MIE 1624

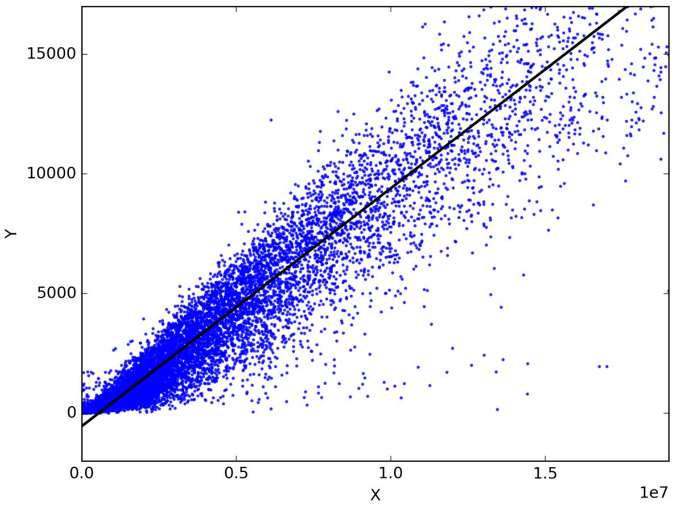
In class presentation: Linear Regression

Group 7:

Akshay Vishwakarma
Haohan Li
Parth Dave
Rui Fang
Xi Sun

What is Linear Regression ?

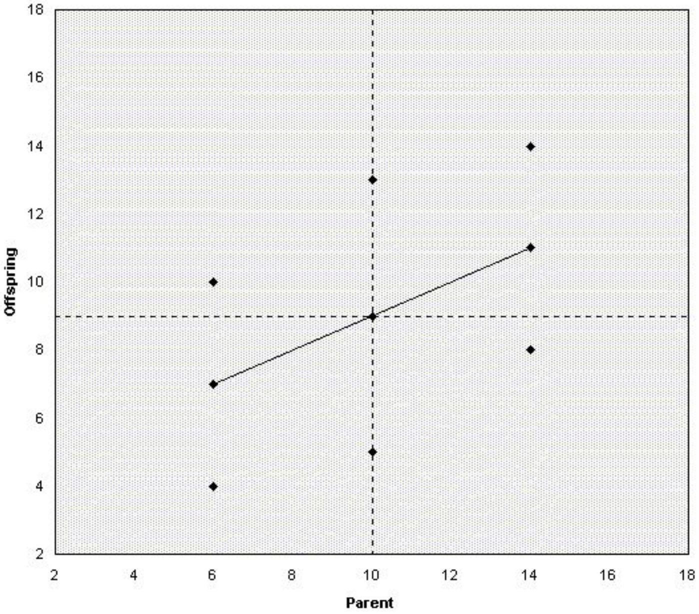
- Prediction tool that helps in predicting an outcome (dependent) variable
- Also helps identify which variable are significant predictors of the outcome variable
    - Indicated by the magnitude and sign of the variable



History behind Linear Regression

Originally invented by Sir Francis Galton in 1875 for the Genetic study.

- First model was pea size study parent pea plant and the offspring
- Later the hereditary problem study used the same model



Objective of Linear Regression

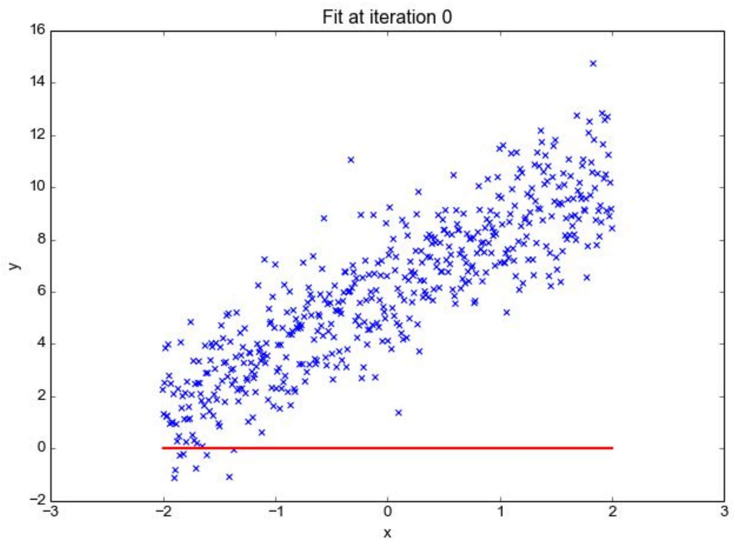
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

