SUBGRAPH MINING

(Cont'd)

Recall: Approach 2 – Apriori like

- The apriori algorithm still holds because a k-graph is frequent only if all of its (k-1) graphs are frequent.

What is a k-graph? k vertices or k edges

Vertex growing

Edge growing

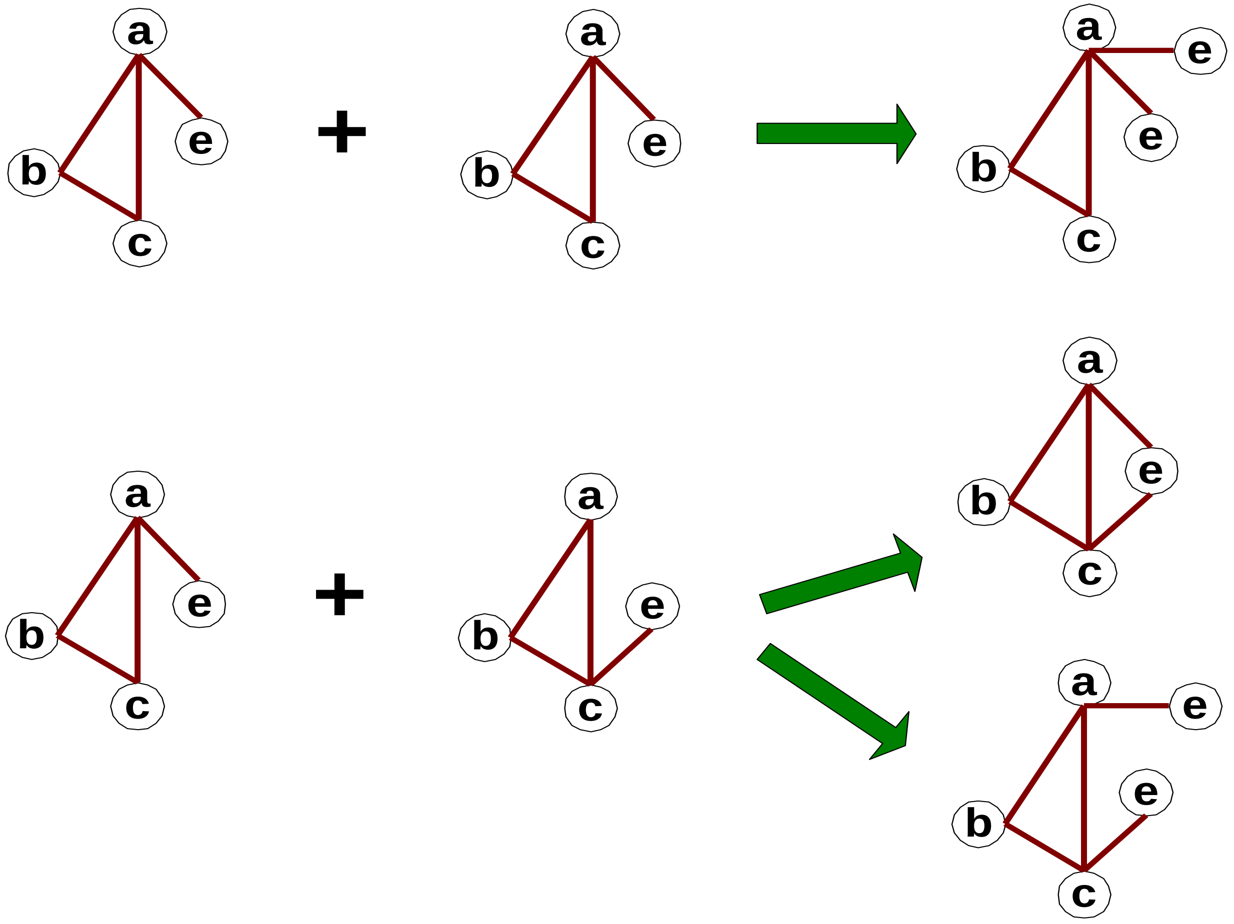
- You start by a small size graph and generate candidates by adding a vertex/edge.
- Candidate generation in graphs is complex

Multiplicity of Candidates (Edge growing)

- In the edge based candidate generation, we increase by one edge at a time.
- Two size k subgraphs are merged if and only if they share the same subgraph with k-1 edges.
- The new candidate will have the core and the two additional edges.
- Edge growing approach creates multiple candidates of different kinds.

Multiplicity of Candidates (Edge growing)

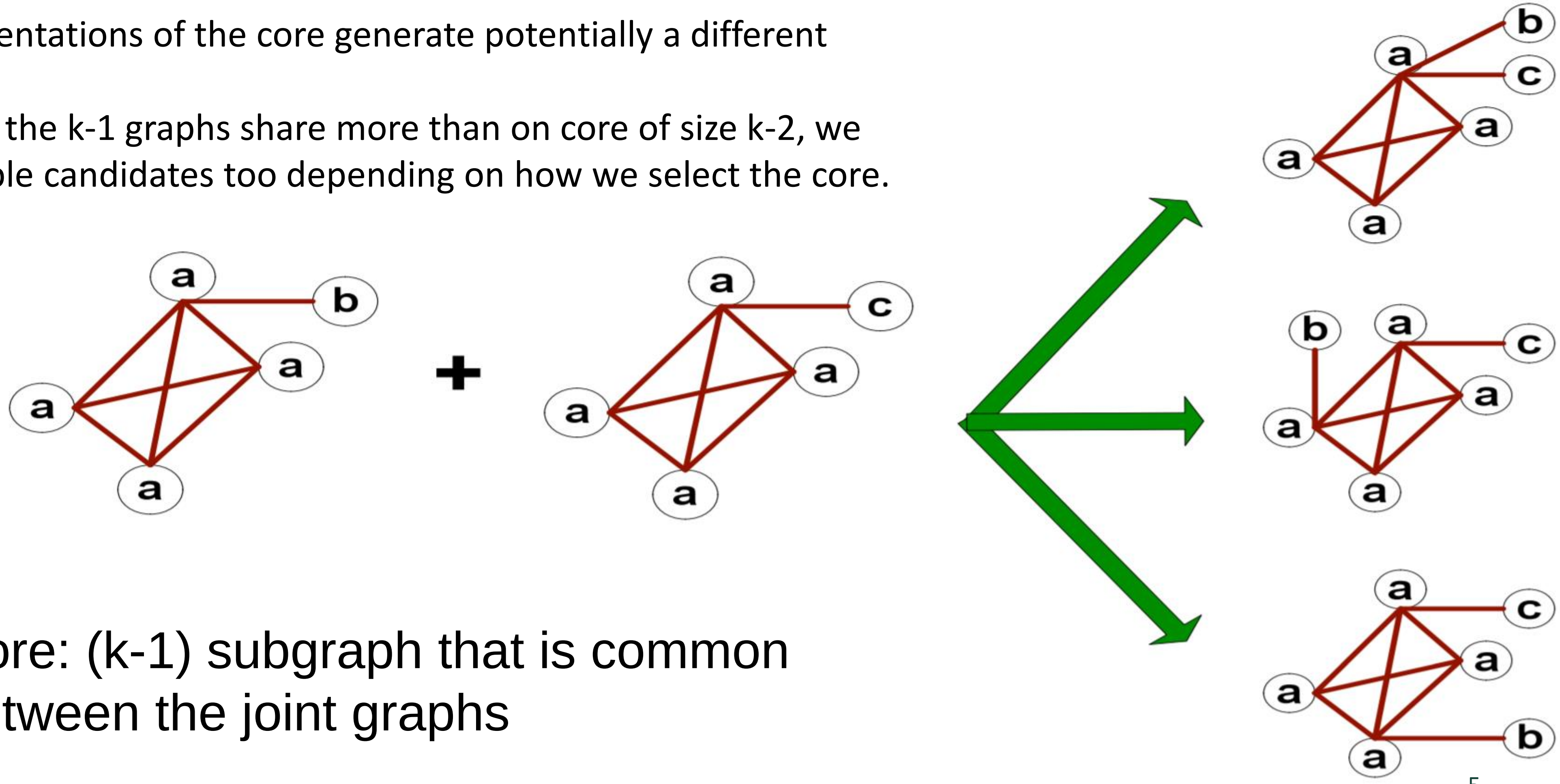
- Case 1: identical vertex labels



Multiplicity of Candidates (Edge growing)

- Case 2: Core contains identical labels

- All symmetric orientations of the core generate potentially a different candidate
- In the case when the k-1 graphs share more than on core of size k-2, we can obtain multiple candidates too depending on how we select the core.

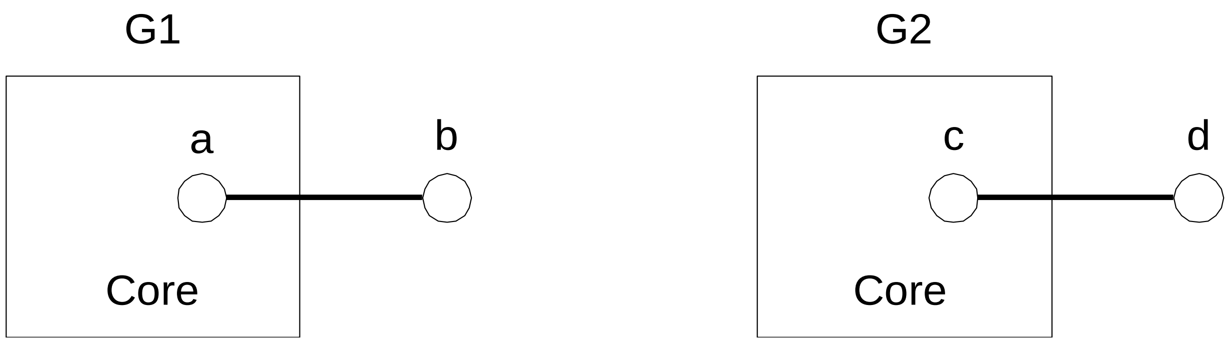


Core: (k-1) subgraph that is common between the joint graphs

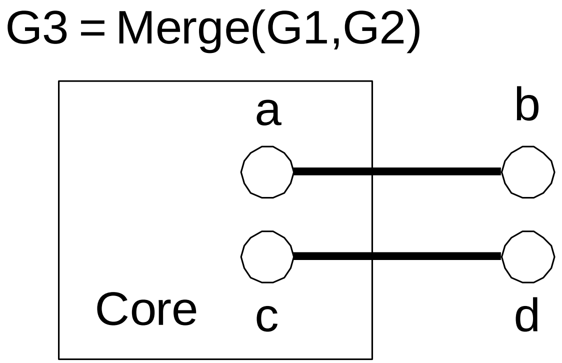
Candidate Generation by Edge Growing

So how do we merge:
- let’s assume that we have 2 graphs,
- A and c are the endpoints of the extra edge.

