Department of Computer Engineering
University of Peradeniya

**Machine Learning Laboratory: Regression**

Guest Lecture
Professor Mahesan Niranjan
University of Southampton
February 2022

## Objective

- Implementing linear regression
- Regularization using quadratic and sparsity-inducing penalties
- Implementing sparse regression on a realistic problem in chemoinformatics

## 1 Linear Least Squares Regression:

We will work with the `Diabetes` dataset from the `UCI Machine Learning` repository [1] taken from the package `sklearn`. Load the data and inspect the features and targets. It is usually a good idea to plot a few histograms of the targets and pair-wise scatters of the features in any new problem you are tasked to solve.

- Implement a linear predictor that is solved by the pseudo-inverse method:

  $$\boldsymbol{w} = \left(X^t X\right)^{-1} X^t \boldsymbol{t},$$

  where $X$ is the $N \times d$ input matrix and $\boldsymbol{t}$ is the $N \times 1$ vector of responses (or targets).

- Solve the same problem using the linear model from `sklearn` and compare the results.

## 2 Regularization

Tikhonov regularization (or $L_2$) minimizes the mean squared error with a quadratic penalty on the weights:

$$\min_{\boldsymbol{w}} ||\boldsymbol{t} - X\boldsymbol{w}||_2^2 + \gamma\, ||\boldsymbol{w}||_2^2$$

Derive and implement a regularized regression. Show, using two bar graphs of the weights side by side to the same scale, how the two solutions differ.

## 3 Sparse Regression

$L_1$ regularization is a method for achieving *sparse* solutions [2]. It minimizes:

$$\min_{\boldsymbol{w}} ||\boldsymbol{t} - X\boldsymbol{w}||_2^2 + \gamma\, ||\boldsymbol{w}||_1$$

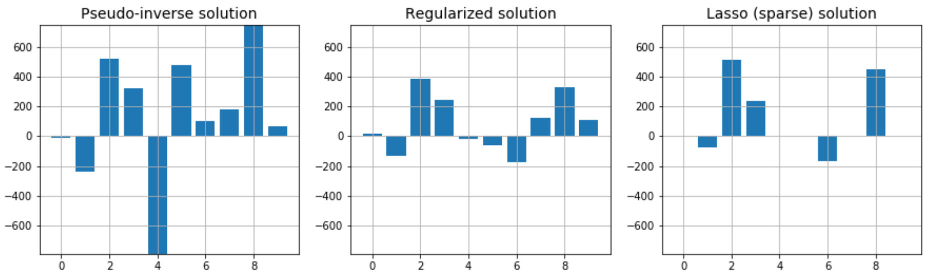


Figure 1: Solutions of Linear and Regularized Regressions

We will use the `sklearn` package for implementing its solution. For the Diabetes problem considered above, solve the `lasso` problem and plot the resulting weights as a bar graph. Observe how the number of non-zero weights change with the regularization parameter $\gamma$. Your comparisons should look similar to Fig. 1. In each of these cases, compare the prediction errors. In the case of the sparse regression, would you say the features with nonzero weights are more meaningful (to answer, you have to find the source of the data and look at the variables)?

### Regularization Path

When implementing the *lasso* it is convenient to study the regularization path (Fig. 2, Image taken from `https://scikit-learn.org/stable/auto_examples/linear_model/plot_lasso_lars.html`) Implement and study the regularization path for the six-variable

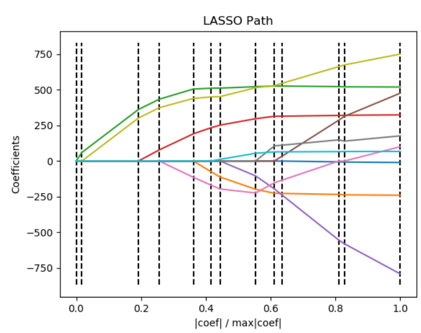


Figure 2: Regularization Path: How regression coefficients change with hyperparameter.

illustrative example considered in [2].

## 4 Solubility Prediction:

We will now look at a large problem of predicting solubility of chemical compounds from features derived from their molecular structure. Predicting function from structural variables is an important problem because it is easy to define and synthesize small chemical compounds, but very expensive to test them experimentally. Hence the step known as *in silico* screening is increasingly popular. The dataset we will use is from Huuskonen *et al.* [3] and the problem has also been considered recently in Pirashvili *et al.* [4] using more sophisticated machinery. Have a skim-read through the introductory and results sections of these papers.

Data used in [3] with several additional features and more compounds is available in the excel spread sheet `Husskonen_Solubility_Features.xlsx`.

- Load the data, split into training and test sets, implement a linear regression and plot the predicted solubilities against the true solubilities on the training and test sets. To facilitate comparison, draw the two scatter plots side by side to the same scale on both axes.

- Implement a `lasso` regularized solution and plot graphs of how the prediction error (on the test data) and the corresponding number of non-zero coefficients change with increasing regularization.

- If you were to select the top ten features to predict solubility, what would they be[1]? How good is the prediction accuracy with these slected features when compared to using all the features and a quadratic regularizer?

- Are you able to make any comment comparing your results to those claimed in [3] or [4]?

## Report

Write a short report of `no more than four pages`, summarising your work.

