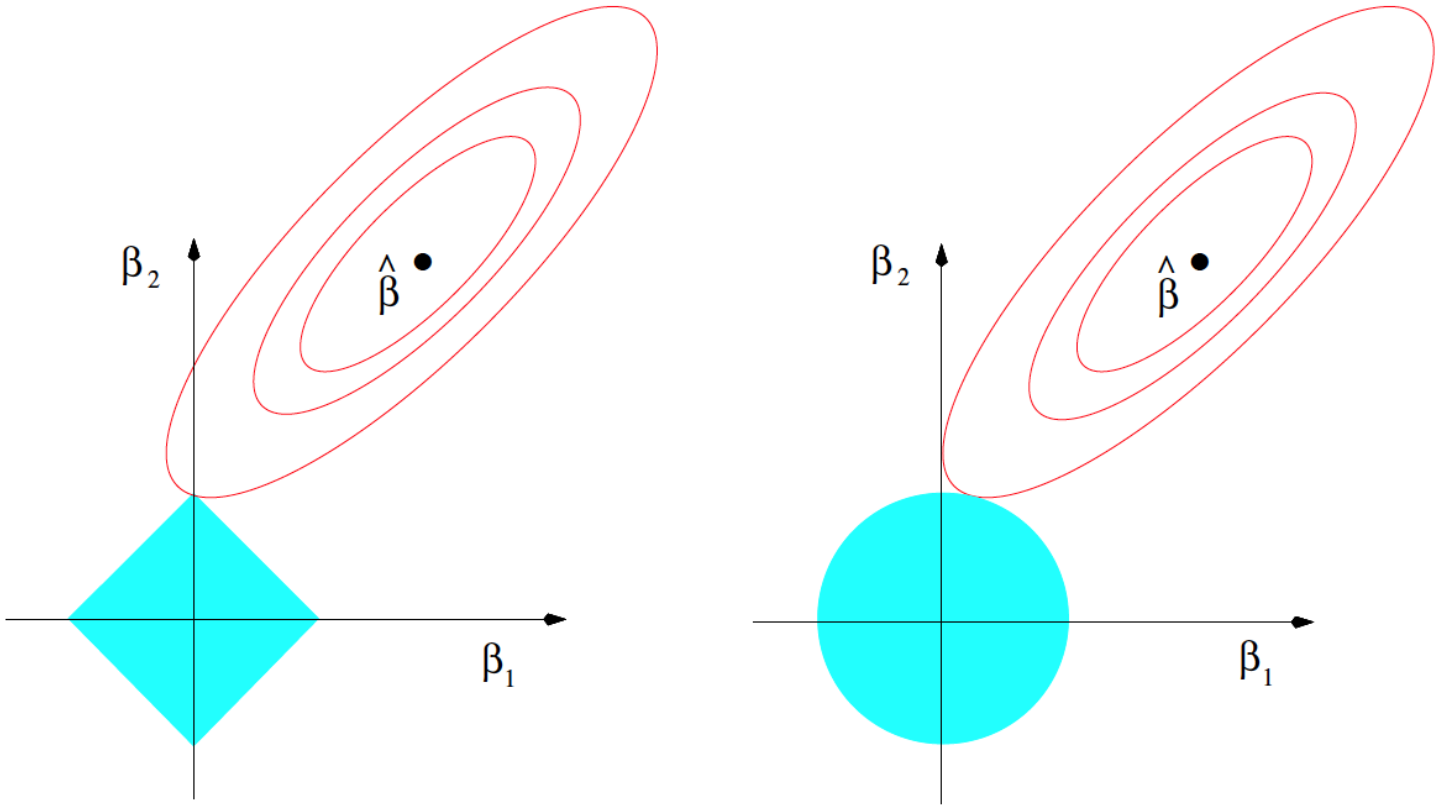


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are the contours of the least squares error function.*

(Elements of Statistical Learning, Hastie, Tibshirani, Friedman, by Springer)

Lasso vs Ridge regression

S.No	L1 Regularization	L2 Regularization
1	Panelizes the sum of absolute value of weights.	penalizes the sum of square weights.
2	It has a sparse solution.	It has a non-sparse solution.
3	It gives multiple solutions.	It has only one solution.
4	Constructed in feature selection.	No feature selection.
5	Robust to outliers.	Not robust to outliers.
6	It generates simple and interpretable models.	It gives more accurate predictions when the output variable is the function of whole input variables.
7	Unable to learn complex data patterns.	Able to learn complex data patterns.
8	Computationally inefficient over non-sparse conditions.	Computationally efficient because of having analytical solutions.

(https://www.analyticssteps.com/blogs/l2-and-l1-regularization-machine-learning)

Elastic net regularization

Which should I choose? Ridge or Lasso? Well, why do I have to choose!

Instead use the **Elastic Net** that minimizes:

$$\hat{\beta}^{elastic\ \ net} = \inf_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \alpha \sum_{j=1}^{p} |\beta_j| + \lambda \sum_{j=1}^{p} \beta_j^2 \right\},$$

for some $\alpha \geq 0$ and $\lambda \geq 0$.

The quadratic penalty term makes the loss function strongly convex, and it therefore has a unique minimum. The elastic net method includes OLS, Lasso, and Ridge regression by setting either $\alpha = 0$, $\lambda = 0$, or both to 0.

Regression Regularization in R: glmnet

We can use the **glmnet** package to apply regularization in R:

```
install.packages("glmnet")
```

The default model used in the package is the "least squares" regression model and glmnet actually optimizes:

$$\hat{\beta}^{elastic\ \ net} = \inf_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \left(\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \right) \right\},$$

with $\alpha = 1$ as a default (so Lasso!).

Kernel-based machine learning

Kernel machines are a class of methods for pattern analysis that use linear classifiers to solve nonlinear problems.

Kernel methods only require a user-specified kernel, i.e., a similarity function over all pairs of data points computed using inner products.

In contrast, many other algorithms require the explicit transformation of the raw data.

Kernel based machine learning

Algorithms that operate with kernels include:

- Kernel perceptron
- Support-vector machines (SVM)
- Gaussian processes
- Principal components analysis (PCA)
- Canonical correlation analysis
- Ridge regression
- Spectral clustering
- Linear adaptive filters

Session Info

```
sessionInfo()
```

```
## R version 4.4.0 (2024-04-24)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.2.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK version 3
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Denver
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.4.0    fastmap_1.1.1     cli_3.6.2         tools_4.4.0
##  [5] htmltools_0.5.8.1 rstudioapi_0.16.0 yaml_2.3.8        rmarkdown_2.26
##  [9] knitr_1.46        xfun_0.43         digest_0.6.35     rlang_1.1.3
## [13] evaluate_0.23
```