- Given

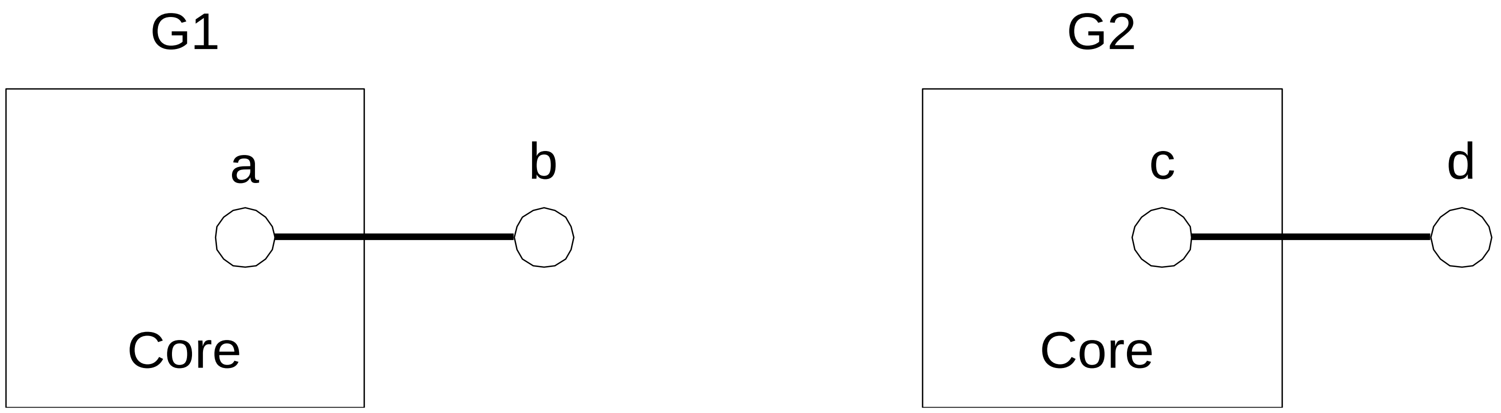


- Case 1: a ≠ c and b ≠ d

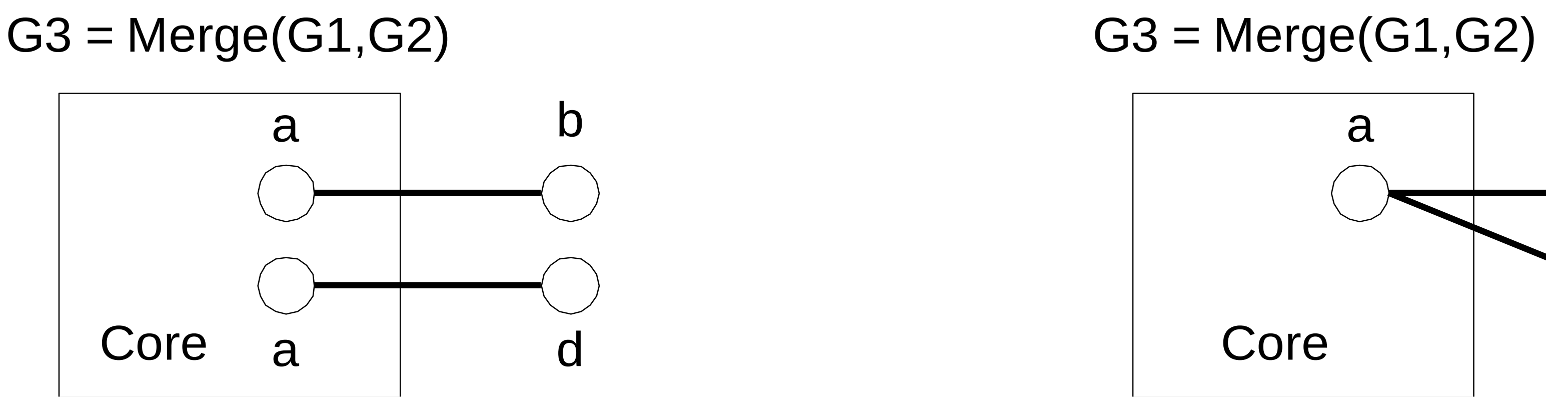


Candidate Generation by Edge Growing

Given

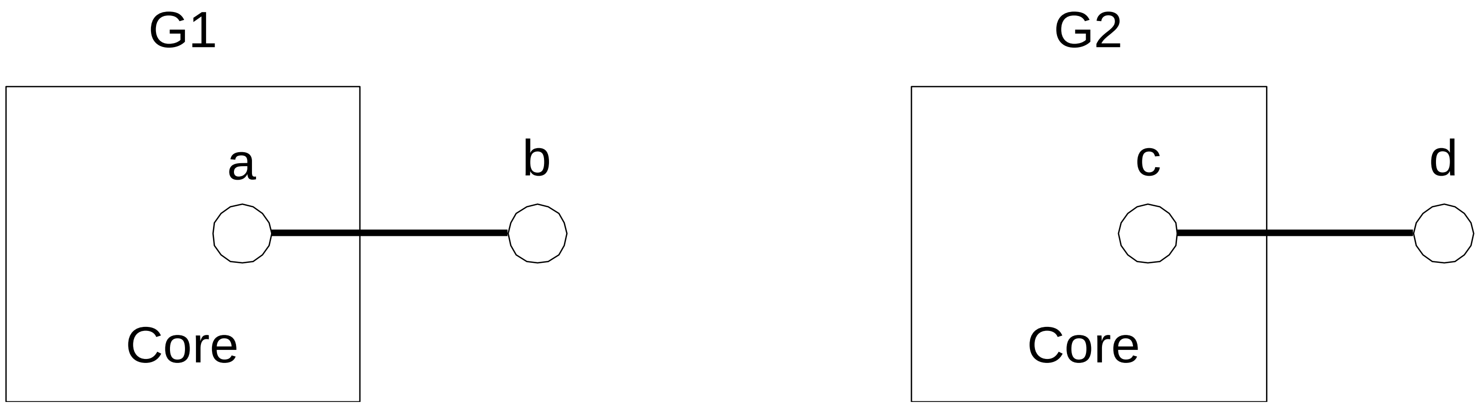


- Case 2: a = c and b ≠ d

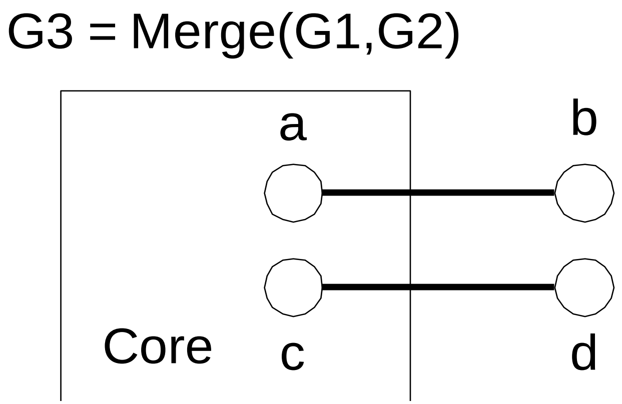


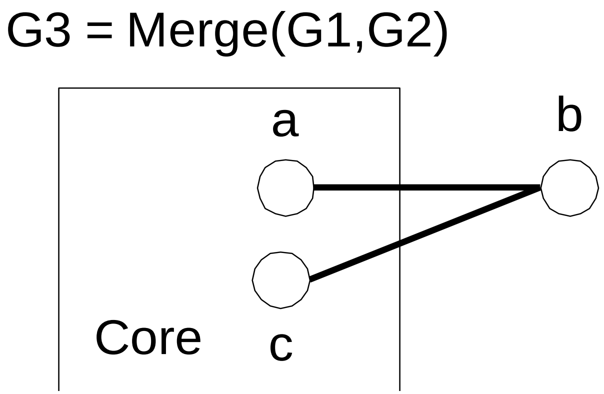
Candidate Generation by Edge Growing

Given



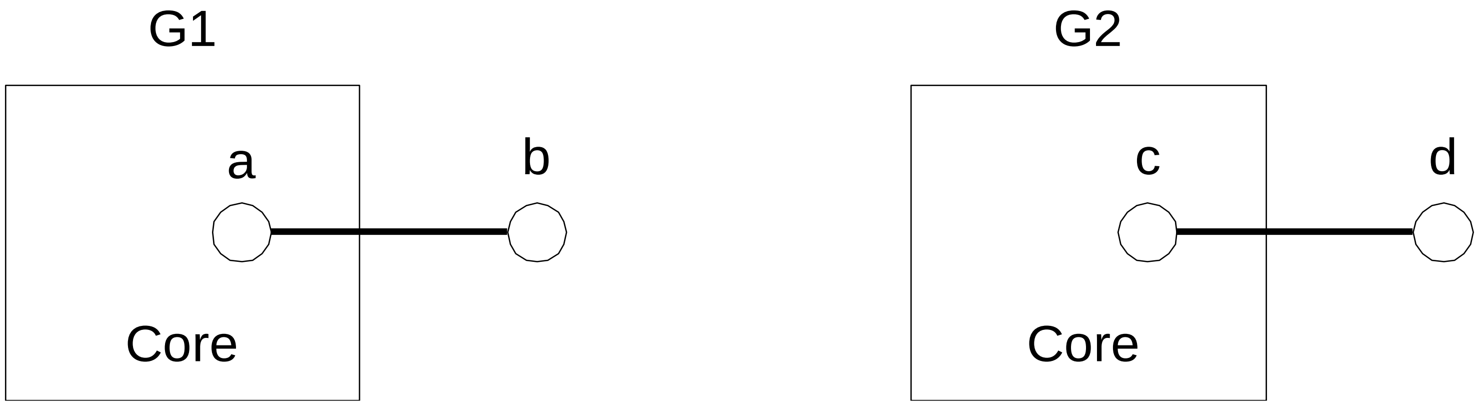
- Case 3: a ≠ c and b = d



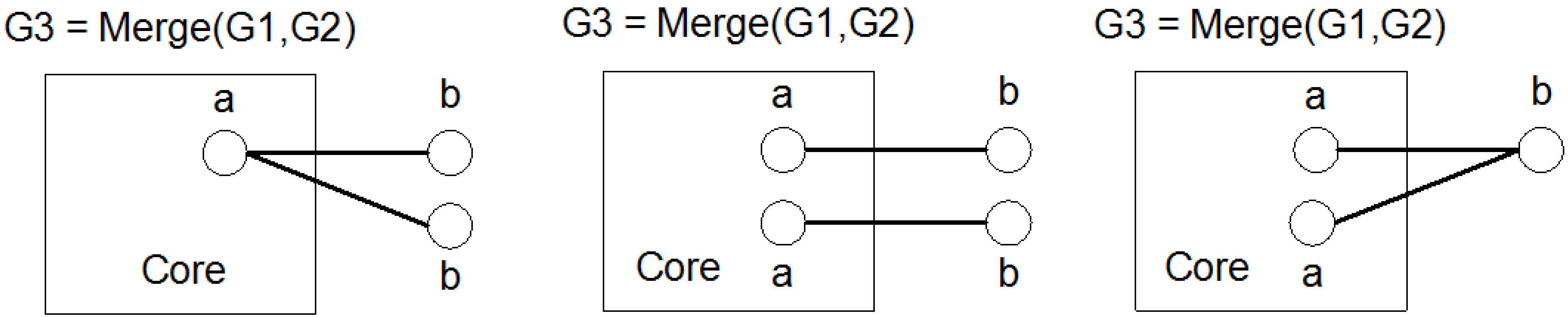


Candidate Generation by Edge Growing

Given

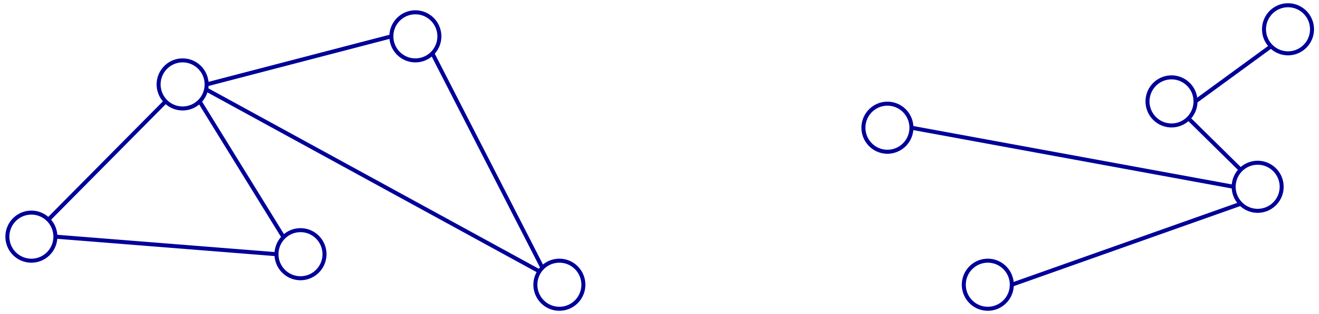


- Case 4: a = c and b = d



Candidate Pruning

- For a candidate k-subgraph, discard it if any of its (k-1)-subgraphs is not frequent
- Successively remove an edge from the k-subgraph
- Check if result is connected. If not, discard it
- If connected, check if it is frequent
  - Determining whether two graphs are topologically equivalent is known as the **graph isomorphism** problem



Applications

- Social Network Analysis
- Mobile call networks
- Biological networks

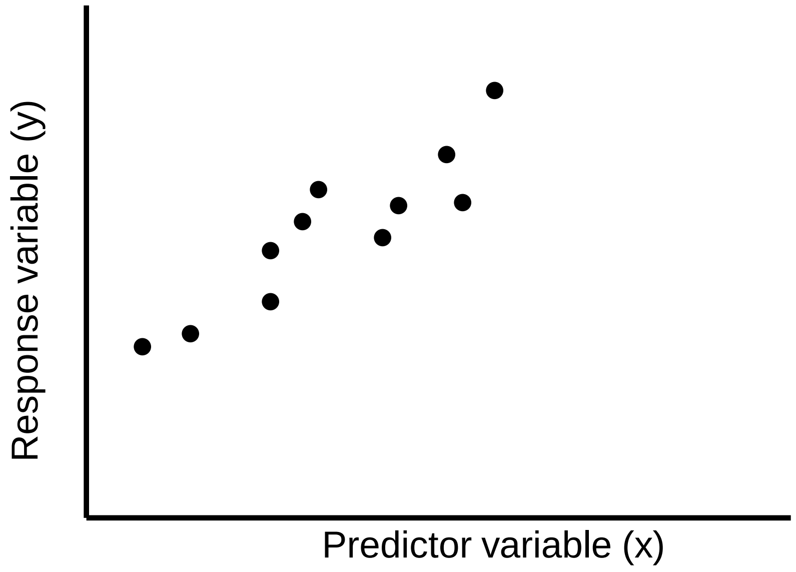
- Analysis:
  - Centrality: Identify most important actors
  - Community Detection
  - Information diffusion: how the information propagate
  - Role identification: who serves as a bridge between groups

Software Packages

- Graph Mining:
  - **gSpan:** graph-based Substructure pattern mining
  - **networkx**
  - Pegasus
  - R
- Sequential pattern mining:
  - SPMF: *www.philippe-fournier-viger.com/spmf/*
  - *R*

REGRESSION ANALYSIS

Regression



- Regression attempts to explain the variability in the dependent (target/response) variable in terms of the variability in independent (predictor) variables.