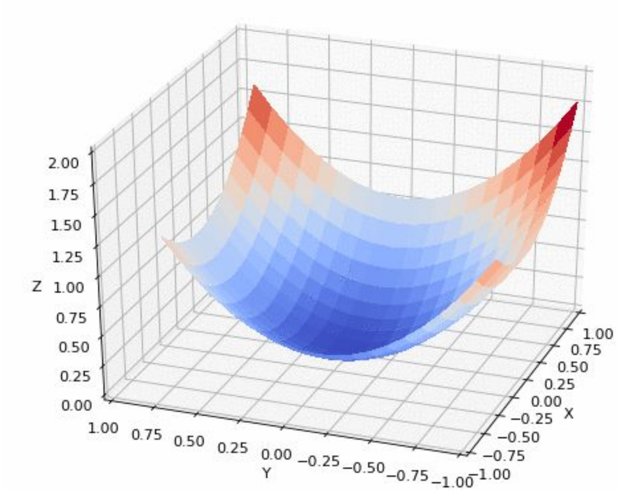
Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Linear Regression in action

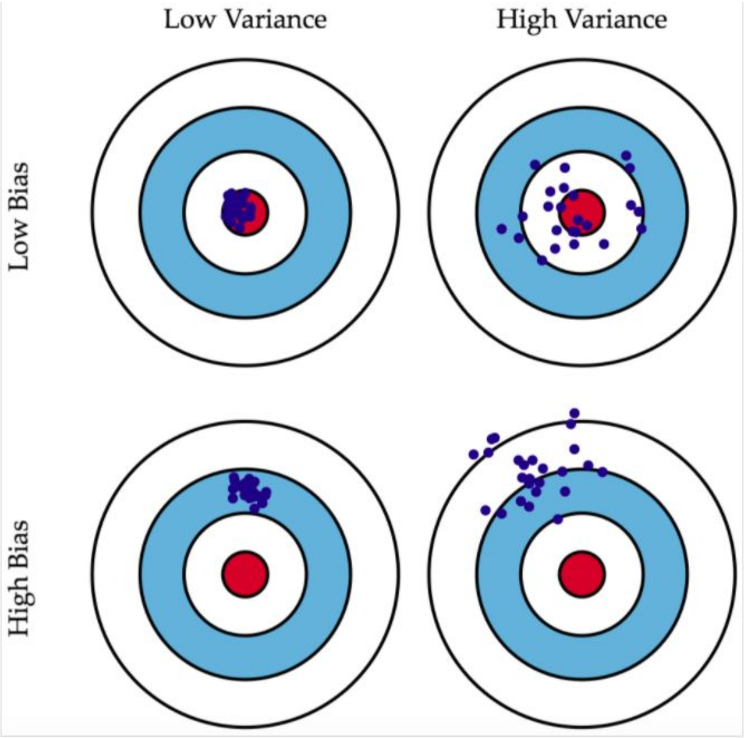




Fitting the linear model

Cost function minimization

Bias-Variance tradeoff



Bias-Variance tradeoff

$$y = f(x) + \varepsilon$$

$$\begin{aligned}
\mathrm{E}\left[(y - \hat{f})^2\right] &= \mathrm{E}[y^2 + \hat{f}^2 - 2y\hat{f}] \\
&= \mathrm{E}[y^2] + \mathrm{E}[\hat{f}^2] - \mathrm{E}[2y\hat{f}] \\
&= \mathrm{Var}[y] + \mathrm{E}[y]^2 + \mathrm{Var}[\hat{f}] + \mathrm{E}[\hat{f}]^2 - 2f\,\mathrm{E}[\hat{f}] \\
&= \mathrm{Var}[y] + \mathrm{Var}[\hat{f}] + (f^2 - 2f\,\mathrm{E}[\hat{f}] + \mathrm{E}[\hat{f}]^2) \\
&= \mathrm{Var}[y] + \mathrm{Var}[\hat{f}] + (f - \mathrm{E}[\hat{f}])^2 \\
&= \sigma^2 + \mathrm{Var}[\hat{f}] + \mathrm{Bias}[\hat{f}]^2
\end{aligned}$$

