Data Mining Quality Analytics

hcai

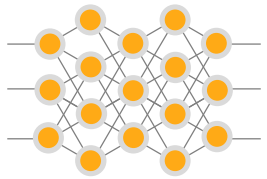
HCAI Open Seminar

Regularization in machine learning

2019. 10. 11

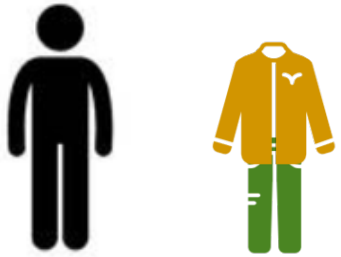
Chang Hyun Lee

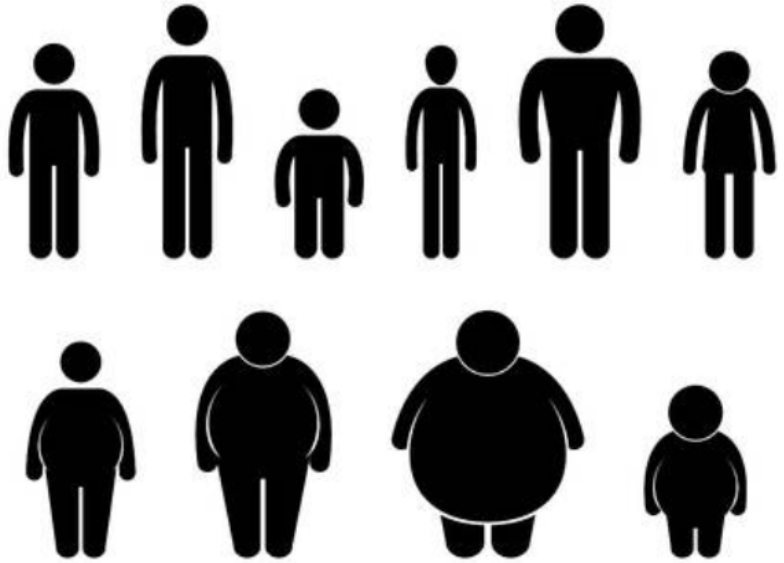
Data Mining & Quality Analytics Lab.



Overfitting

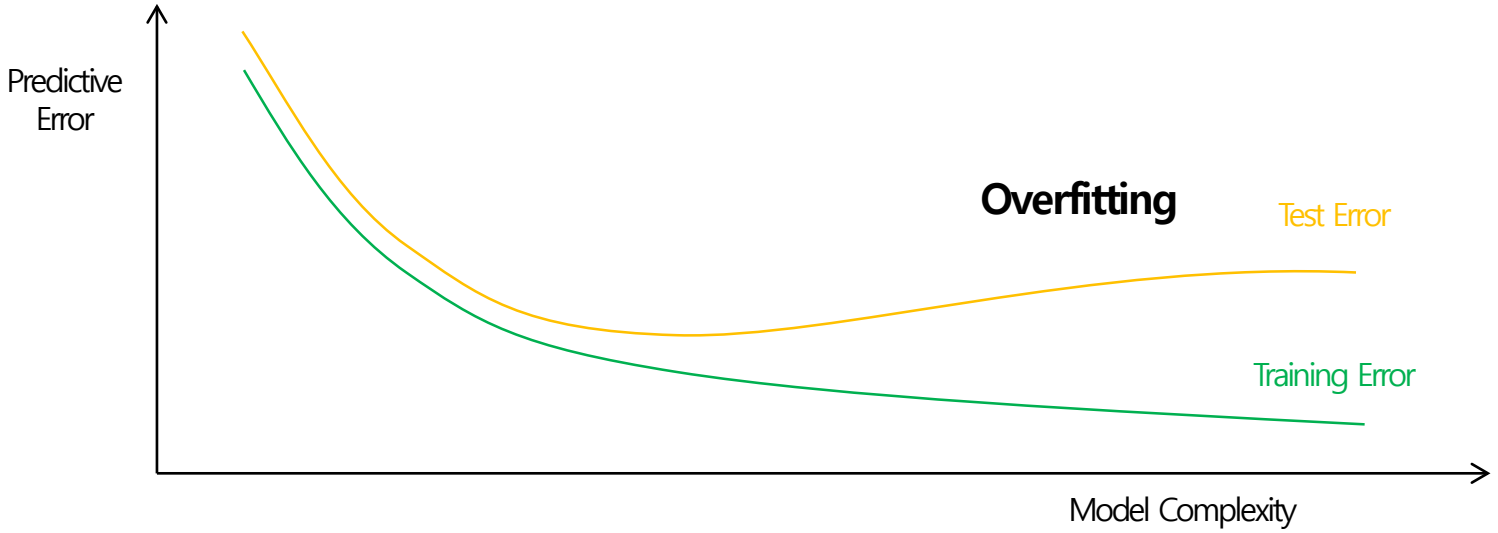
In statistics, overfitting is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably". - Wikipedia





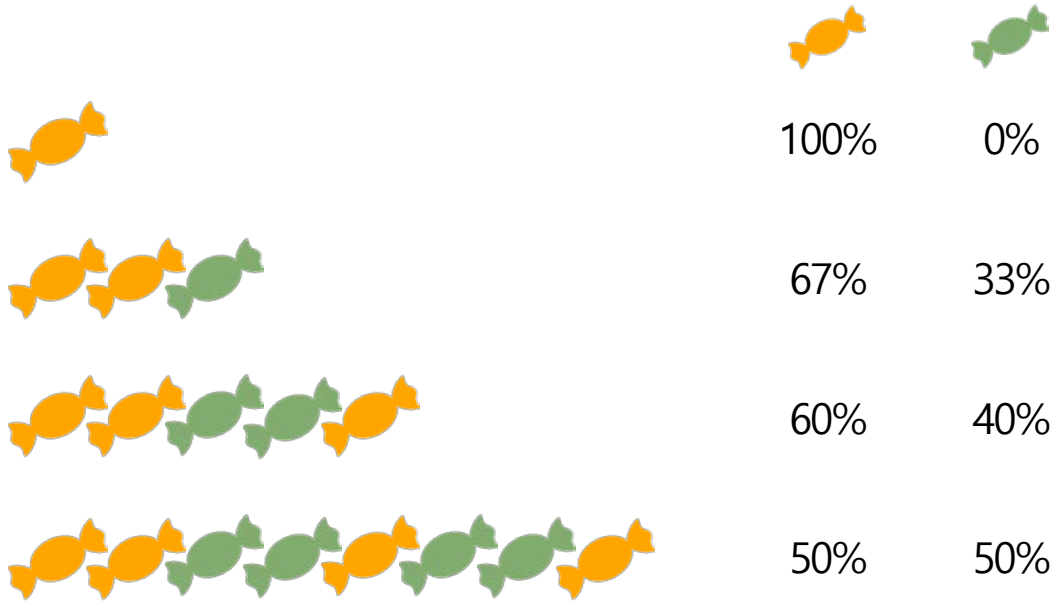
Overfitting

In statistics, overfitting is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably". - Wikipedia



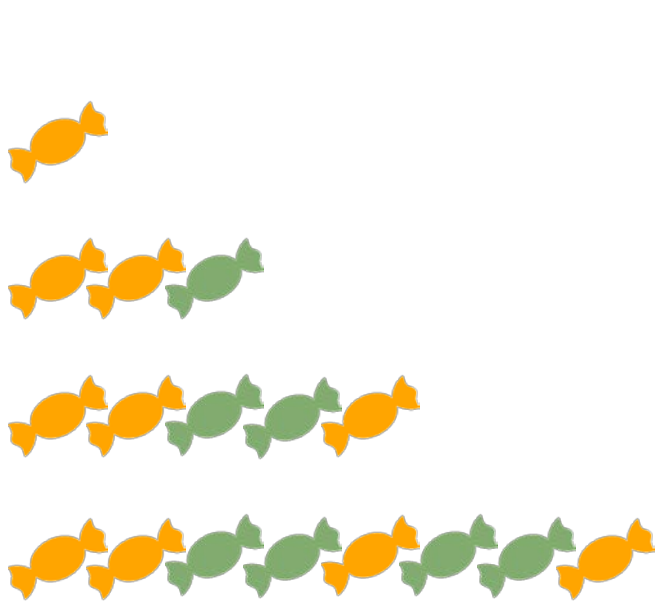
Increasing Data Size

To prevent overfitting, the best solution is to use more training data. A model trained on more data will naturally generalize better. - Tensorflow



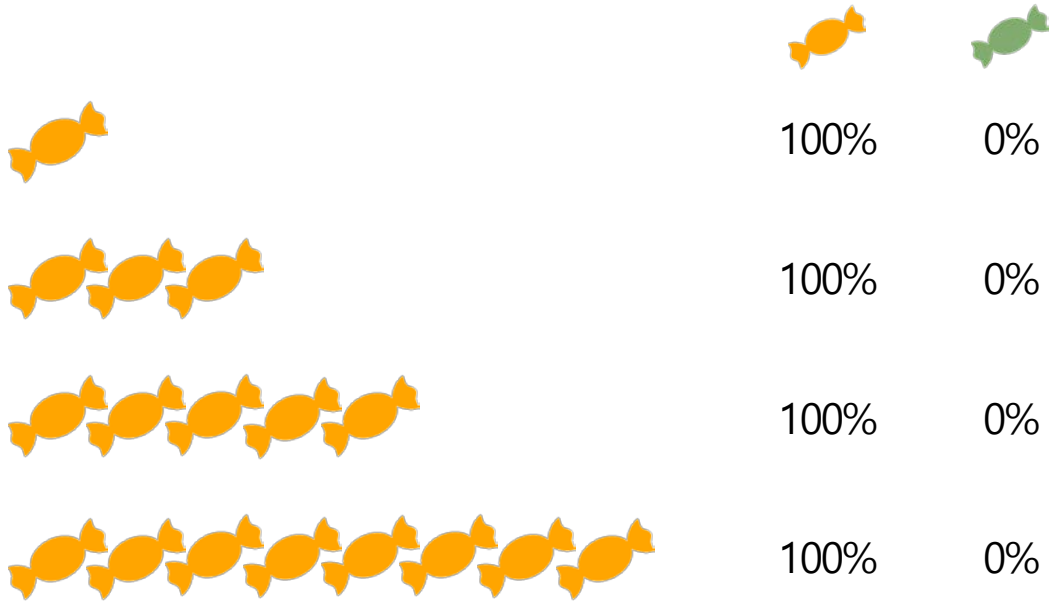
Increasing Data Size

To prevent overfitting, the best solution is to use more training data. A model trained on more data will naturally generalize better. - Tensorflow



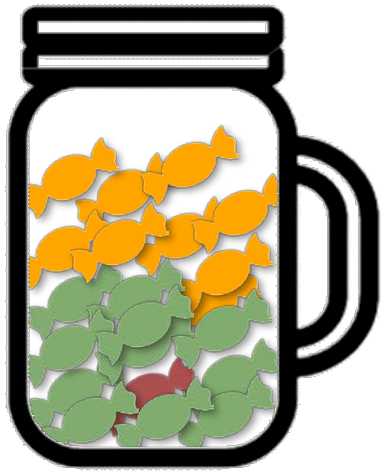
Increasing Data Size

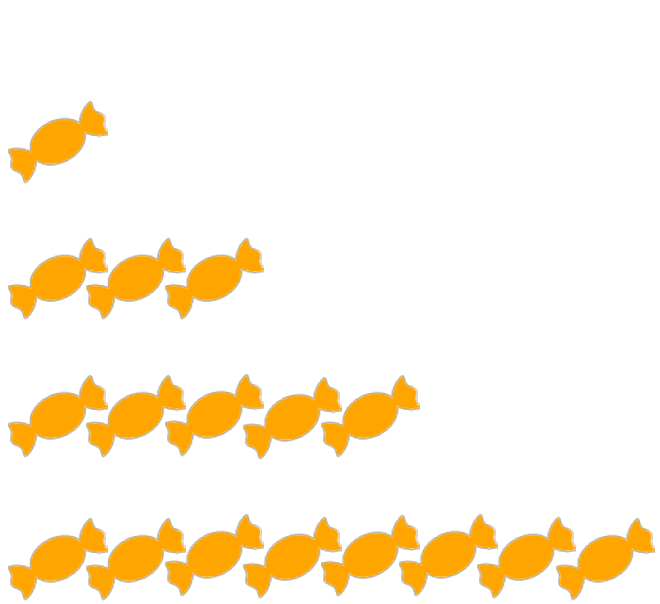
To prevent overfitting, the best solution is to use more training data. A model trained on more data will naturally generalize better. - Tensorflow