- If the independent (predictor) variable(s) sufficiently explain the variability in the dependent (target/response) variable, then the model can be used for prediction.

Examples of Regression

Task	Independent variables (Predictor), x	Target/Response variable, y
Forecasting the monthly sales of a company	Historical monthly sales and other predictor variables (inventory, etc)	Monthly sales at time t
Predicting power consumption at data centers	Sensor measurements of temperature, fan speed, etc	Expected power consumed
Predicting crime rate	Statistics about housing, job/income, education, etc	Crime rate in a given city or region

Problem Definition

- Given:
  - A training set $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2),\ldots, (\boldsymbol{x}_N, y_N)\}$, where each $\boldsymbol{x}_i$, corresponds to a set of independent (predictor) variables and y_i is the corresponding value of the dependent (target/response) variable

- Task
  - Learn a target function $f(\boldsymbol{x};\boldsymbol{w})$ to predict the value of y for any given input $\boldsymbol{x}$
    - $\boldsymbol{w}$ is the model parameter

Regression Models

- Linear models
  - Multiple linear regression
  - Ridge regression
  - Lasso regression
- Nonlinear models
  - Neural networks
  - Kernel ridge regression
  - Support vector regression
  - Locally weighted regression
  - Regression trees

Multiple Linear Regression (MLR)

- Assume the target function is linear

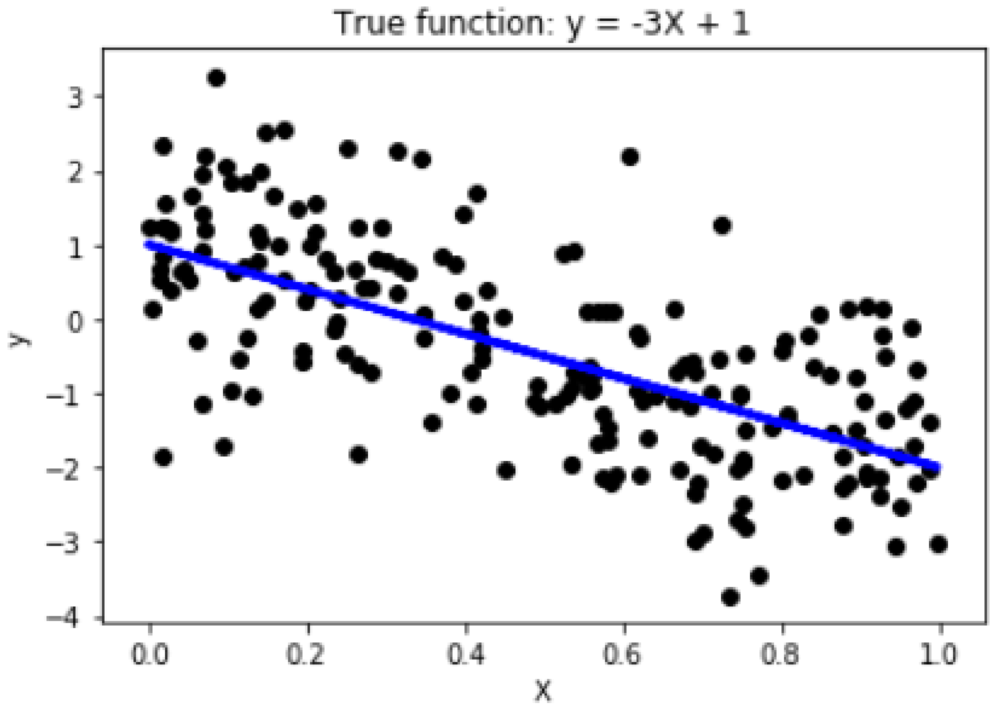
$$f(x;w) = \sum_{j=1}^{d} w_j x_j + w_0 = \sum_{j=0}^{d} w_j x_j = w^T x$$