Overfitting

data

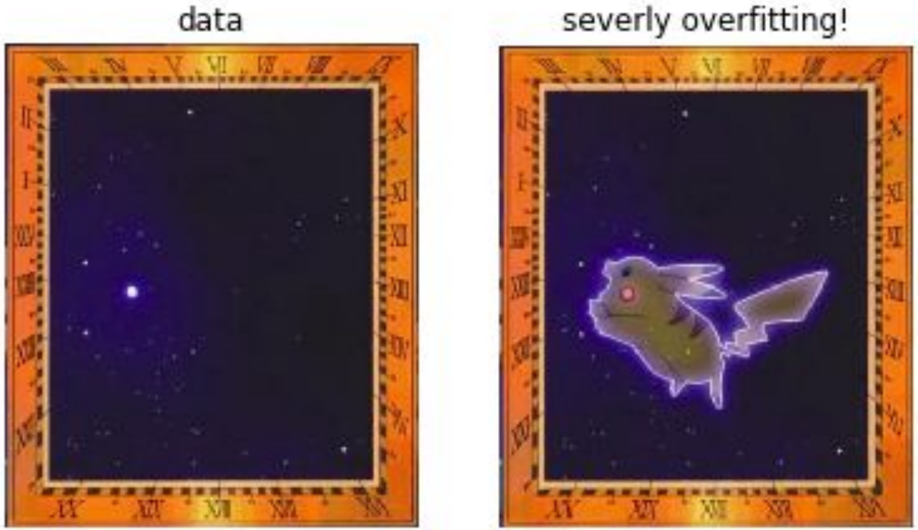
data

data

Overfitting

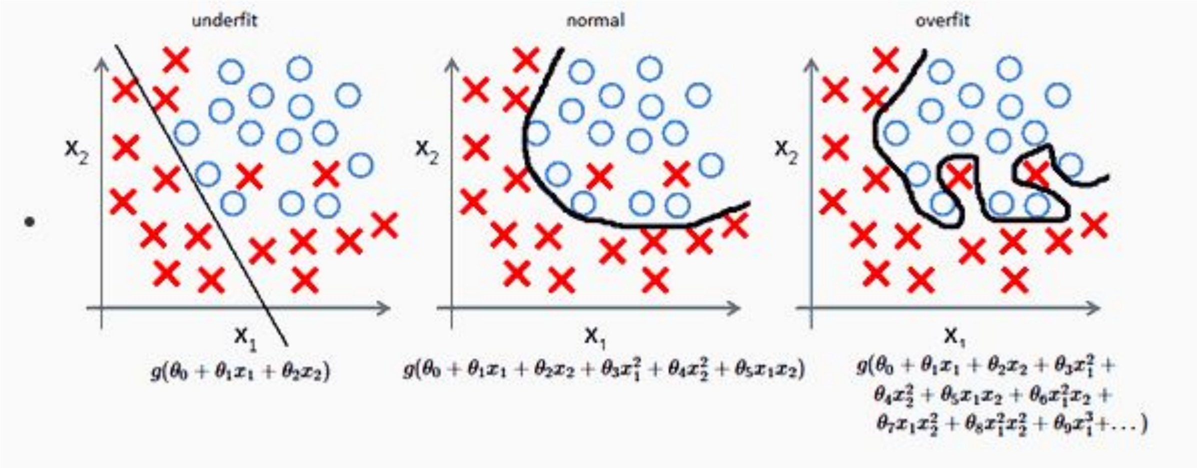
Overfitting

Overfitting



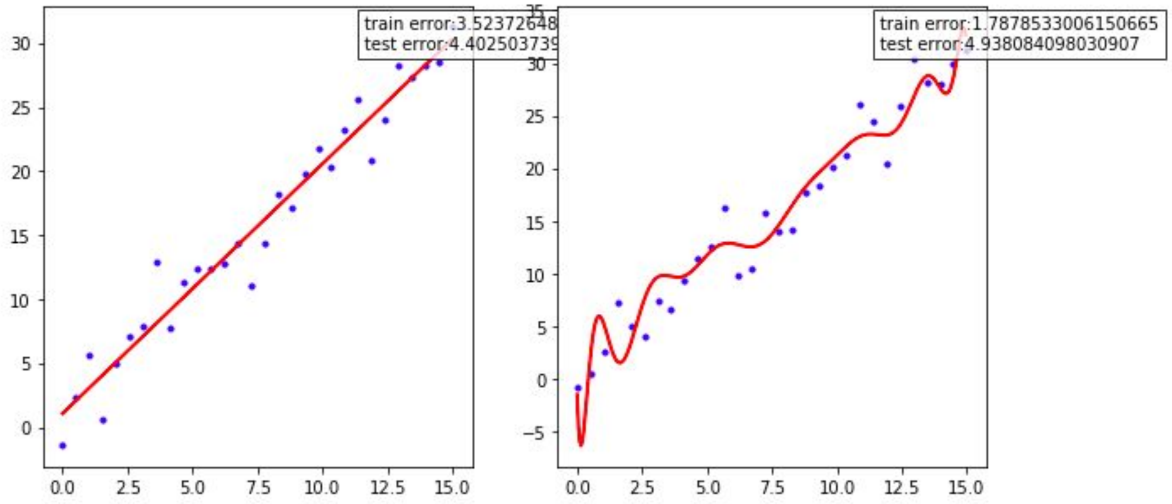
Overfitting

On Euclidean Space

- 

Overfitting

```python
#real model
f=lambda x:2*x+x**0.01

#generate test data
x_train = np.linspace(0, 15, 30)
y_train = np.vectorize(f)(x_train) + 2 * np.random.normal(size=x_train.size)#y=f(x)+noise

#generate test data
x_test = np.linspace(0, 15, 30)
y_test = np.vectorize(f)(x_test) + 2 * np.random.normal(size=x_test.size)#y=f(x)+noise

prop = np.polyfit(x_train,y_train,1)
prop_   = np.poly1d(prop)
overf   = np.polyfit(x_train, y_train, 15)
overf_  = np.poly1d(overf)

Plotter.train_error=list()
Plotter.test_error=list()
Plotter.train_error.append(np.mean((y_train - np.polyval(prop, x_train)) ** 2))