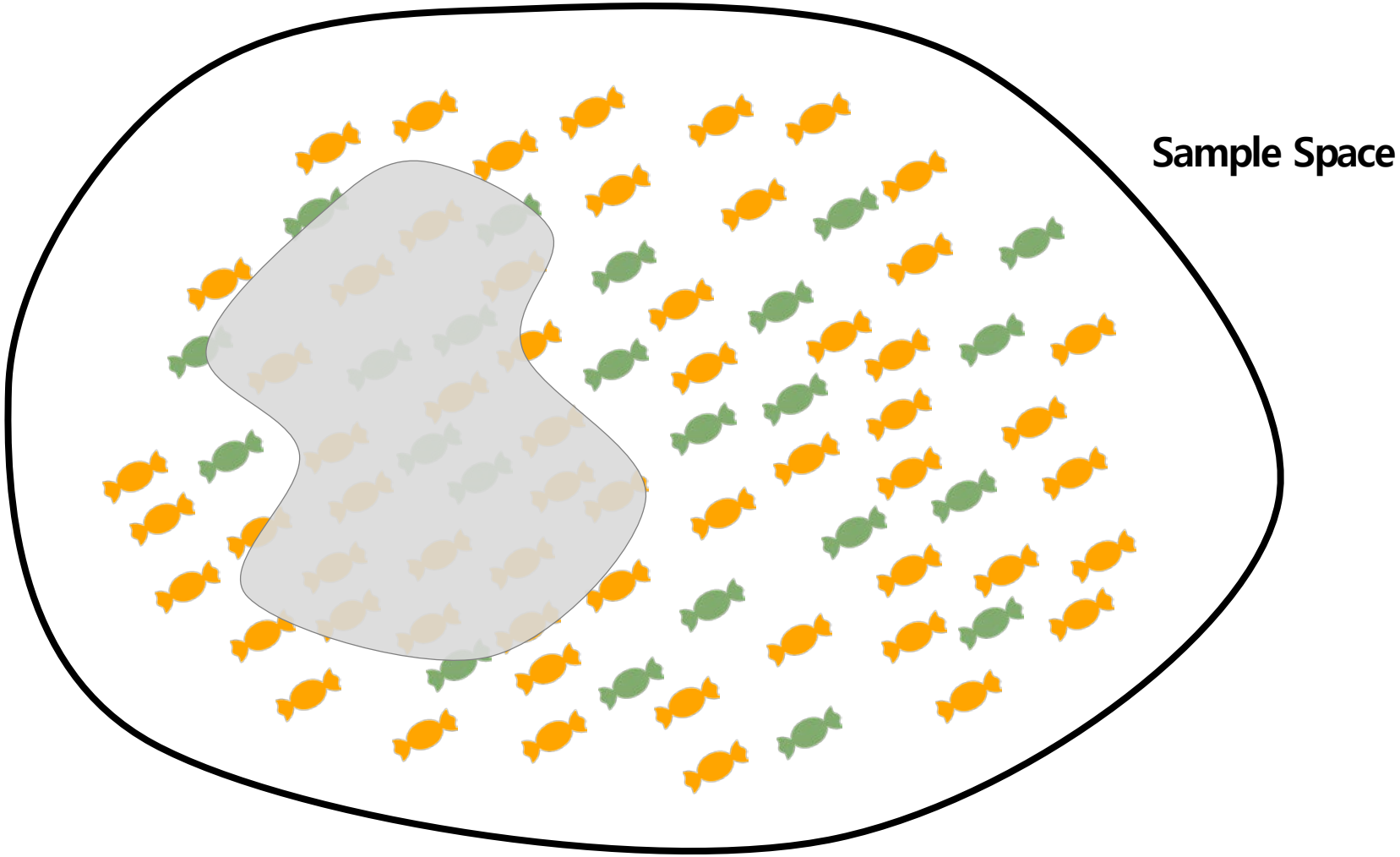


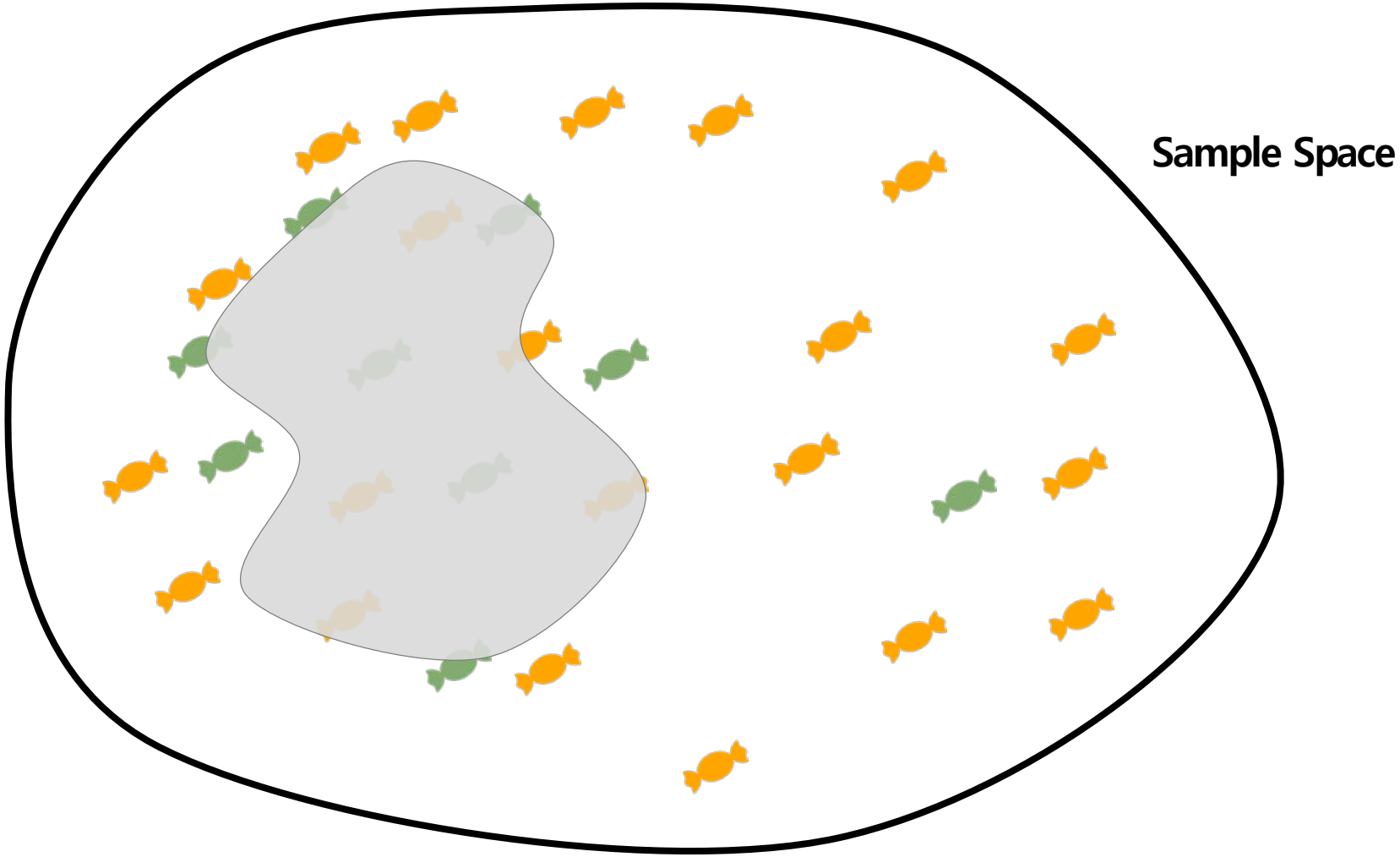
Increasing Data Size

To prevent overfitting, the best solution is to use more training data. A model trained on more data will naturally generalize better. - Tensorflow



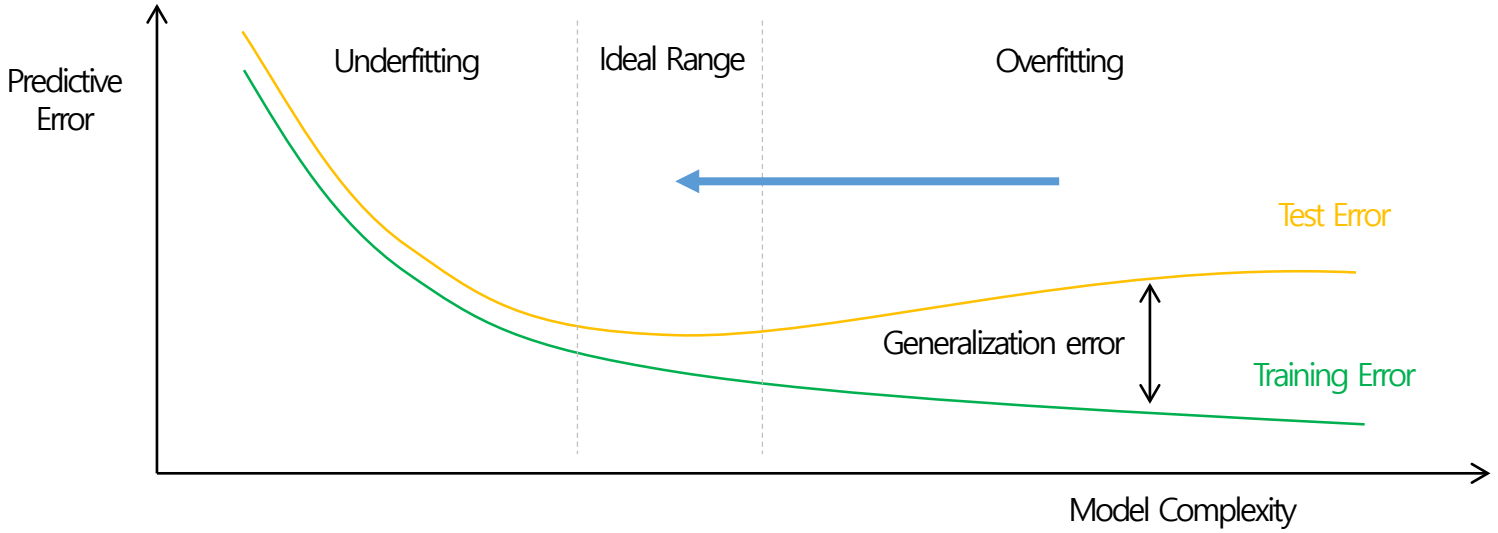



Sample Space


Sample Space

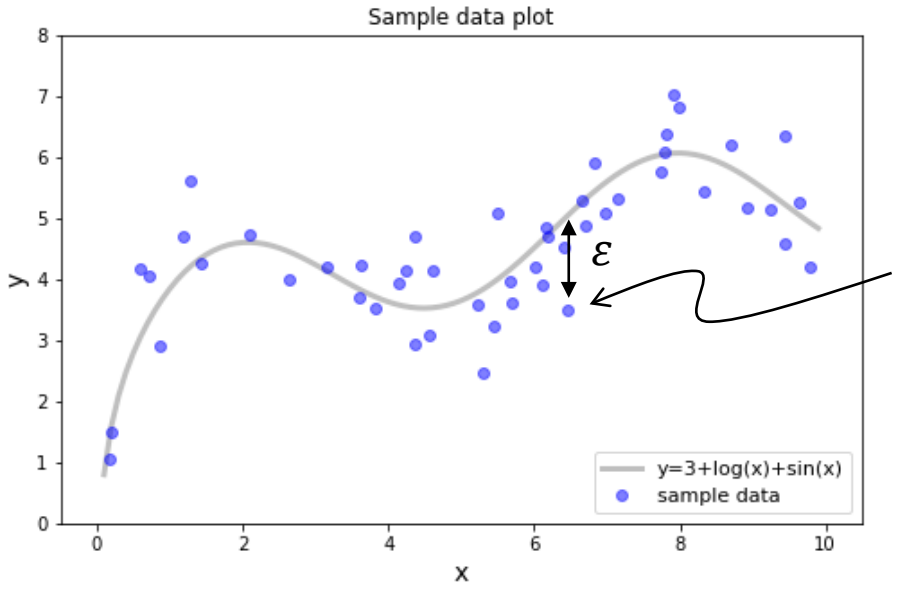
Regularization

"Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error." - Goodfellow



Regularization

Sample Data for Regularization

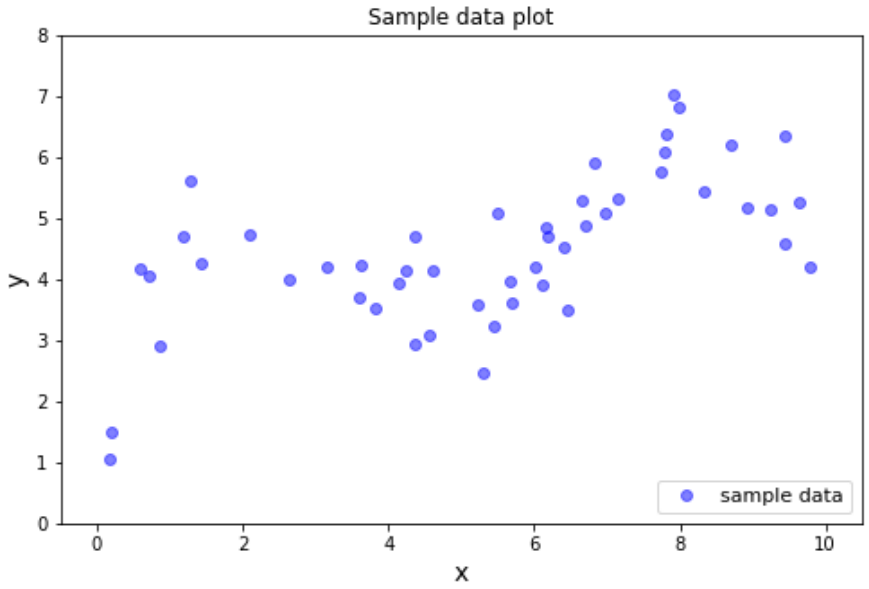


$$y = 3 + log(x) + sin(x) + \varepsilon$$

$$\varepsilon \sim N(0,1)$$

Linear Regression

In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).