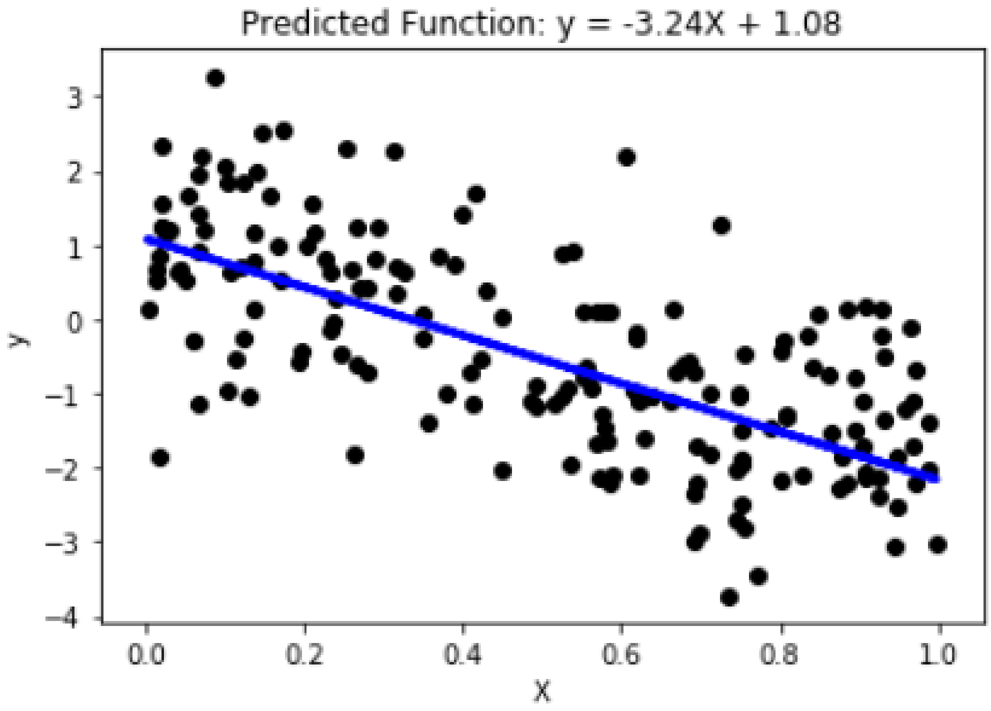
- <u>Estimation</u>: find w that minimizes residual sum of square

$$\min_{w} \sum_{i=1}^{N} (y_i - w^T x_i)^2 \longrightarrow w = \left[X^T X\right]^{-1} X^T y$$

- <u>Prediction</u>: given a test $\hat{x} \longrightarrow f(\hat{x}) = \hat{x}^T \left[X^T X\right]^{-1} X^T y$

Example


True function: y = -3X + 1
3
2
1
0
−1
−2
−3
−4
y
0.0
0.2
0.4
0.6
0.8
1.0
X


Predicted Function: y = -3.24X + 1.08
3
2
1
0
−1
−2
−3
−4
y
0.0
0.2
0.4
0.6
0.8
1.0
X

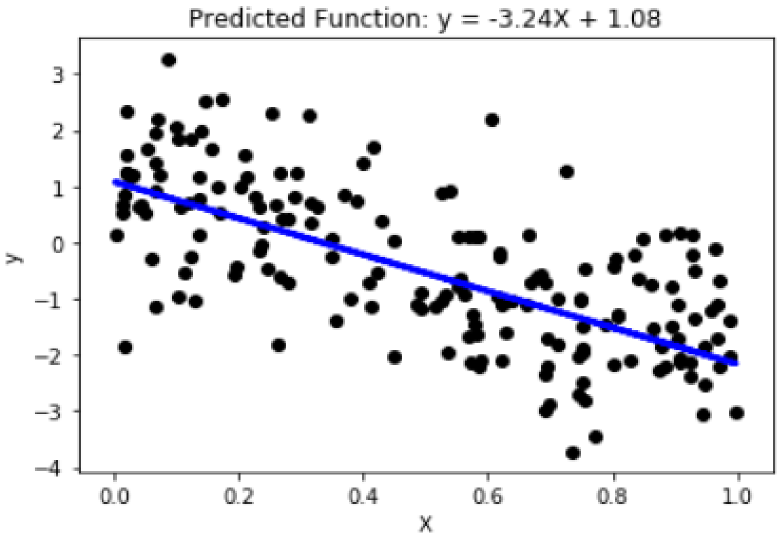
Model Evaluation

- **Root mean square error**
  - Most commonly used measure
    - $\text{RMSE} = \sqrt{\frac{\sum_i (y_i - \hat{y}_i)^2}{N}}$
  - Exaggerate effect of outliers
  - By squaring the errors, larger errors (outliers) are amplified
- **Mean absolute error**
  - Does not exaggerate effect of outliers
    - $MAE = \frac{\sum_i |y_i - \hat{y}_i|}{N}$
- **Relative absolute error**
  - Example: 10% error is equally important
    - Looking at multiple target variables whose scales are different
    - Predict car speed and direction
    - $y = 500$, $\hat{y} = 550$ (degrees)
    - $z = 25.0$, $\hat{z} = 27.5$ (mph)

$$RAE = \frac{\sum_i |y_i - \widehat{y_i}|}{\sum_i |y_i - \bar{y}|}$$

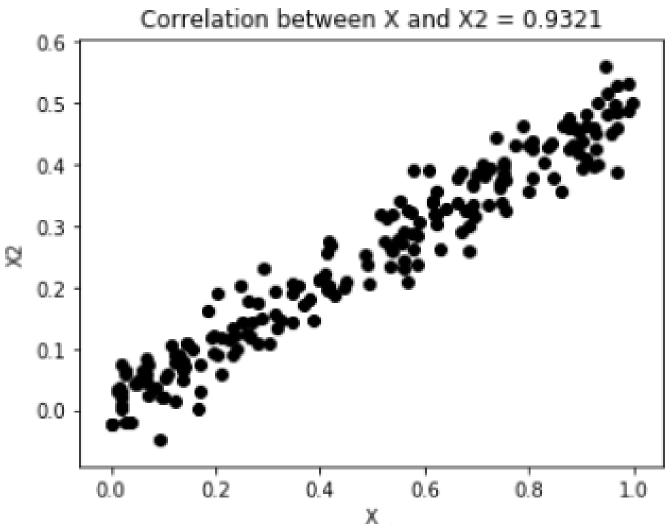
where $\bar{y}$ is calculated from the training data
$\bar{y} = \frac{1}{N} \sum_i y_i$

Effect of Correlated Features



- Suppose we add a correlated feature to the data

$$x_2 = 0.5x + \varepsilon(0, 0.04^2)$$



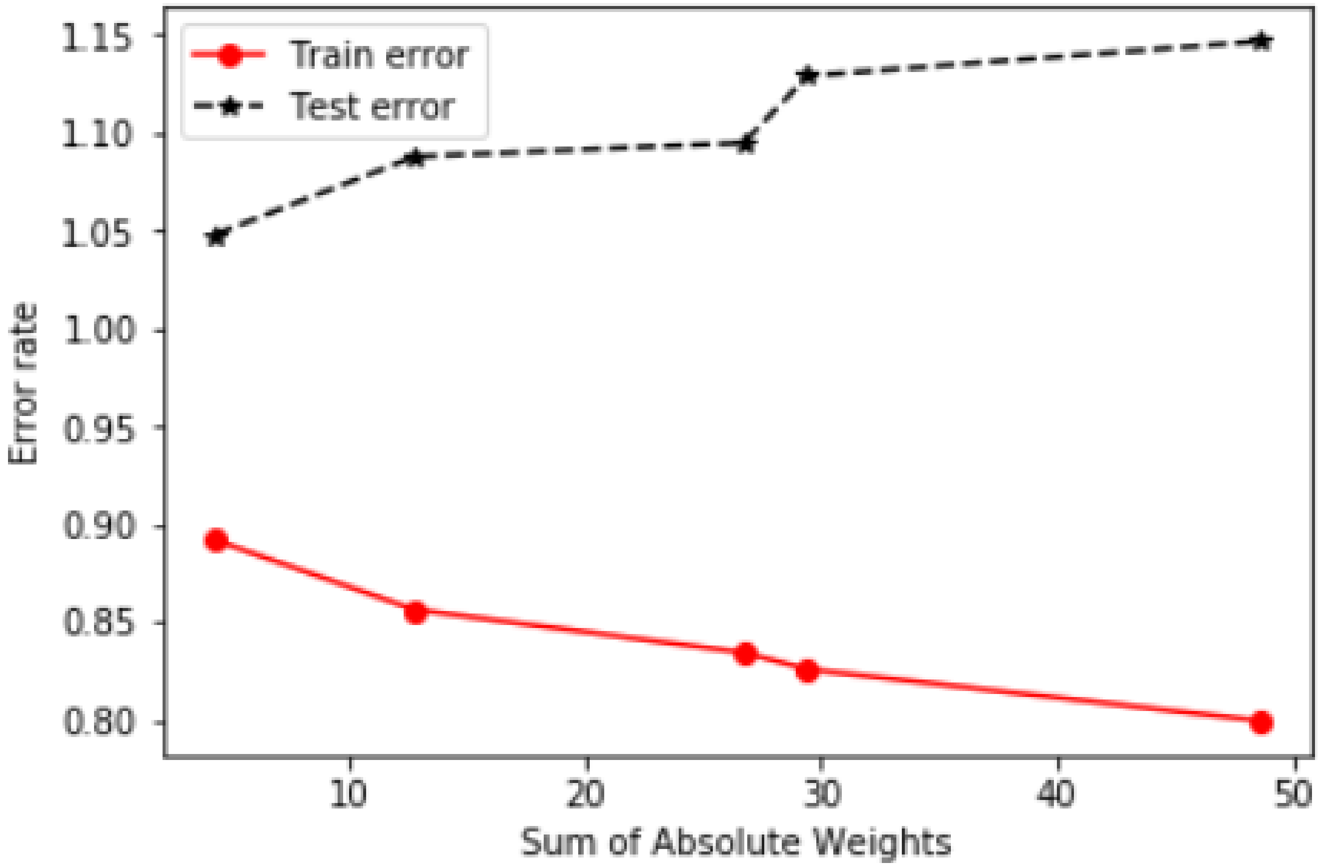
	f(x)	Training error	Test error	$\|w\|_1$
Ground truth	$y = -3x + 1$			4
Original feature	$y = -3.24x + 1.08$	0.8919	1.0476	4.323
Correlated feature x_2	$y = -5.90x + 5.92x_2 + 1.00$	0.8562	1.0876	12.817