Plotter.train_error.append(np.mean((y_train - np.polyval(overf, x_train)) ** 2))
Plotter.test_error.append(np.mean((y_test - np.polyval(prop, x_test)) ** 2))
Plotter.test_error.append(np.mean((y_test - np.polyval(overf, x_test)) ** 2))

thisPlotter = Plotter([prop_,x_train,y_train],[overf_,x_test,y_test])
```



The Problem of Overfitting

Too many complicated predictors in the model

- **Misleading statistics:** R-squared values, p-values, regression coefficients

- **Prediction error:** Not reflecting the overall population

Avoiding Overfitting: Regularization

Adding a complexity penalty to the loss function

1. **L1 regularization (Lasso):** sum of the weights

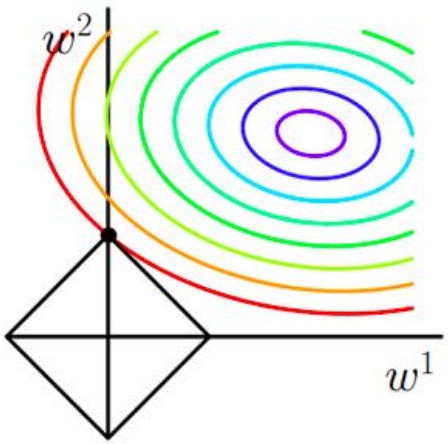
$$\min_{\boldsymbol{\beta}} \|\boldsymbol{X}\beta - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1$$