$$f(x) = \beta^T x + b$$

$$\min_{\beta} \sum_{i=1} (y_i - f(x_i))^2$$

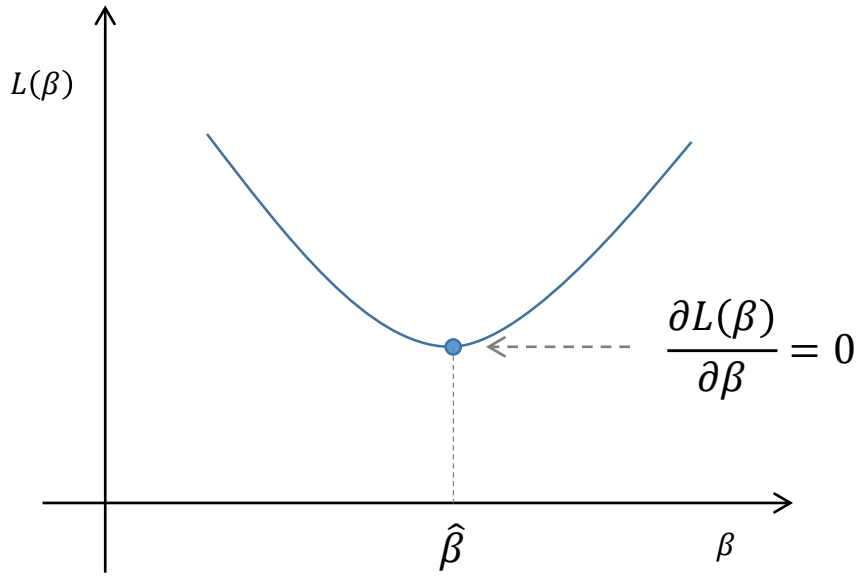
$$\hat{\beta} = (x^T x)^{-1} x^T y$$

Linear Regression

$$f(x) = \beta x$$

$$L(\beta) = \sum_i (y_i - f(x_i))^2$$

$$= \sum_i (y_i - \beta x_i)^2$$

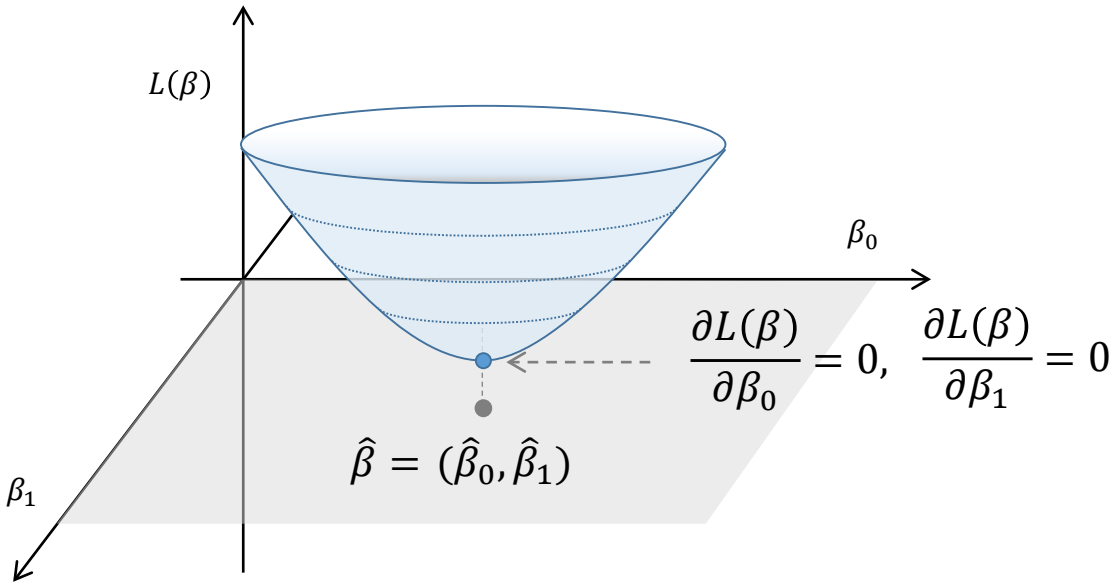


Linear Regression

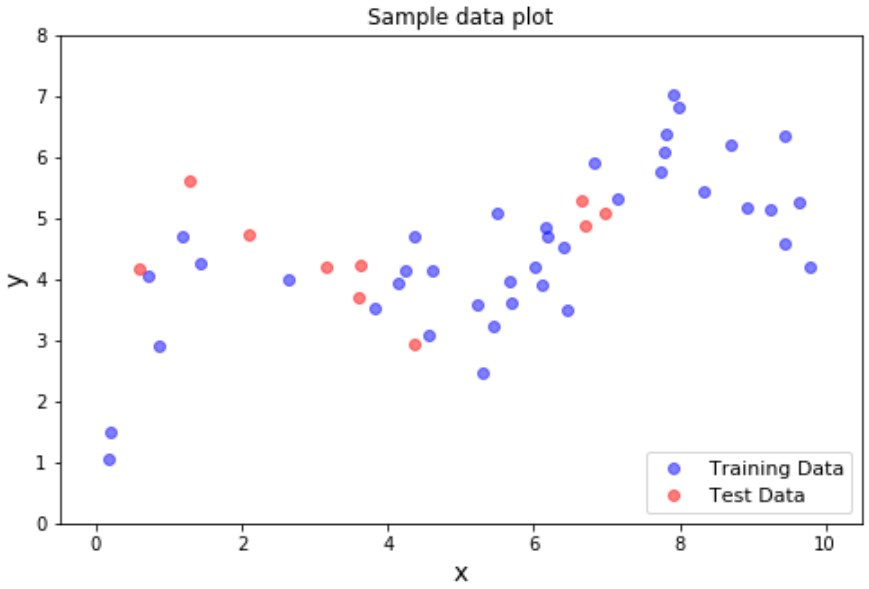
$$f(x) = \beta_0 + \beta_1 x$$

$$L(\beta) = \sum_i (y_i - f(x_i))^2$$

$$= \sum_i (y_i - \beta_0 - \beta_1 x_i)^2$$



Linear Regression

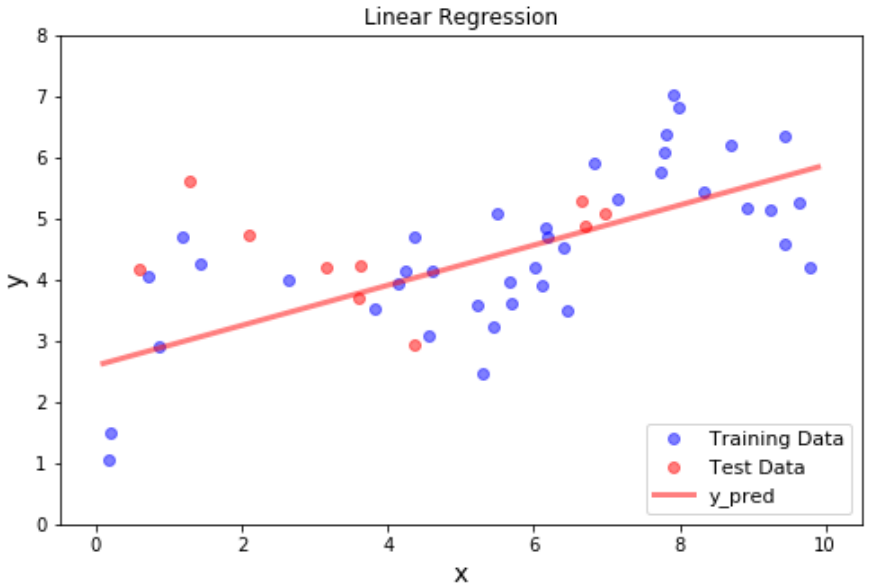


$$f(x) = \beta^T x + b$$

$$\min_{\beta} \sum_{i=1} (y_i - f(x))^2$$

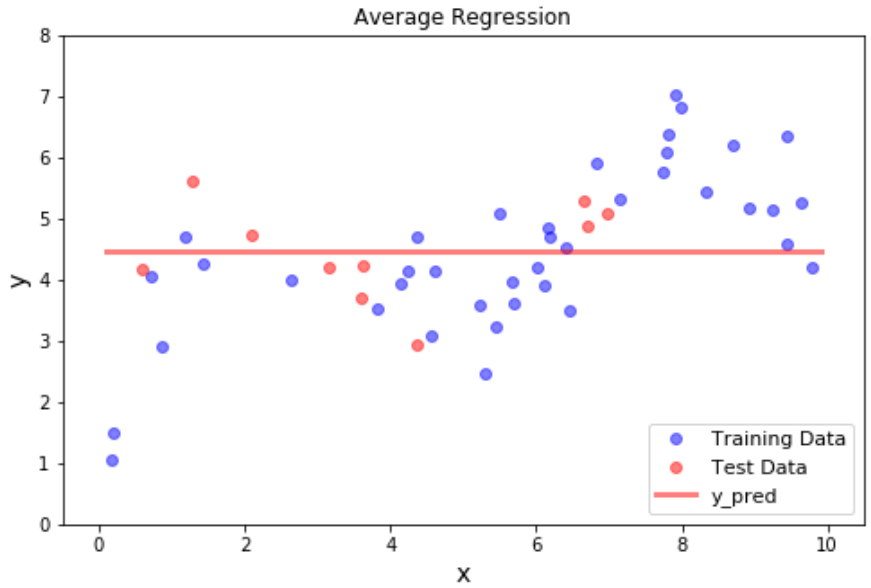
$$\hat{\beta} = (x^T x)^{-1} x^T y$$

Linear Regression



$$f(x) = 0.328x + 2.596$$

Training RMSE : 0.9516
Test RMSE : 1.1319

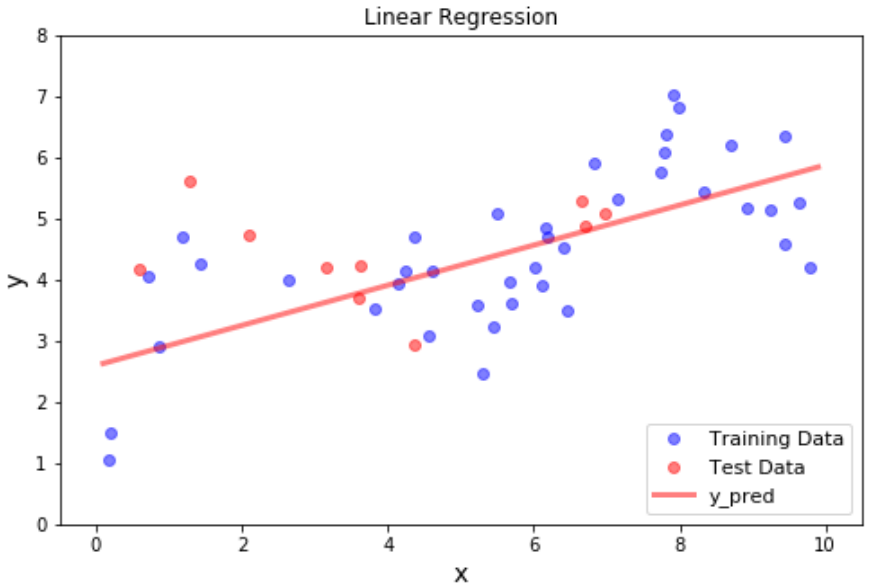


$$f(x) = 0x + 4.482$$

Training RMSE : 1.3056
Test RMSE : 0.7576

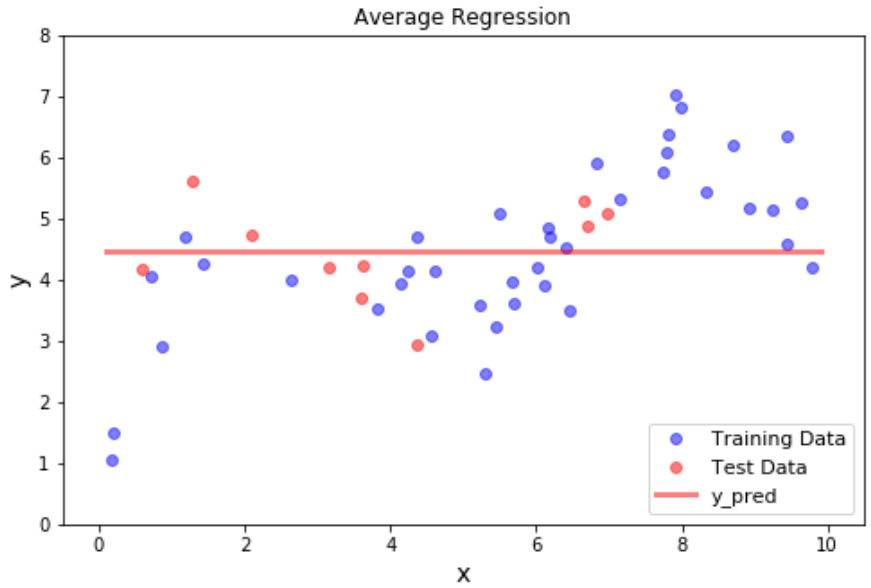
Large Weights

Large weights tend to cause sharp transitions in the node functions and thus large changes in output for small changes in the inputs.



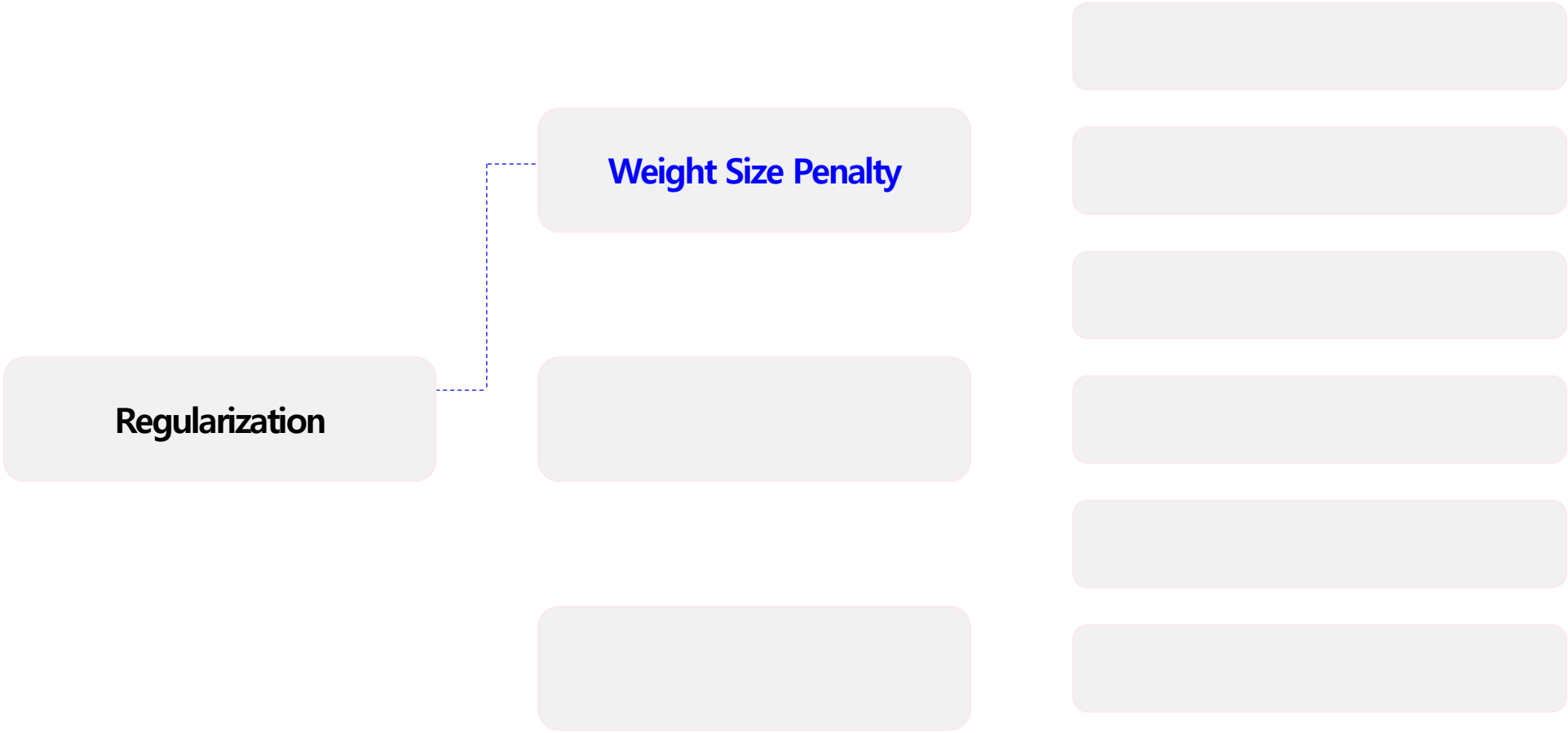