Effect of Correlated Features

Suppose we add more correlated features (x_3, x_4, x_5)

	f(x)	Training error	Test error	$\|\|w\|\|_1$
Ground truth	y = -3x + 1			4
Original feature	y = -3.24x + 1.08	0.8919	1.0476	4.323
Correlated feature x_2	y = - 5.90x +5.92x_2 + 1.00	0.8562	1.0876	12.817
Correlated feature x_3	y = - 6.22x - 2.30x_2 +17.14x_3 + 1.08	0.8342	1.0947	26.744
Correlated feature x_4	y = - 7.16x + 0.93x_2 + 8.39x_3 + 11.85x_4 + 1.12	0.8257	1.1289	29.454
Correlated feature x_5	y = - 7.16x + 4.50x_2 + 3.52x_3 - 6.55x_4 + 25.68x_5 + 1.20	0.7994	1.1465	48.615

Effect of Correlated Features



When model becomes overly complex, it is susceptible to <u>overfitting</u> problem

Solution?

Ridge Regression

- Ridge regression shrinks the regression coefficients, so that variables, with minor contribution to the outcome, have their coefficients close to zero.

- The shrinkage of the coefficients is achieved by adding an L2-norm penalty term to the regression model, which is the sum of the squared coefficients.

Ridge Regression

- Uses an L_2-norm to regularize ||w||
- Objective function:

$$\min_{w} \|y - Xw\|^2 + \lambda \|w\|^2$$

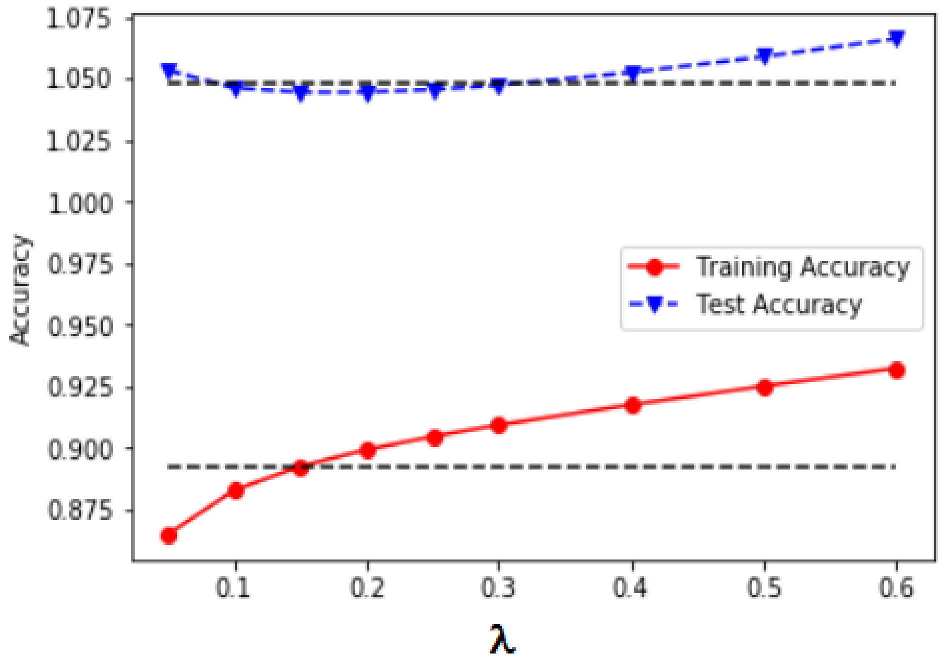
  - where λ is the regularization parameter
  - Increasing λ will reduce the weights of the model parameters
  - λ is typically chosen via cross-validation
- Can be solved in closed form

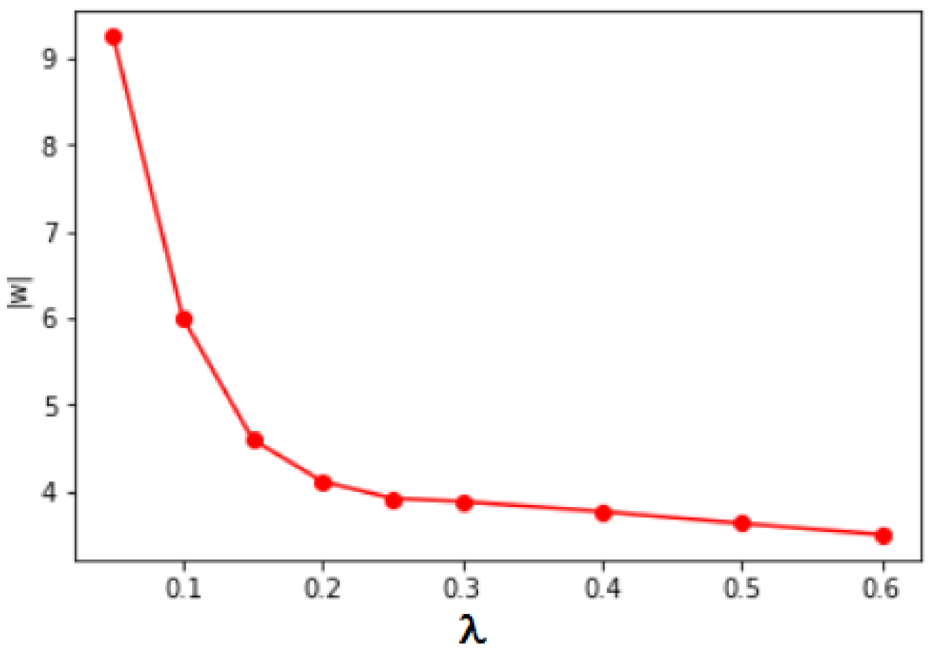
$$w = \left[X^T X + \lambda \mathrm{I}\right]^{-1} X^T y \longrightarrow$$

Reduces to MLR solution when λ goes to zero

Ridge Regression

- Effect of varying regularization parameter λ

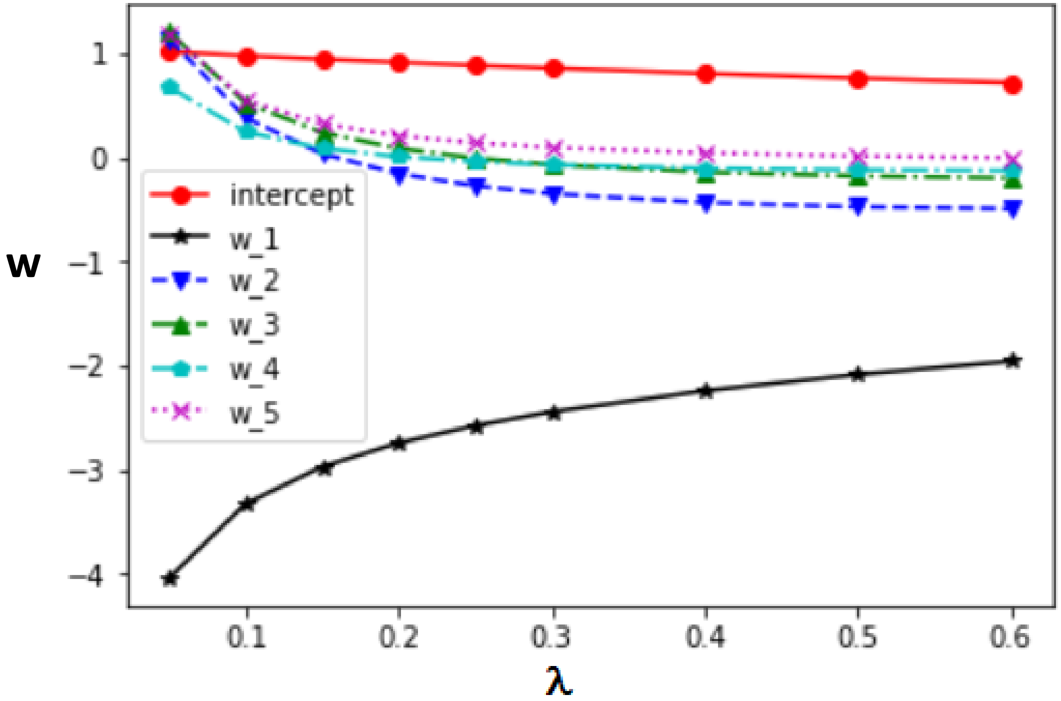




Dashed lines represent the training and test accuracies of MLR without correlated features