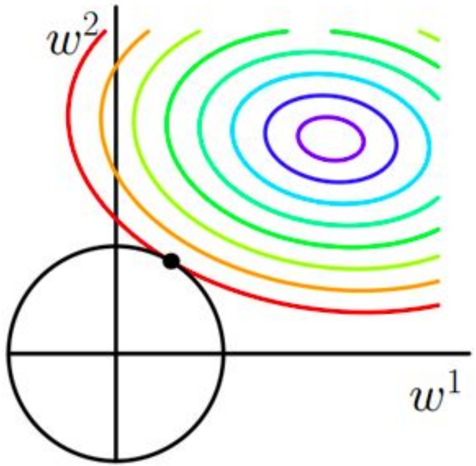
2. **L2 regularization (Ridge):** sum of the square of the weights

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{X}\beta - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2$$

Avoiding Overfitting: Regularization

On Euclidean Space





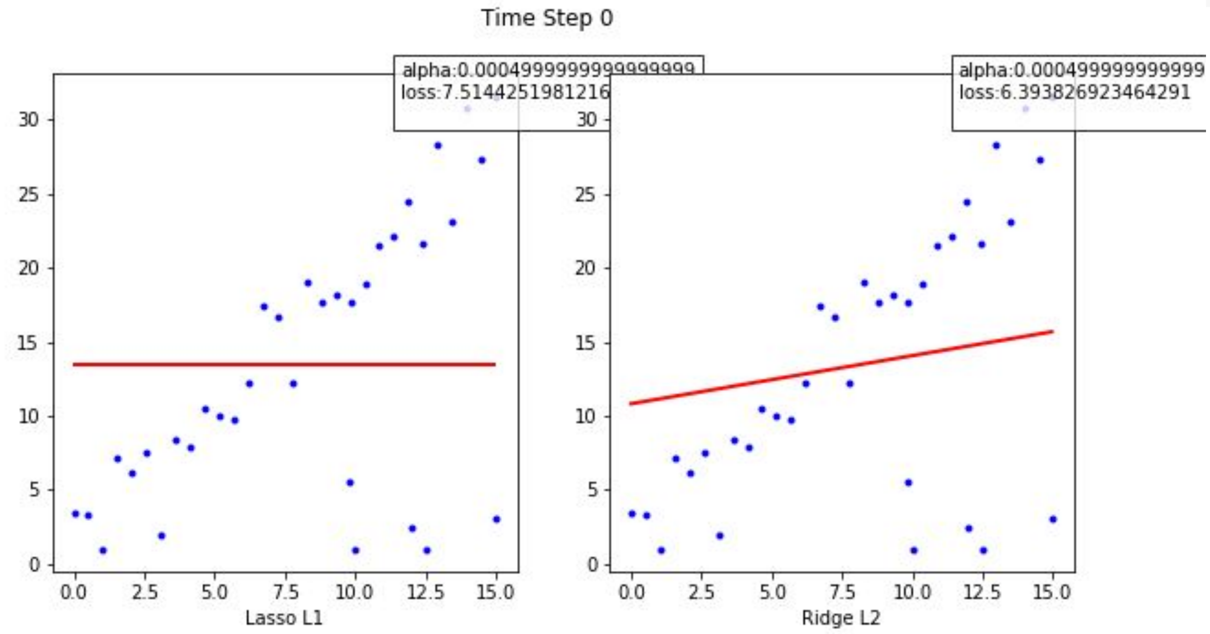
Avoiding Overfitting: Regularization

Bayesian Explanation (from MLE to MAP)

$$\underbrace{P(\theta \mid \mathcal{D})}_{\text{posterior}} \propto \underbrace{P(\mathcal{D} \mid \theta)}_{\text{likelihood}} \underbrace{P(\theta)}_{\text{prior}}$$