$$f(x) = 0.328x + 2.596$$

Training RMSE : 0.9516
Test RMSE : 1.1319



$$f(x) = 0x + 4.482$$

Training RMSE : 1.3056
Test RMSE : 0.7576


Regularization
Weight Size Penalty

Weight Size Penalty

A solution to overfitting problem is to update the learning algorithm to encourage the network to keep the weights small.

$$\min_{\beta} \sum_{i=1} (y_i - f(x))^2$$

$$subject\ to \sum_{j=1}^{p} \beta^2 < t$$

Euclidean distance, L2-norm

Ridge Regression

$$subject\ to \sum_{j=1}^{p} |\beta| < t$$

Manhattan distance, L1-norm

Lasso Regression

Ridge Regression

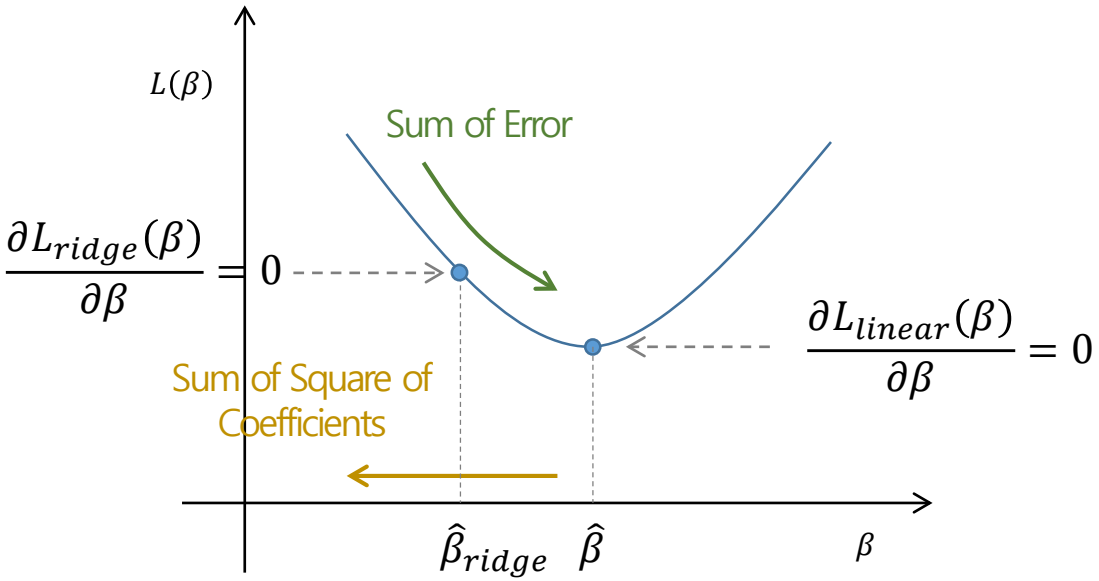
$$\min_{\beta} \sum_{i} (y_i - f(x))^2$$

$$subject\ to \sum_{j} \beta^2 < t$$

$$=$$

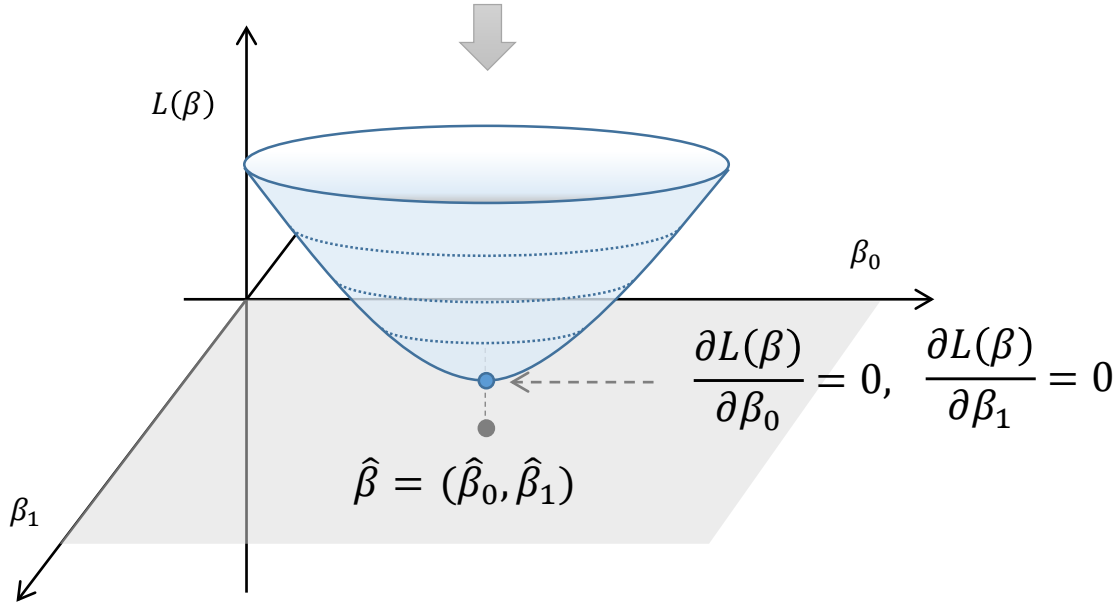
$$\min_{\beta} \underbrace{\sum_{i} (y_i - f(x))^2}_{\text{Sum of Error}} + \lambda \underbrace{\sum_{j} \beta^2}_{\text{Sum of Square of Coefficients}}$$

Ridge Regression



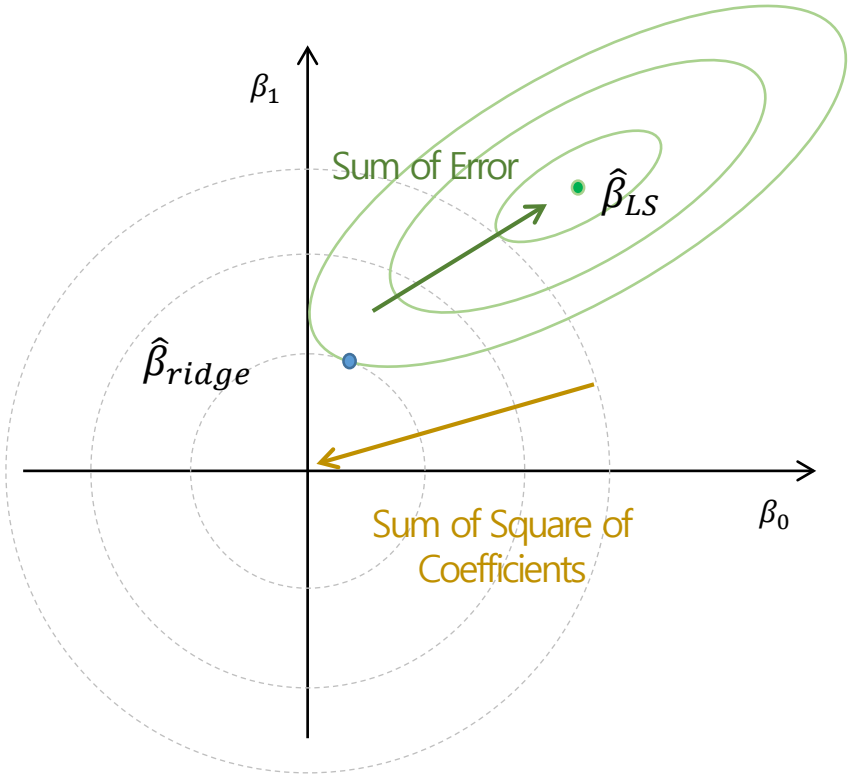
$$\min_{\beta} \underbrace{\sum_{i}(y_i - f(x))^2}_{\text{Sum of Error}} + \lambda \underbrace{\sum_{j}\beta^2}_{\text{Sum of Square of Coefficients}}$$