Ridge Regression

intercept: weight associated with x_0
w_1: weight associated with x_1
w_2: weight associated with x_2
w_3: weight associated with x_3
w_4: weight associated with x_4
w_5: weight associated with x_5



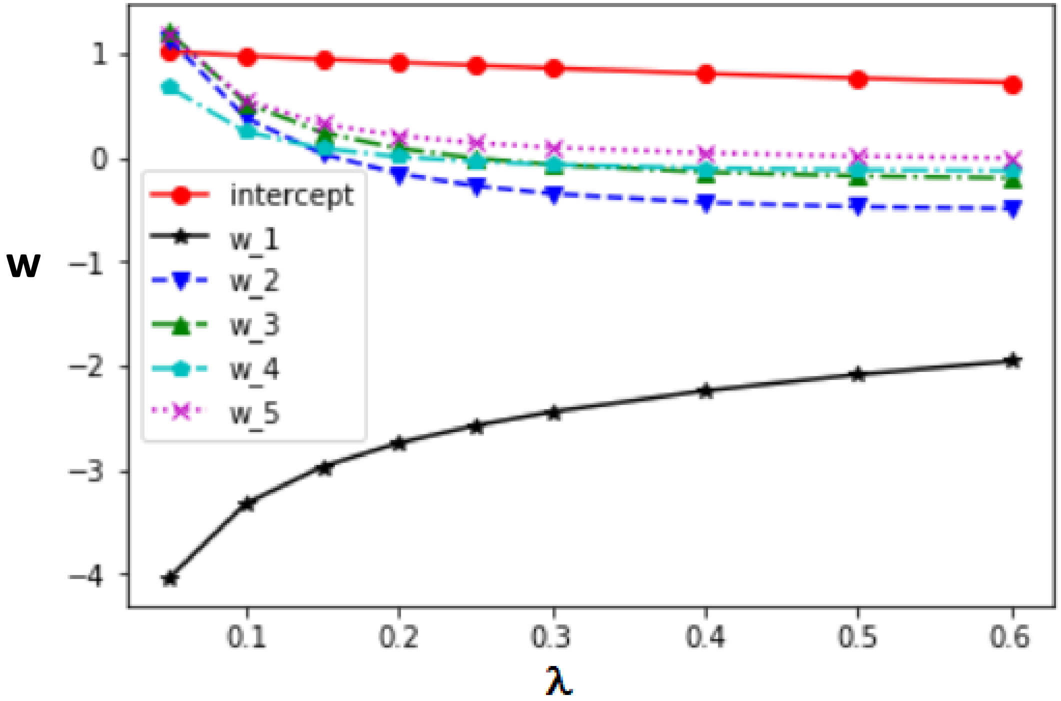
the higher the lamda, the higher the bias

Increasing λ helps to shrink w (but may not be able to zero it out)

λ	intercept	w1	w2	w3	w4	w5
0.05	1.014095	-4.038929	1.136601	1.204033	0.675139	1.189813
0.10	0.976472	-3.325774	0.381790	0.524730	0.245722	0.548883
0.15	0.943383	-2.969599	0.035098	0.239958	0.087161	0.322764
0.20	0.912547	-2.741218	-0.155993	0.086614	0.007009	0.208739
0.25	0.883526	-2.574462	-0.272487	-0.006898	-0.039875	0.140917
0.30	0.856117	-2.442877	-0.347892	-0.068291	-0.069655	0.096521
0.40	0.805593	-2.240061	-0.433153	-0.140596	-0.103277	0.043078
0.50	0.760076	-2.084117	-0.473429	-0.178442	-0.119748	0.013134
0.60	0.718857	-1.956330	-0.491438	-0.199119	-0.127943	-0.005228

Ridge Regression

- Issues



- we don't want to choose big λ values because the coefficients will become very small and therefore they might not be accurately reflecting what's going on

- In other words, the higher the lamda, the lower the variance and the higher the bias.
  -- underfit the target

- need to have a trade off between the variance and the bias

the higher the lamda, the higher the bias

Lasso Regression

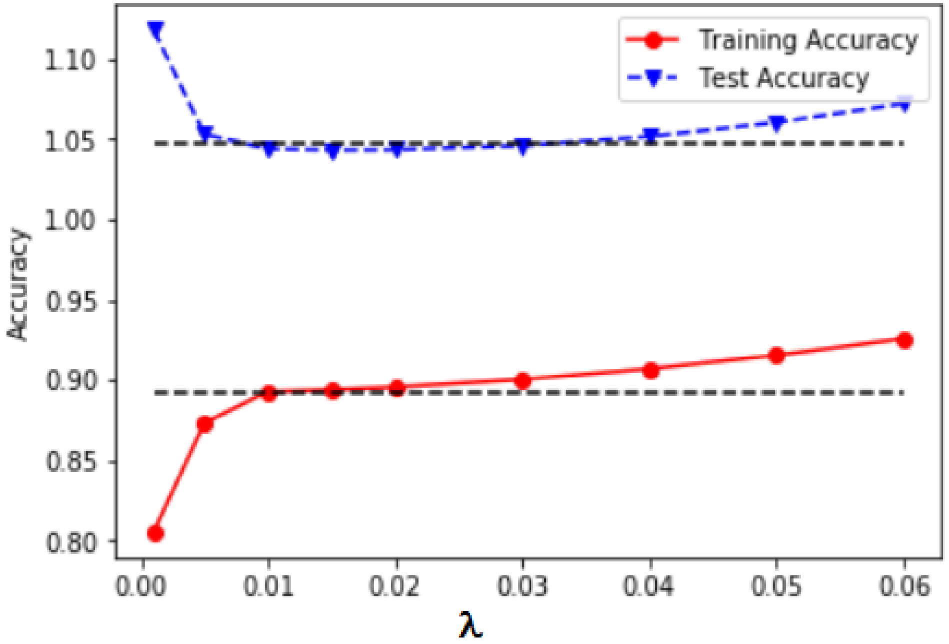
- **Lasso is similar to ridge regression except it uses L1 regularization**
- Uses an L_1-norm to regularize ||w||
- Objective function:

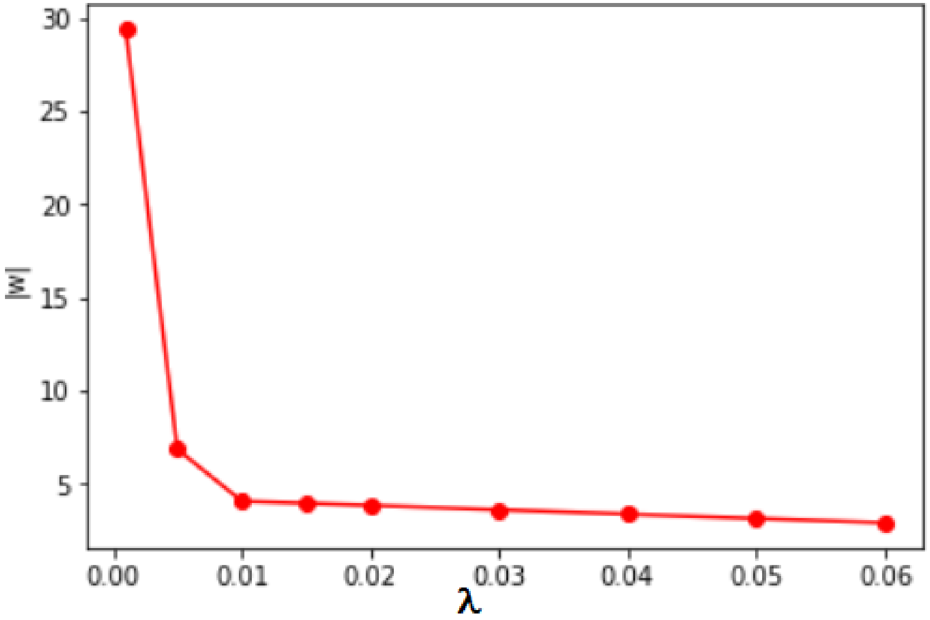
$$\min_{w} \frac{1}{2}\|y - Xw\|^2 + \lambda\|w\|_1$$

  - where λ is the regularization parameter
  - Increasing λ will reduce the weights of the model parameters
  - λ is typically chosen via cross-validation
- Cannot be solved in closed form because $||w||_1$ is not a differentiable function
  - Must be solved iteratively – longer training time (e.g., proximal gradient descent)