Regularization

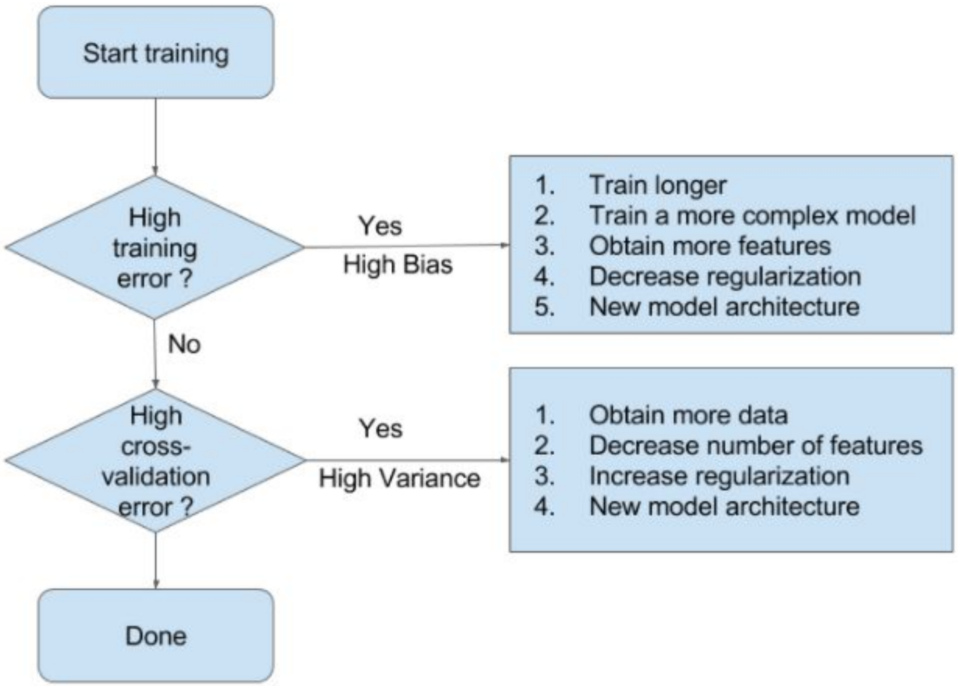
```python
#L1 classifiers
lassos=list()
for alpha in alphas:
    thisLasso = Lasso(alpha=alpha)
    thisLasso.fit(x_train,y_train)
    lassos.append(thisLasso.predict)

#L2 classifiers
ridges=list()
for alpha in alphas:
    thisRidge = Ridge(alpha=alpha)
    thisRidge.fit(x_train,y_train)
    ridges.append(thisRidge.predict)

#accuracies
lassos_test_loss=[(abs(func(x_test)-y_test.reshape(-1))).mean() for func in lassos]
ridges_test_loss=[(abs(func(x_test)-y_test)).mean() for func in ridges]

#generate animation
thisPlotter = Plotter(*[[x_train, y_train]]*2,\
                      plot1={'func':lassos,'alpha':1/alphas,'name':'Lasso L1','loss':lassos_test_loss},\
                      plot2={'func':ridges,'alpha':1/alphas,'name':'Ridge L2','loss':ridges_test_loss})
```



Hyper-Parameter Tuning and Evaluation

A Boston House Price Model
Enjoy!

Conclusion


Start training
High training error ?
Yes
High Bias
1. Train longer
2. Train a more complex model
3. Obtain more features
4. Decrease regularization
5. New model architecture
No
High cross-validation error ?
Yes
High Variance
1. Obtain more data
2. Decrease number of features
3. Increase regularization
4. New model architecture
Done

Practical application of Linear regression

(a) Trend line

- Business analytics tool for forecasting , predicting

And mostly monitoring tool used in various sector

- Technical analysis in share market is interpretation

of trend line.

(b) crop yields on rainfall :

Yield is Dependent variable Rainfall is explanatory variable and by the use of data , prediction model are prepared using simple linear regression

(c) Economics

- To predict consumption/spending
- Fixed investment spending
- Inventory
- Export –import analysis
- Labor demand and supply
- Resource management
- CAPM

MANY MORE…

- SAT /GRE score for college admission
- Product price and sales
- Metal mining
- Impact of extraction on aquatic ecosystem
- Tobacco smoking v/s mortality

Q&A

THANK YOU !