Ridge Regression



$$\min_{\beta} \underbrace{\sum_i (y_i - f(x))^2}_{\text{Sum of Error}} + \lambda \underbrace{\sum_j \beta^2}_{\text{Sum of Square of Coefficients}}$$

Ridge Regression



$$\min_{\beta} \sum_{i} (y_i - f(x))^2 + \lambda \sum_{j} \beta^2$$

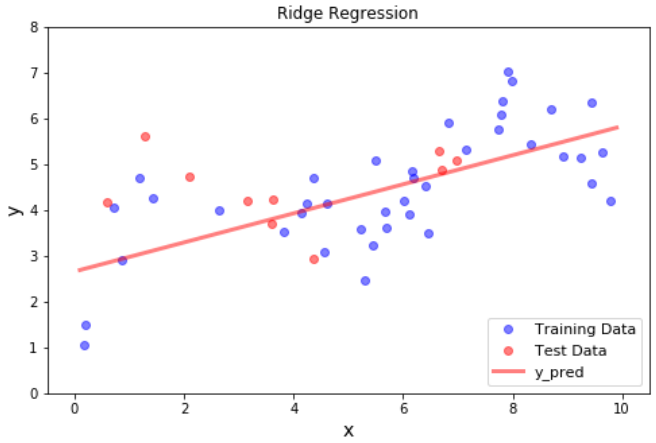
Sum of Error

Sum of Square of Coefficients

Ridge Regression

$$\min_{\beta} \sum_{i} (y_i - f(x))^2 + \lambda \sum_{j} \beta^2$$

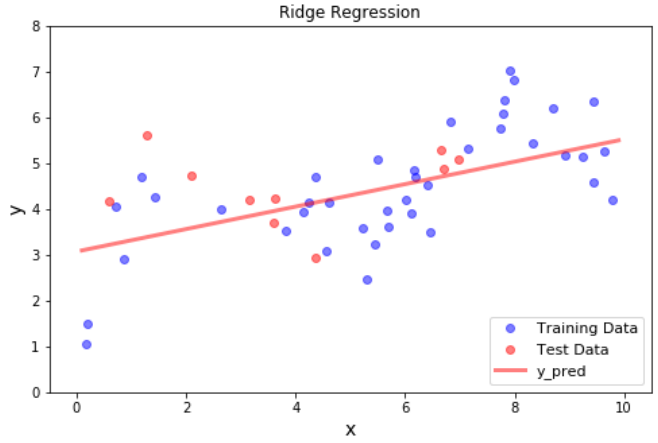
$\lambda = 10$



$f(x) = 0.317x + 2.657$

Training RMSE : 0.9521
Test RMSE : 1.1094

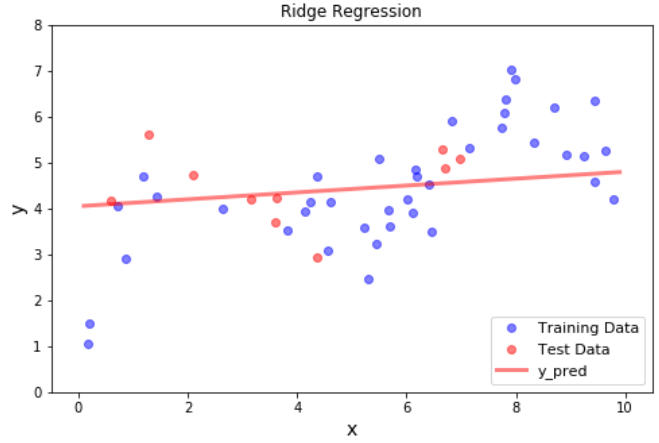
$\lambda = 100$



$f(x) = 0.245x + 3.072$

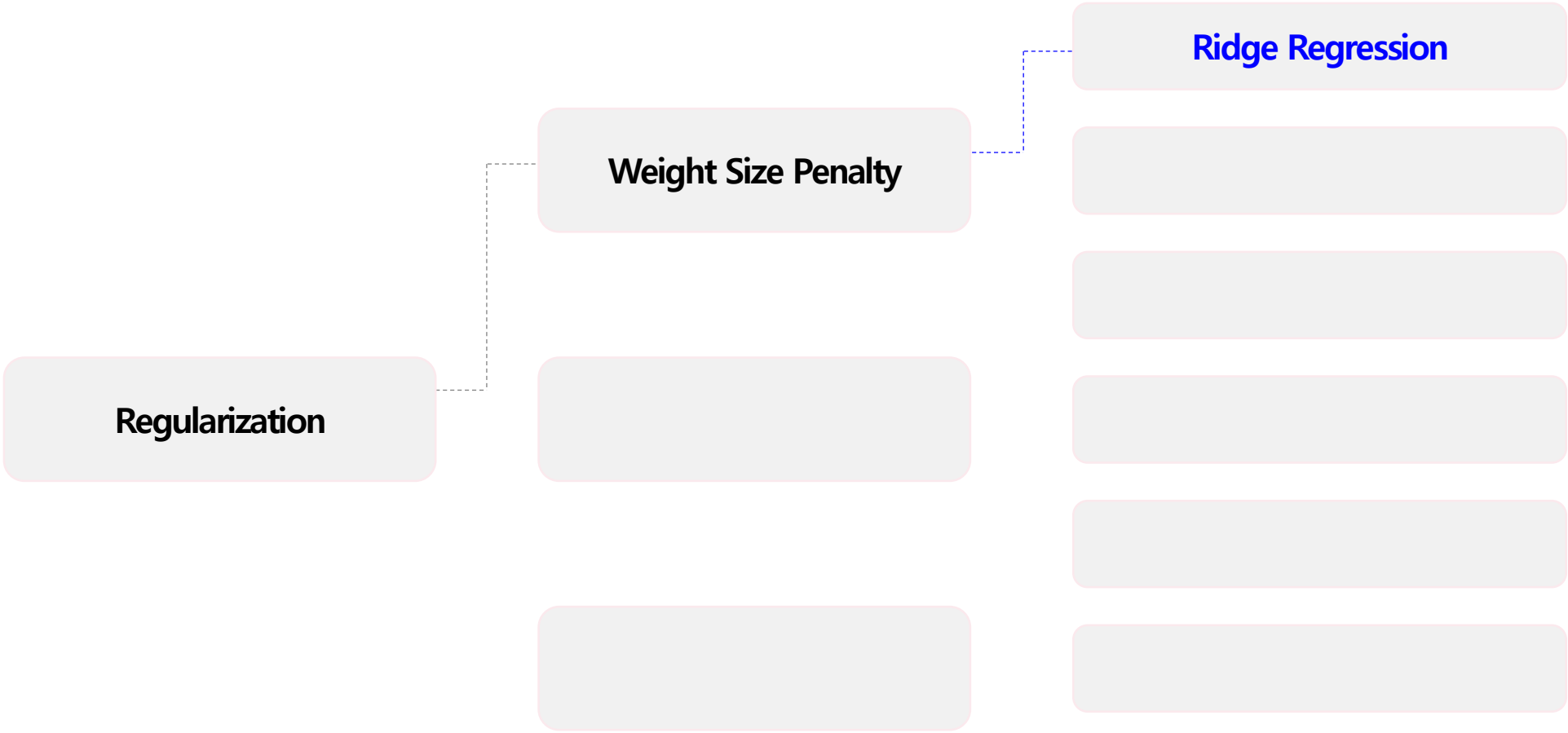
Training RMSE : 0.9779
Test RMSE : 0.9692

$\lambda = 1000$

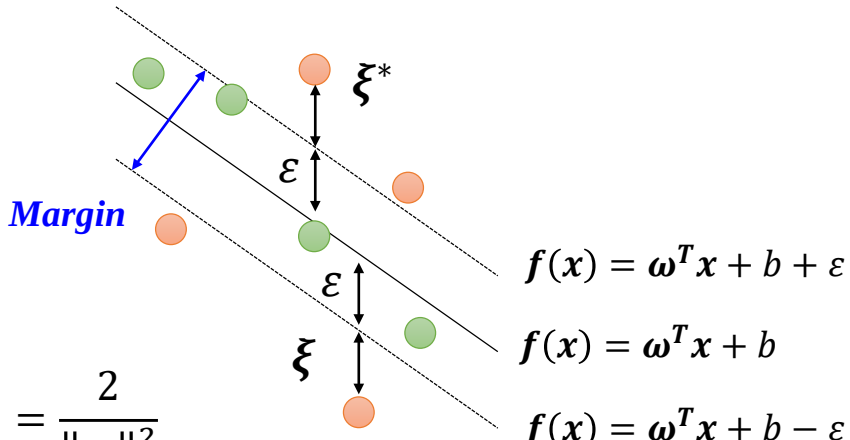


$f(x) = 0.075x + 4.051$

Training RMSE : 1.1750
Test RMSE : 0.7633


Regularization
Weight Size Penalty
Ridge Regression

Support Vector Regression



$$Margin = \frac{2}{\|\omega\|_2^2}$$

Maximize margin

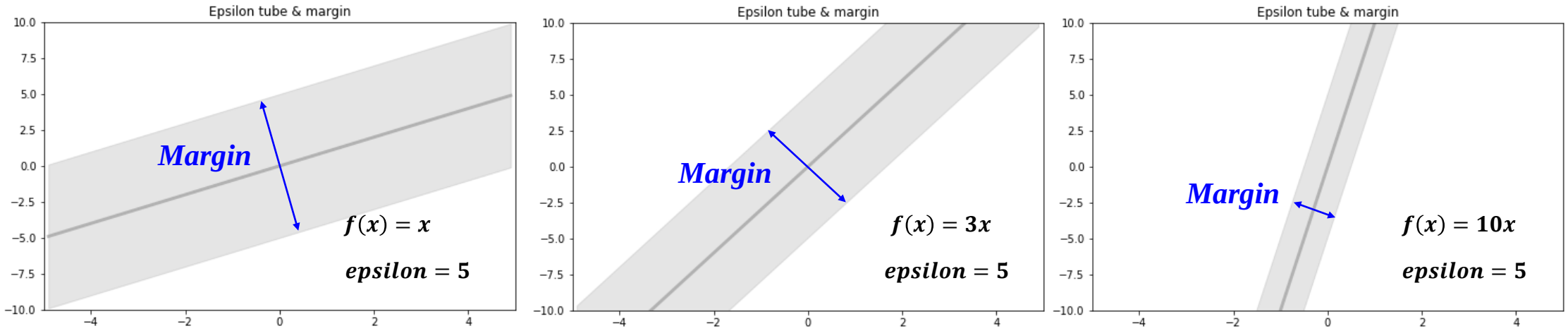
$$Minimize \quad \frac{1}{2}\|\omega\|_2^2 + C\sum_i (\xi_i + \xi_i^*)$$

$$subject\ to \quad (\omega^T x + b) - y_i \leq \varepsilon + \xi_i$$

$$y_i - (\omega^T x + b) \leq \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \geq 0, \qquad i = 1, \cdots, n$$

Maximize Margin



Support Vector Regression

Support Vector Regression

$$f(x) = \boldsymbol{\omega}^{\boldsymbol{T}}\boldsymbol{x} + b$$

$$\min_{\boldsymbol{\omega}} \ \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}(\xi_i + \xi_i^*)$$