Lasso Regression

- Effect of varying regularization parameter λ



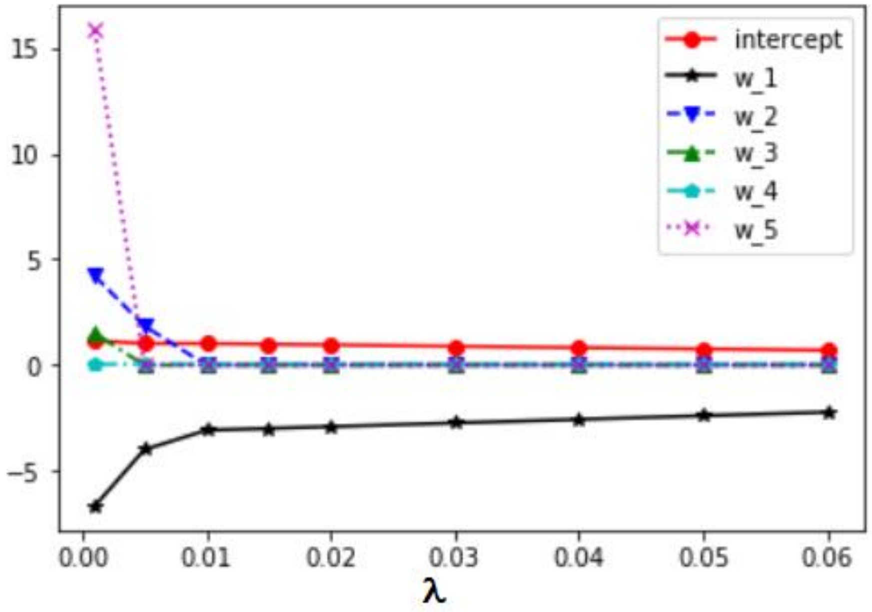


Lasso Regression

- Effect of correlated features

When $\lambda \geq 0.005$, weights of w_3, w_4, and w_5 go to 0

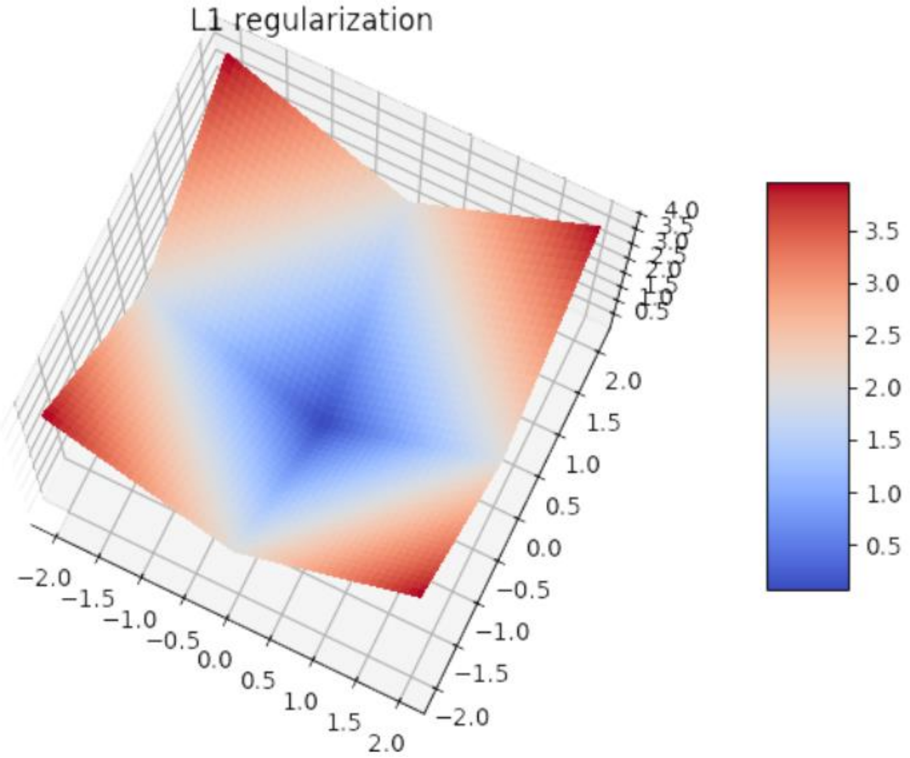
When $\lambda \geq 0.01$, weights of w_2, w_3, w_4, and w_5 go to 0



λ	intercept	w1	w2	w3	w4	w5
0.001	1.153167	-6.655387	4.215581	1.510823	0.0	15.835091
0.005	1.024513	-3.988557	1.851604	0.000000	0.0	0.000000
0.010	1.017915	-3.071683	0.000000	0.000000	0.0	0.000000
0.015	0.986573	-2.986347	0.000000	0.000000	0.0	0.000000
0.020	0.955231	-2.901011	0.000000	0.000000	0.0	0.000000
0.030	0.892546	-2.730340	-0.000000	0.000000	-0.0	0.000000
0.040	0.829862	-2.559668	-0.000000	-0.000000	-0.0	0.000000
0.050	0.767177	-2.388997	-0.000000	-0.000000	-0.0	0.000000
0.060	0.704493	-2.218325	-0.000000	-0.000000	-0.0	-0.000000

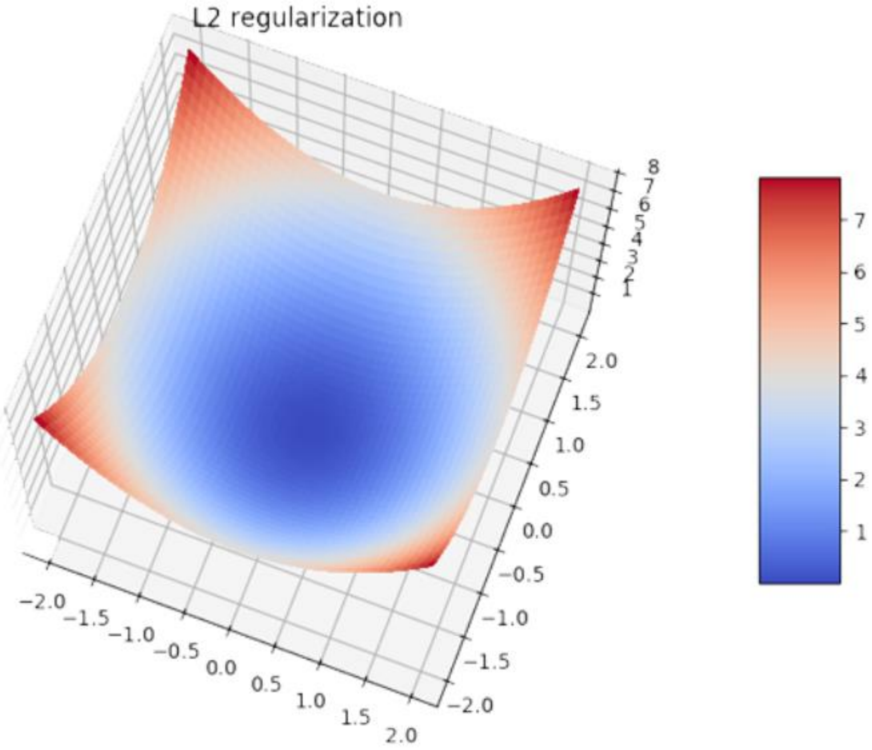
L1 vs L2

Common: The values of the weights try to be as low as possible to reduce penalty



L1 regularization’s shape is diamond-like. Corners of the diamond leads to sparse matrices (some axis/features will be zero).

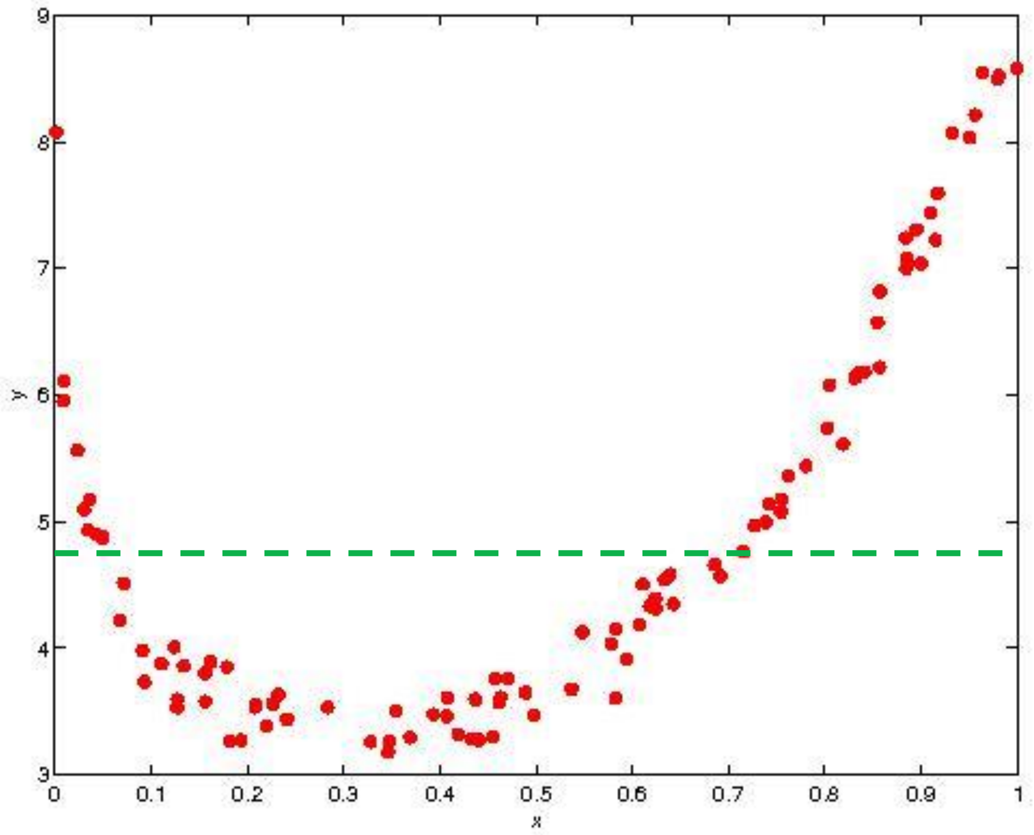
Sparsity



L2 regularization mainly focuses on keeping the weights as low as possible

Smoothness

Nonlinear function



Method	Test error
Linear	14.699
Ridge $\lambda = 0.1$	14.487

Kernel Ridge Regression

- Extends ridge regression to deal with nonlinear features
- combines **ridge regression** (linear least squares with l2-norm regularization) with the **kernel** trick.

$$\min_{w} \|y - \Phi w\|^2 + \lambda \|w\|^2$$

- where

$$\Phi = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_m(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_m(x_2) \\ \dots & \dots & \dots & \dots \\ \phi_1(x_N) & \phi_2(x_N) & \dots & \phi_m(x_N) \end{bmatrix}$$

replaces all data-cases with their feature vector

Kernel Ridge Regression

- What if we don't know the appropriate feature function Φ?
  - Assume Φ is infinite-dimensional and then compute the regression function in infinite-dimensional space

$$L = \left\| y - \Phi w \right\|^2 + \lambda \left\| w \right\|^2$$

$$\nabla_w L = -2\Phi^T y + 2\Phi^T \Phi w + 2\lambda w = 0$$

$$\longrightarrow w = \left[\Phi^T \Phi + \lambda I \right]^{-1} \Phi^T y$$

  - Then apply kernel trick!