## Appendix: Snippets of Code

1. Linear regression on diabetes dataset

[1] Of course, we will not know enough chemistry to interpret these, but in a real-world setting, we will be working with a friend in the School of Chemistry!

```
from sklearn import datasets
from sklearn.linear_model import LinearRegression

# Load data, inspect and do exploratory plots
#
diabetes = datasets.load_diabetes()

X = diabetes.data
t = diabetes.target

# Inspect sizes
#
NumData, NumFeatures = X.shape
print(NumData, NumFeatures)     # 442 X 10
print(t.shape)                  # 442

# Plot and save
#
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))
ax[0].hist(t, bins=40)
ax[1].scatter(X[:,6], X[:,7], c='m', s=3)
ax[1].grid(True)
plt.tight_layout()
plt.savefig("DiabetesTargetAndTwoInputs.jpg")
```

2. Comparing pseudo-inverse solution to `sklearn` output

```
# Linear regression using sklearn
#
lin = LinearRegression()
lin.fit(X, t)
th1 = lin.predict(X)

# Pseudo-incerse solution to linear regression
#
w = np.linalg.inv(X.T @ X) @ X.T @ t
th2 = X @ w

# Plot predictions to check if they look the same!
#
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(10,5))
ax[0].scatter(t, th1, c='c', s=3)
```

3. Tikhanov (quadratic) Regularizer

```
gamma = 0.5
wR = np.linalg.inv(X.T @ X + gamma*np.identity(NumFeatures)) @ X.T @ t

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(8,4))
ax[0].bar(np.arange(len(w)), w)

plt.savefig("LeastSquaresAndRegularizedWeights.jpg")
```

4. Sparsity inducing (lasso) regularizer

```
from sklearn.linear_model import Lasso
ll = Lasso(alpha=0.2)
ll.fit(X, t)
th_lasso = ll.predict(X)

fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(15,4))
ax[0].bar(np.arange(len(w)), w)
#
#...
#
plt.savefig("solutions.png")
```

5. Lasso Regularization path on a synthetic example (Set up data):

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import lasso_path
from sklearn import datasets

# Synthetic data:
# Problem taken from Hastie, et al., Statistical Learning with Sparsity
# Z1, Z2 ~ N(0,1)
# Y = 3*Z1 -1.5*Z2 + 10*N(0,1) Noisy response
# Noisy inputs (the six are in two groups of three each)
# Xj= Z1 + 0.2*N(0,1) for j = 1,2,3, and
# Xj= Z2 + 0.2*N(0,1) for j = 4,5,6.

N = 100
y = np.empty(0)
X = np.empty([0,6])
for i in range(N):
    Z1= np.random.randn()
    Z2= np.random.randn()
    y = np.append(y, 3*Z1 - 1.5*Z2 + 2*np.random.randn())
    Xarr = np.array([Z1,Z1,Z1,Z2,Z2,Z2])+ np.random.randn(6)/5
    X = np.vstack ((X, Xarr.tolist()))
```

6. Lasso Regularization path on a synthetic example (Regression and paths):

```
# Compute regressions with Lasso and return paths
#
alphas_lasso, coefs_lasso, _ = lasso_path(X, y, fit_intercept=False)

# Plot each coefficient
#
fig, ax = plt.subplots(figsize = (8,4))
for i in range(6):
    ax.plot(alphas_lasso, coefs_lasso[i,:])

ax.grid(True)
ax.set_xlabel("Regularization")
ax.set_ylabel("Regression Coefficients")
```

7. Predicting Solubility of Chemical Compounds

```
sol = pd.read_excel("Husskonen_Solubility_Features.xlsx", verbose=False).dropna()
print(sol.shape)

colnames = sol.columns
print(colnames)

t = sol["LogS.M."].to_numpy()
X = sol[colnames[5:len(colnames)-1]].to_numpy()
N, p = X.shape

# Split data into training and test sets
#
from sklearn.model_selection import train_test_split
X_train, X_test, t_train, t_test = train_test_split(X, t, test_size=0.3)

# Regularized regression
#
gamma = 0.1
w = np.linalg.inv(X_train.T @ X_train + gamma*np.identity(p)) @ X_train.T @ t_train
th_train = X_train @ w
th_test  = X_test @ w

# Plot training and test predictions
#
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(10,4))
ax[0].scatter(t_train, th_train, c='m', s=3)
ax[0].grid(True)
ax[0].set_title("Training Data", fontsize=14)

ax[1].scatter(t_test, th_test, c='m', s=3)
ax[1].grid(True)
ax[1].set_title("Test Data", fontsize=14)

plt.savefig("SolubilityPrediction.jpg")


# Over to you for implementing Lasso
#
```

## References


[1] K. Bache and M. Lichman, "UCI machine learning repository." http://archive.ics.uci.edu/ml, 2013.

[2] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC, 2015.

[3] J. Huuskonen, M. Salo, and J. Taskinen, "Aqueous solubility prediction of drugs based on molecular topology and neural network modeling," *Journal of Chemical Information and Computer Sciences*, vol. 38, no. 3, pp. 450–456, 1998.

[4] M. Pirashvili, L. Steinberg, B. G. F., M. Niranjan, J. G. Frey, and J. Brodzki, "Improved understanding of aqueous solubility modeling through topological data analysis," *Journal of Cheminformatics*, vol. 10, no. 1, p. 54, 2018.



Mahesan Niranjan

February 2022