Maximize margin — Minimize error

= Minimize Coefficient

= Regularized coefficient

Ridge Regression

$$f(x) = \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x} + b$$

$$\min_{\beta} \sum_{i=1}(y_i - \hat{y}_i)^2 + \lambda\|\boldsymbol{\beta}\|^2$$

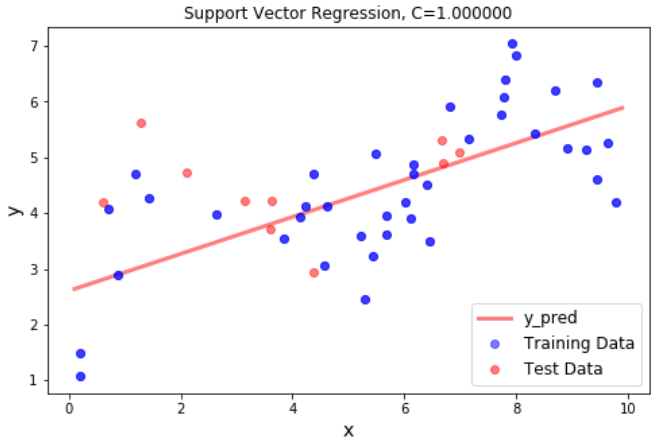
Minimize error — Minimize Coefficient

= Regularized coefficient

Support Vector Regression

$$\min_{\omega} \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}(\xi_i + \xi_i^*)$$

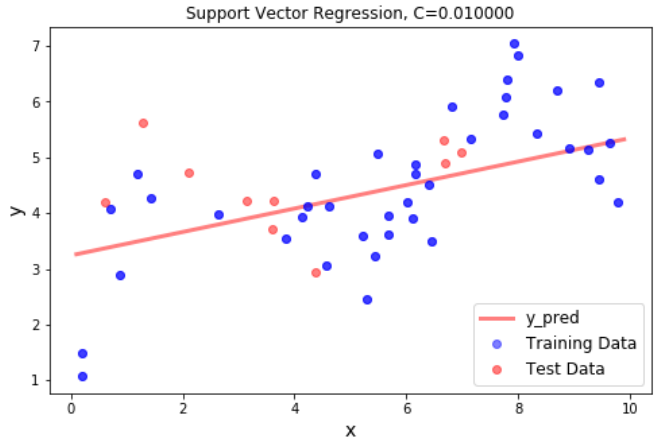
$C = 1.0$



$f(x) = 0.331x + 2.605$

Training RMSE : 0.9521
Test RMSE : 1.1240

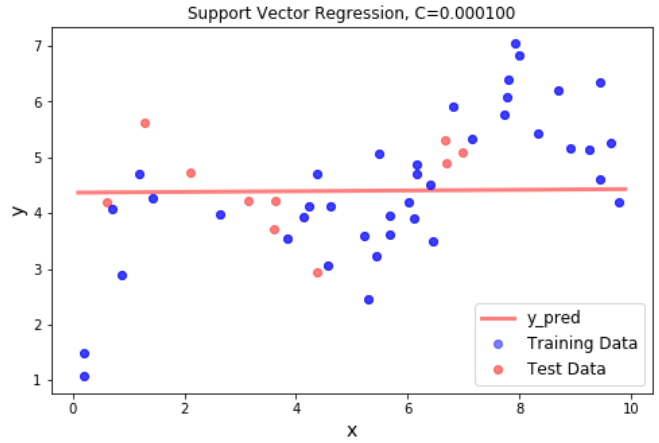
$C = 0.01$



$f(x) = 0.210x + 3.240$

Training RMSE : 1.0048
Test RMSE : 0.9245

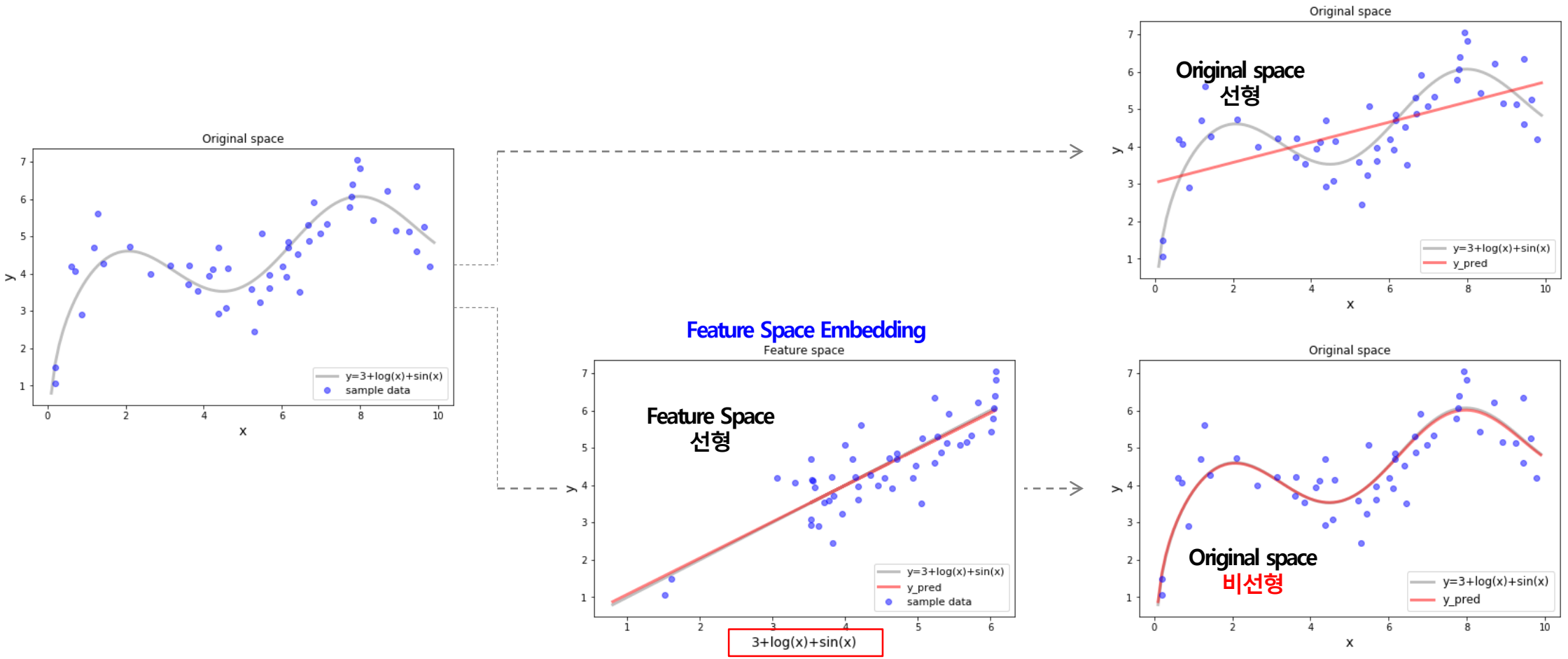
$C = 0.0001$



$f(x) = 0.006x + 4.367$

Training RMSE : 1.2962
Test RMSE : 0.7619

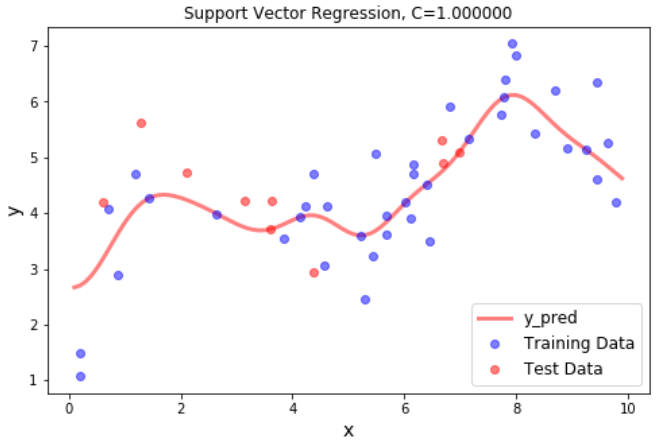
Kernel Trick



Support Vector Regression

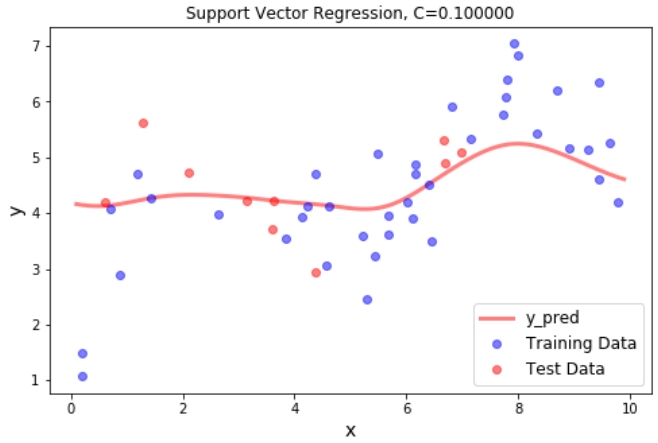
$$\min_{\omega} \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}(\xi_i + \xi_i^*)$$

$C = 1.0$



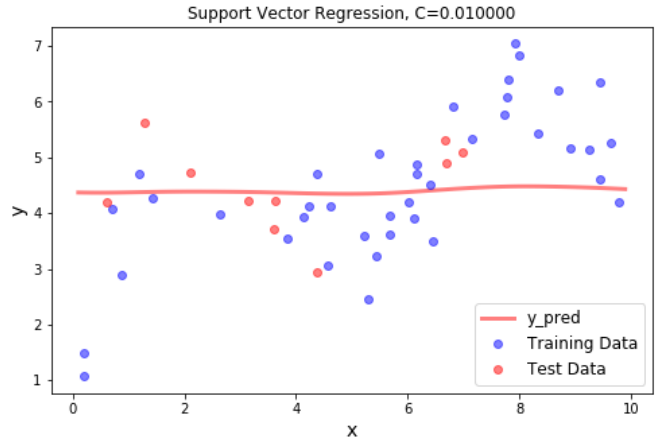
Training RMSE : 0.6681
Test RMSE : 0.7231