Kernel Ridge Regression

- Kernel ridge regression requires computing the dot product $\Phi\Phi^{\top}$ in high-dimensional space

$$w = \left[\Phi^T \Phi + \lambda I\right]^{-1} \Phi^T y$$

- Kernel trick:

$$w = [K + \lambda I]^{-1} \Phi^{\top} y$$

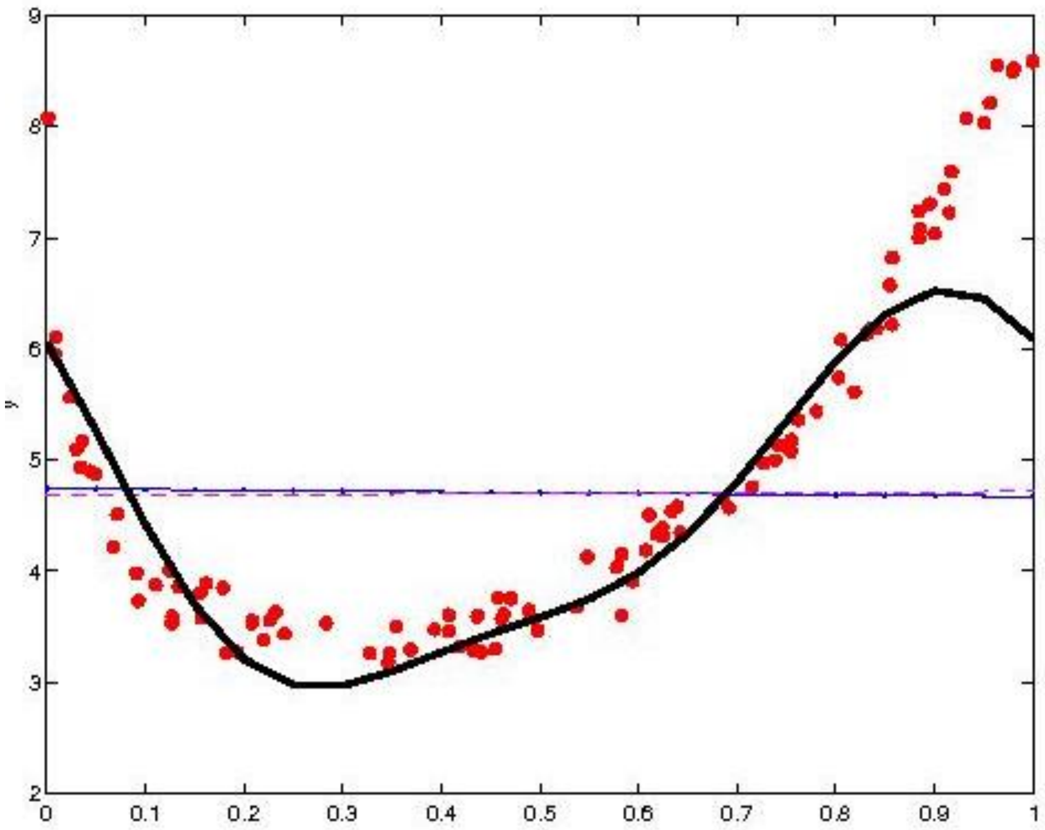
  - The inner product $\Phi\Phi^{\top}$ can be computed in its original feature space (instead of some transformed high-dimensional feature space Φ)

$$K(x, y) = (x \cdot y + 1)^p$$

$$K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$$

$$K(x, y) = \tanh(kx \cdot y - \delta)$$

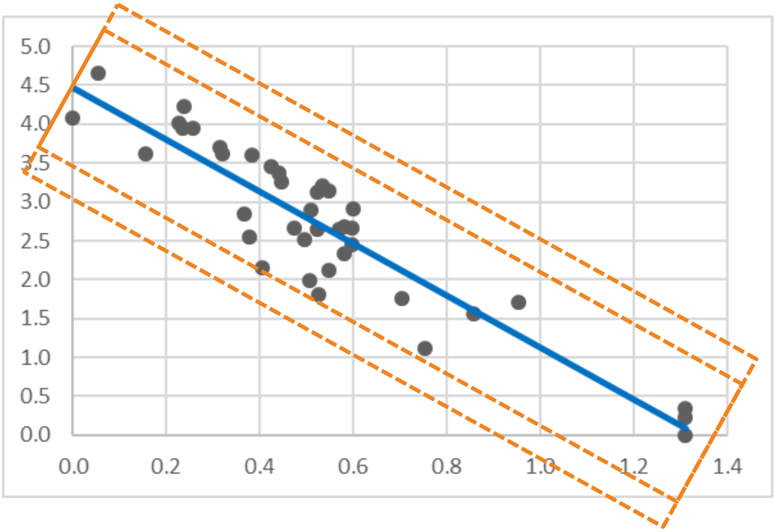
Nonlinear function



Method	Test error
Linear	14.699
Ridge $\lambda = 0.1$	14.487
Kernel	6.286

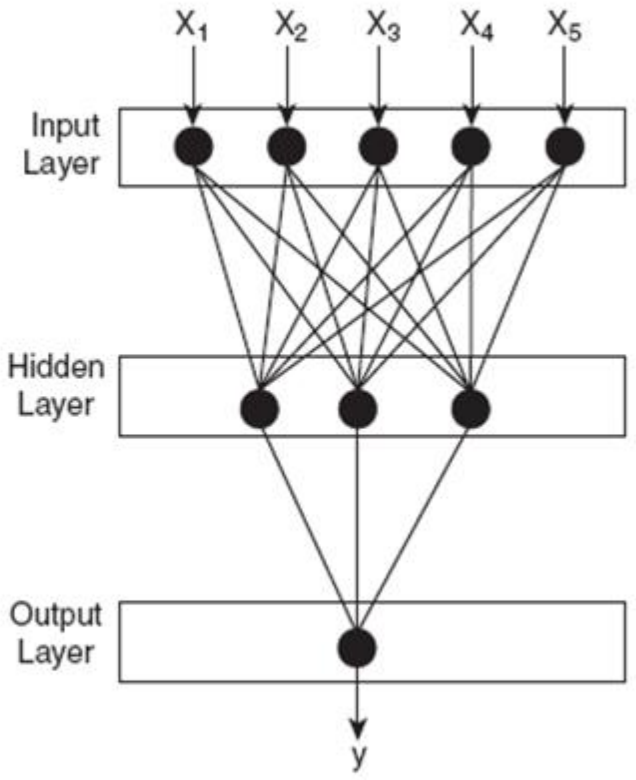
Support Vector Regression

- Similar to linear regression, learn a function to minimize prediction error
  - Disregard small errors
- User specifies ϵ, radius of a tube around the regression function
  - Points within this tube, error = 0
  - If tube can fit all training data
    - Function in the middle of the flattest tube that encloses them is returned
    - Training error = 0
  - Otherwise
    - Tradeoff between prediction error and tube flatness



Neural Networks for Regression

- Similar network structure to classification
- Different output layer and loss function
  - Classification
    - Output node for each class, class predicted as:
    - Sign function – 2 classes
    - Softmax function – 3+ classes
  - Regression:
    - Output 1 node



Regression Example
Predict Vehicle Miles per Gallon

- Output: MPG (column 1)
- Input: (columns 2-8)
  - Number of cylinders
  - Displacement
  - Horsepower
  - Weight
  - Acceleration
  - Model Year
  - Origin

Learn Regression Models

```
import numpy as np
from sklearn.model_selection import train_test_split
# Load the data
data = np.loadtxt('auto-mpg.csv', delimiter=',')
y = data[:,0]
x = data[:,1:8]
# Split into training and test
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.5, random_state=891)
```

Regression Algorithms

- Multiple Linear (Ordinary least squares)
- Ridge
- Lasso
- Kernel
- Neural Network

Multiple Linear / Ordinary Least Squares

```
from sklearn import linear_model
# Train model
reg = linear_model.LinearRegression()
reg.fit(X_train,y_train)
# View coefficients
print(reg.coef_)
```

```
[-0.65834328  0.01405478
-0.0237873  -0.00567093
-0.05662719  0.72666556
0.74193786]
```

```
from sklearn.metrics import mean_squared_error
# Predict test set
y_pred = reg.predict(X_test)
print(mean_squared_error(y_test,y_pred))
```

```
11.944338673592521
```

Ridge Regression

```
from sklearn import linear_model
# Train model
reg = linear_model.Ridge (alpha = .5)
reg.fit(X_train,y_train)
# View coefficients
print(reg.coef_)
```

[-0.65183737 0.0139174 -0.0236812 -0.0056743 -0.05646149 0.72650549 0.73621476]

alpha = λ

```
from sklearn.metrics import mean_squared_error
# Predict test set
y_pred = reg.predict(X_test)
print(mean_squared_error(y_test,y_pred))
```

11.948494405035525

Regression Algorithms

- Multiple Linear (Ordinary least squares)
  - reg = linear_model.LinearRegression()
- Ridge
  - reg = linear_model.Ridge (alpha = .5)
- Lasso
  - reg = linear_model.Lasso(alpha = 0.1)
- Kernel
  - from sklearn.kernel_ridge import KernelRidge
  - reg = KernelRidge(alpha=1.0)
- Support Vector*
  - from sklearn.svm import SVR
  - reg = SVR(gamma='scale', C=1.0, epsilon=0.2)
- Neural Network
  - from sklearn.neural_network import MLPRegressor
  - reg = MLPRegressor()

*Sometimes Support vector regression produces an error, use LIBSVM for Support Vector Machines