$C = 0.1$

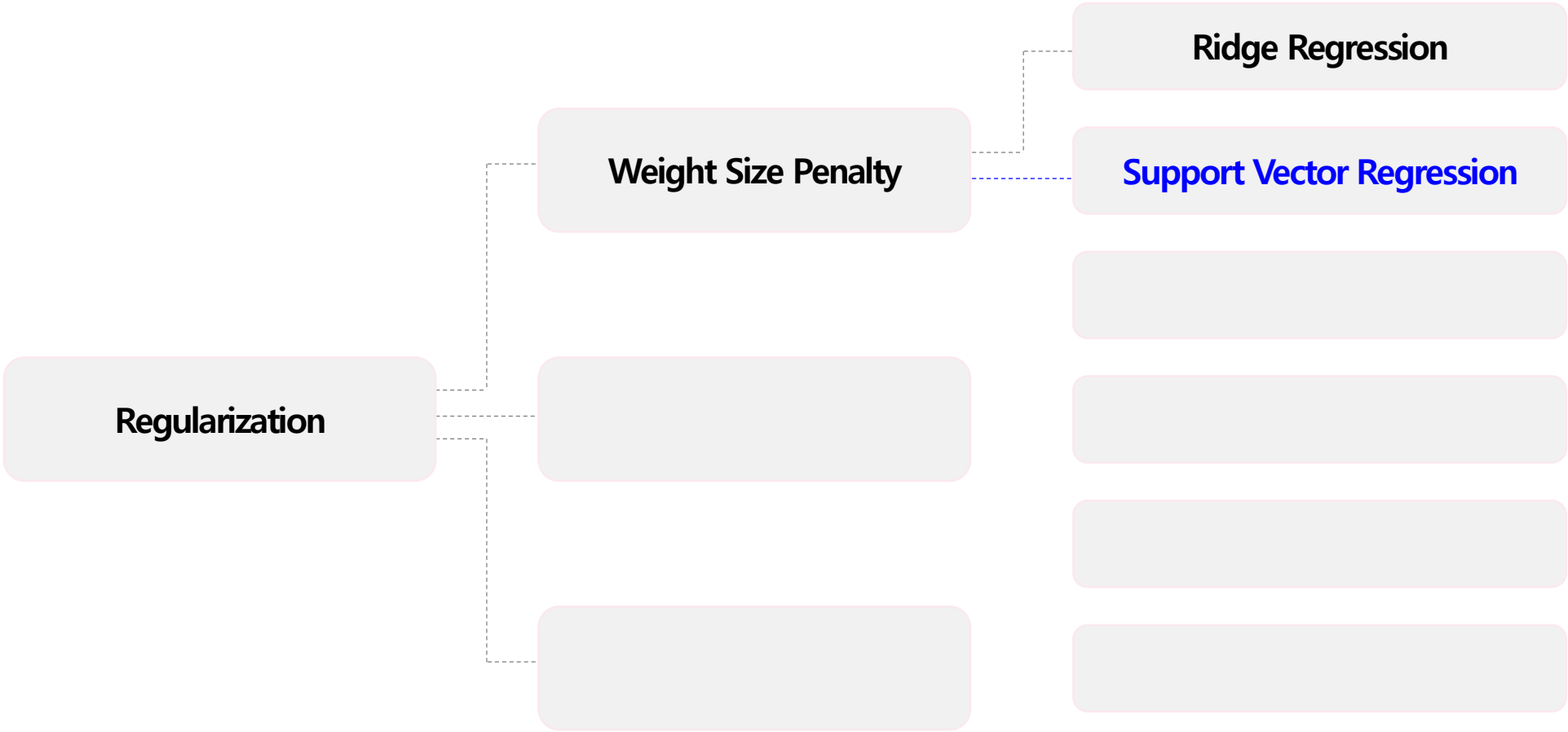


Training RMSE : 1.0105
Test RMSE : 0.6516

$C = 0.01$



Training RMSE : 1.1273
Test RMSE : 0.7452


Regularization
Weight Size Penalty
Ridge Regression
Support Vector Regression

LASSO Regression

least absolute shrinkage and selection operator

$$\min_{\beta} \sum_{i=1} (y_i - f(x))^2$$

$$subject\ to \sum_{j=1}^{p} |\beta| < t$$

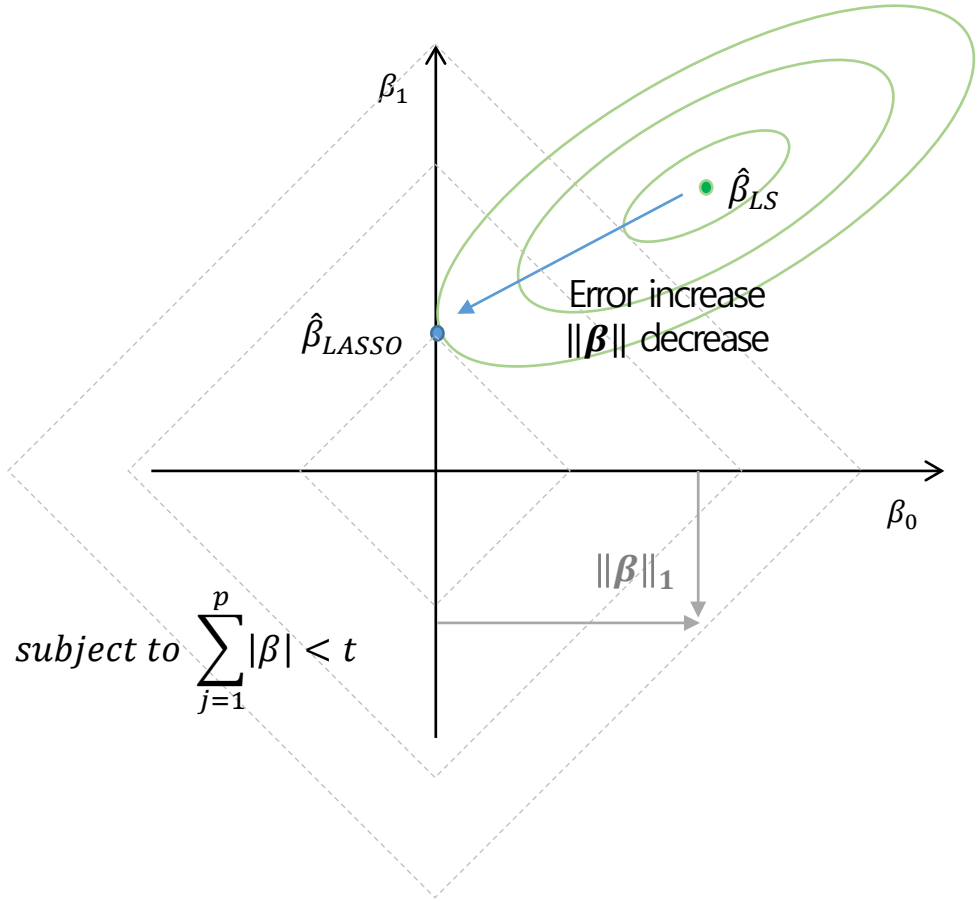
$$=$$

$$\min_{\beta} \sum_{i=1} (y_i - f(x))^2 + \lambda \sum_{j=1}^{p} |\beta|$$

Sum of Error

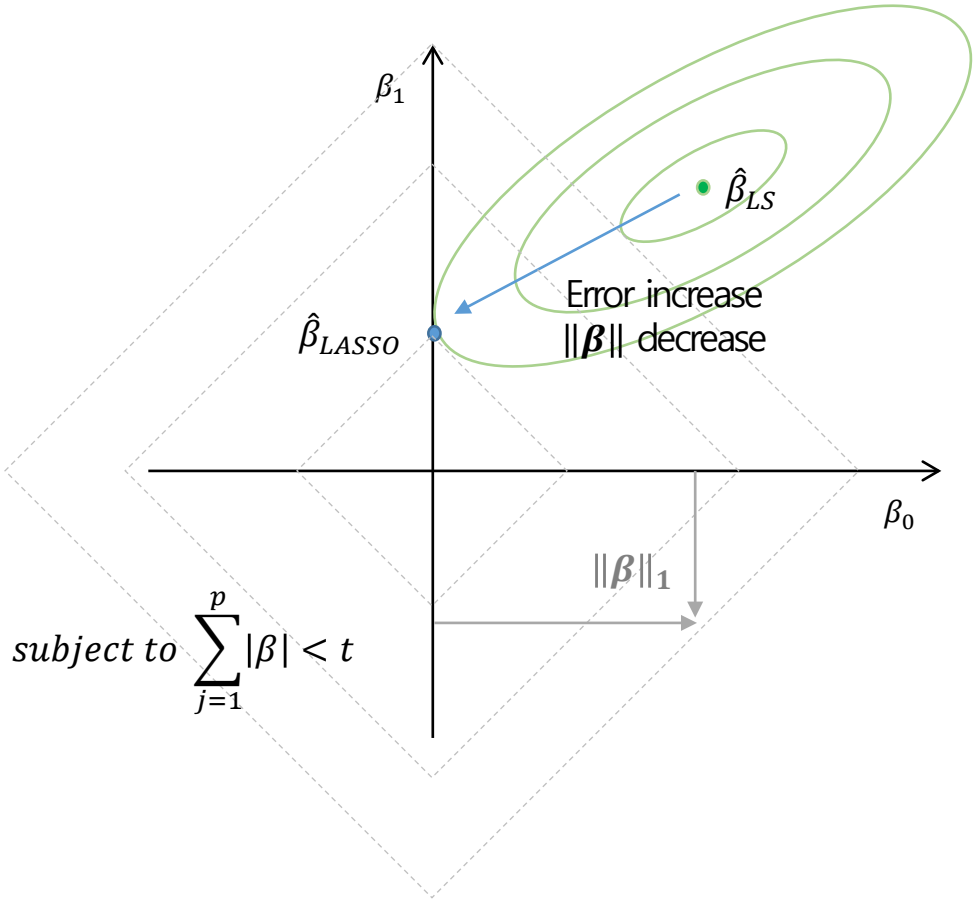
Sum of Absolute Coefficients

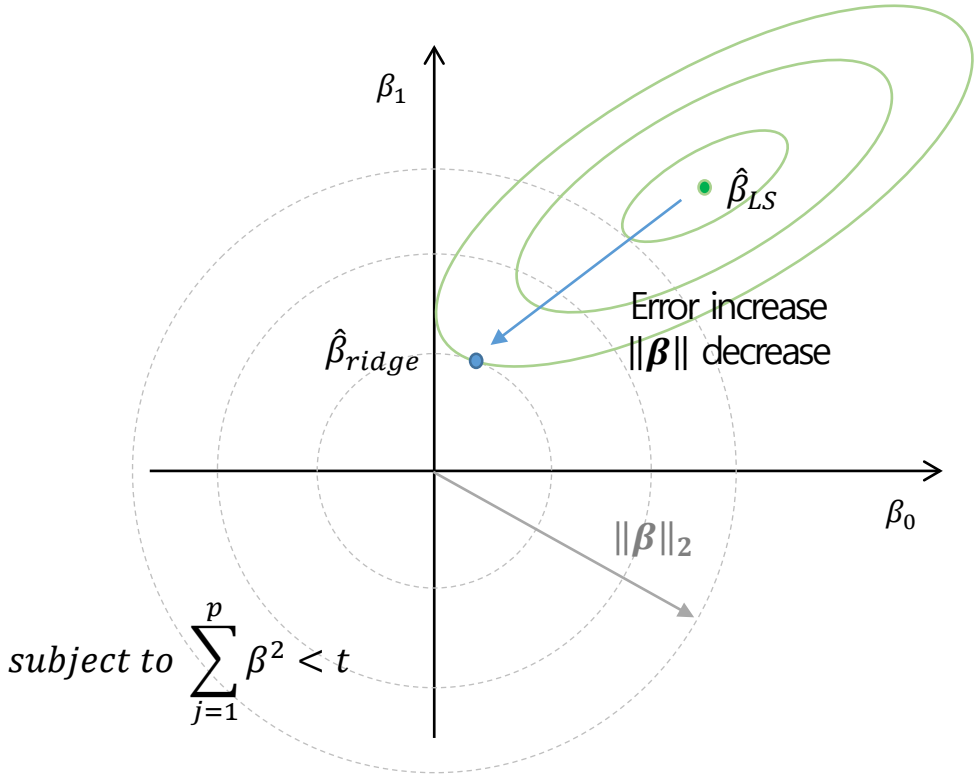
LASSO Regression



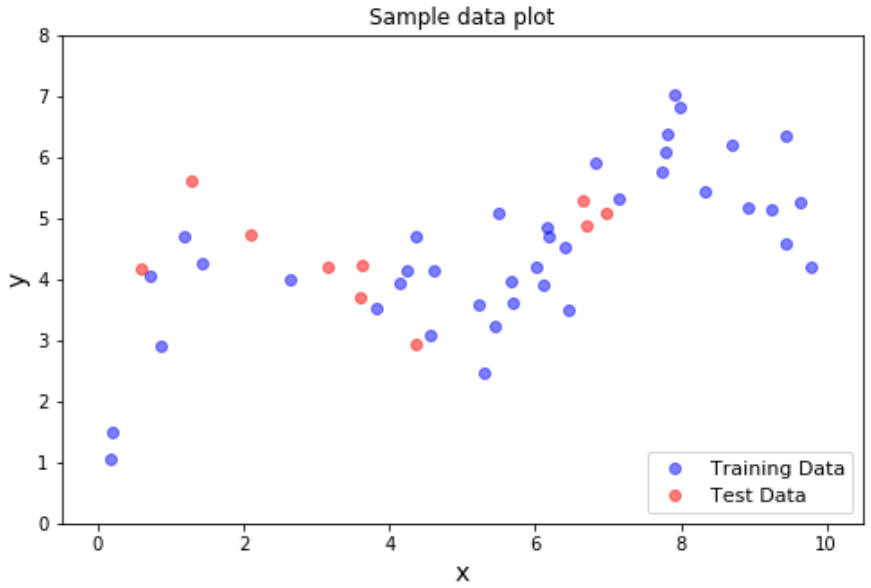
$$\min_{\beta} \underbrace{\sum_{i=1} (y_i - f(x))^2}_{\text{Sum of Error}} + \lambda \underbrace{\sum_{j=1}^{p} |\beta|}_{\text{Sum of Absolute Coefficients}}$$

LASSO Regression





LASSO Regression



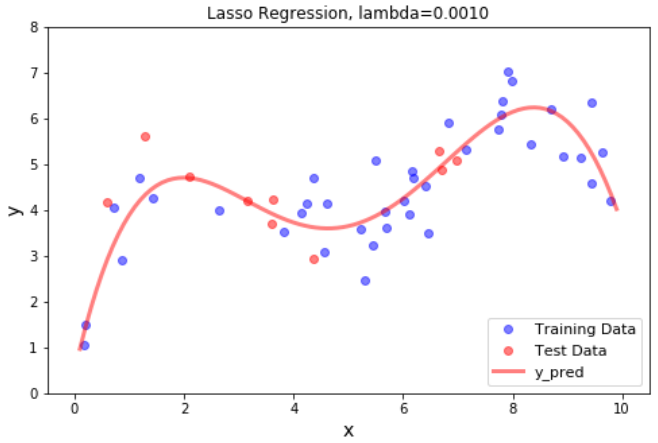
$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \beta_5 x^5 + \beta_6 x^6$$

$$\min_{\beta} \sum_{i} (y_i - f(x))^2 + \lambda \sum_{j} |\beta_j|$$

LASSO Regression

$$\min_{\beta} \sum_{i} (y_i - f(x))^2 + \lambda \sum_{j} |\beta_j|$$

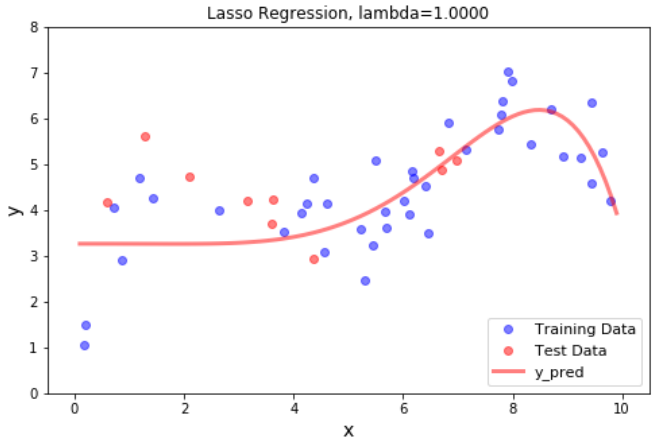
$\lambda = 0.001$



$$f(x) = 0.0001x^6 - 0.0018x^5 - 0.0005x^4 + 0.2737x^3 - 2.1021x^2 + 5.2628x + 0.4608$$

Training RMSE : 0.6183
Test RMSE : 0.6352

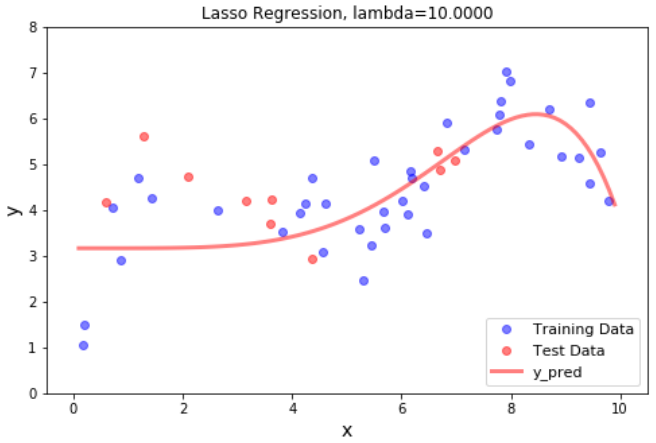
$\lambda = 1$



$$f(x) = -0.0001x^6 + 0.0007x^5 - 0.0014x^4 + 3.2642$$

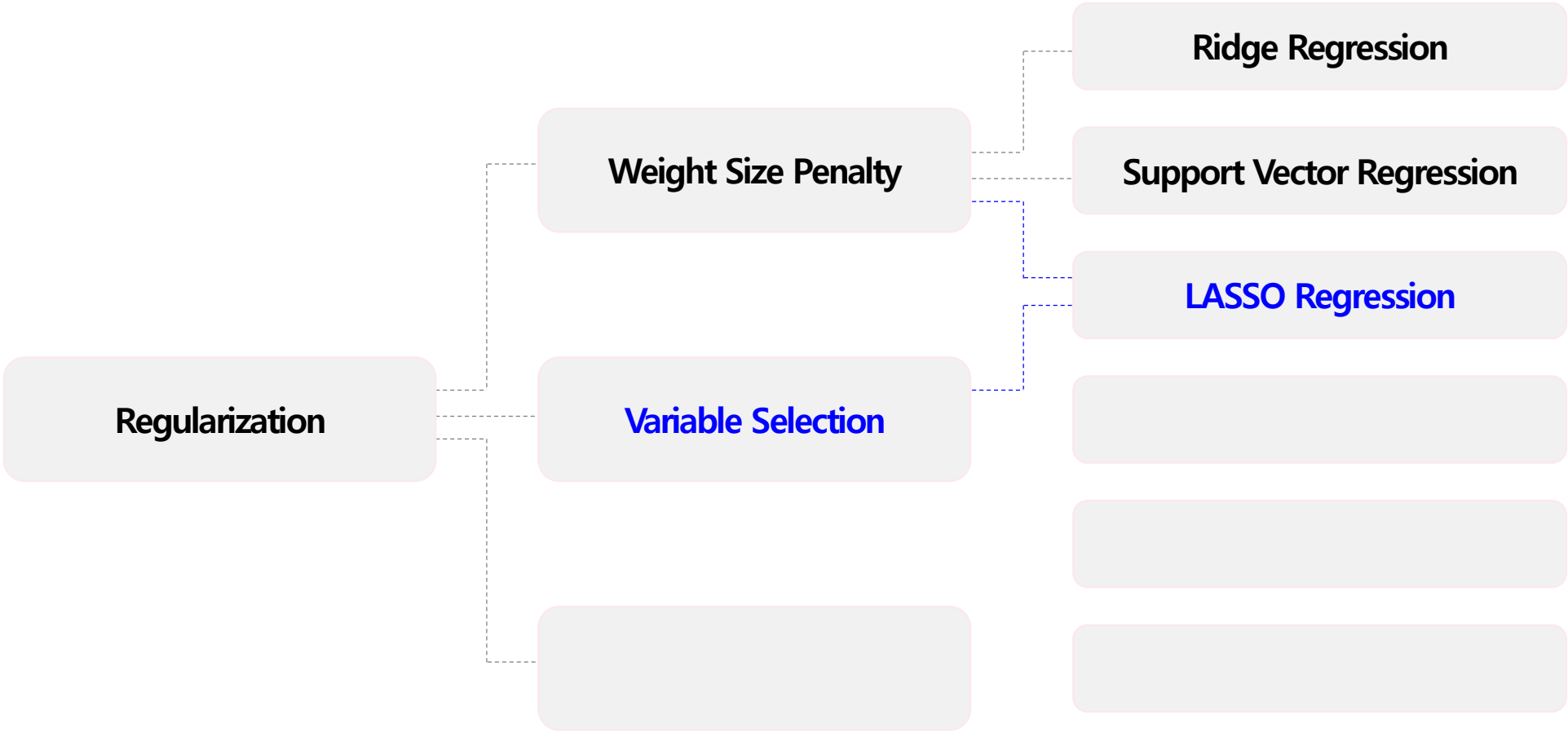
Training RMSE : 0.8264
Test RMSE : 1.0387

$\lambda = 10$



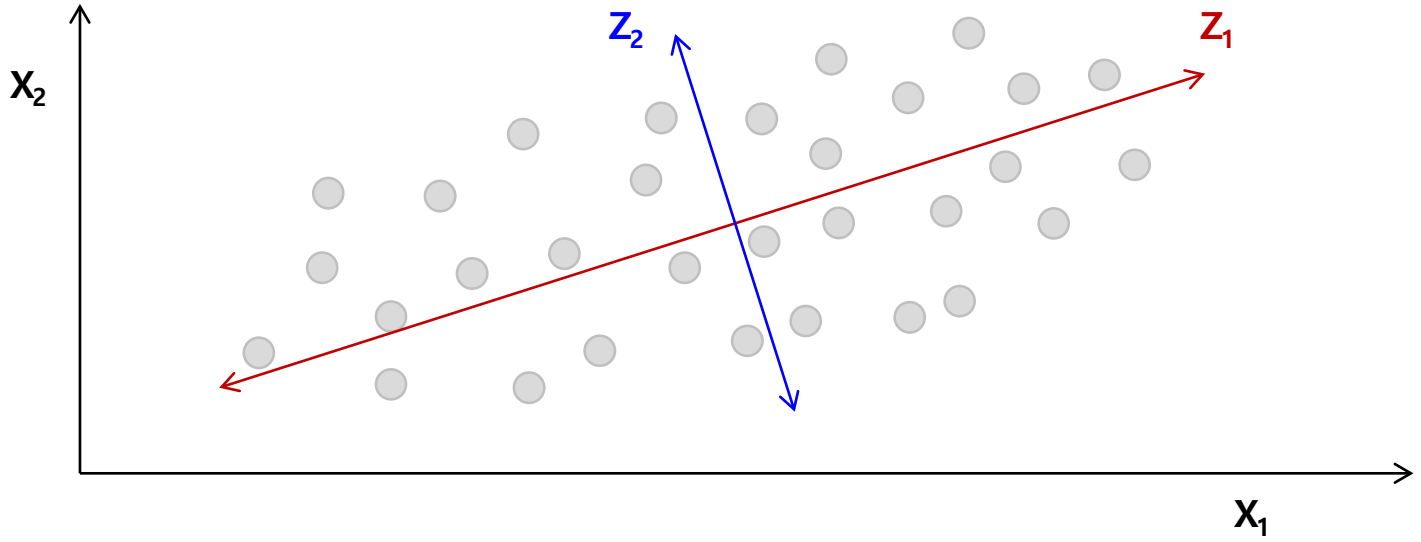
$$f(x) = -0.0000x^6 + 0.0004x^5 + 3.1664$$

Training RMSE : 0.8313
Test RMSE : 1.0872


Regularization
Weight Size Penalty
Variable Selection
Ridge Regression
Support Vector Regression
LASSO Regression

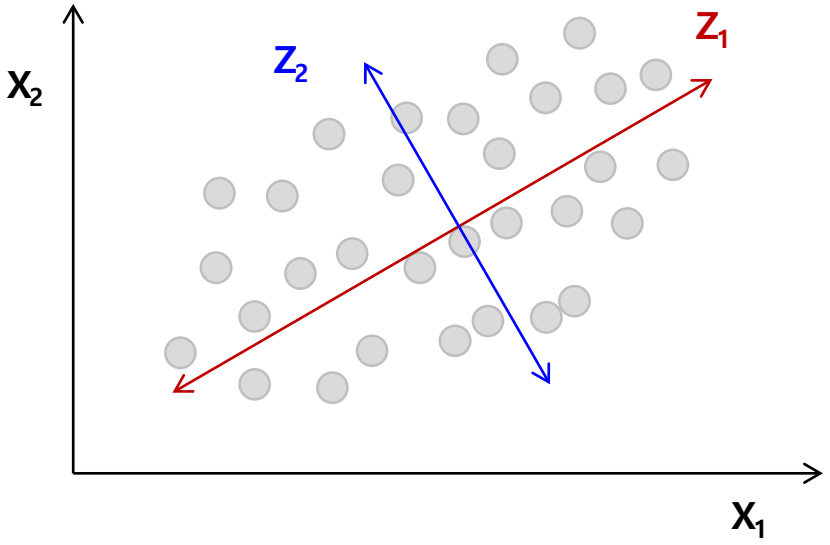
Principal Component Analysis

To find a set orthogonal bases(principal component)
to preserve the variance of the original data.

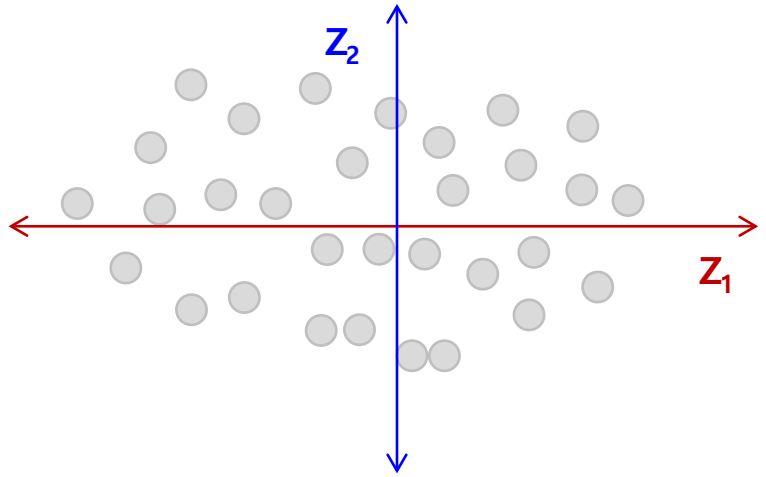


Principal Component Analysis

To find a set orthogonal bases(principal component)
to preserve the variance of the original data.

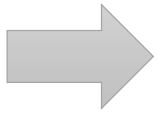


Principal Component
Space Embedding



Principal Component Analysis

	X_1	X_2	X_3	...	X_p
N_1	x_{11}	x_{12}	x_{13}	...	x_{1p}
N_2	x_{21}	x_{22}	x_{23}	...	x_{2p}
N_3	x_{31}	x_{32}	x_{33}	...	x_{3p}
⋮	⋮	⋮	⋮		⋮
N_n	x_{n1}	x_{n2}	x_{n3}	...	x_{np}



Principal Component Space Embedding

	Z_1	Z_2	Z_3	...	Z_p
N_1	z_{11}	z_{12}	z_{13}	...	z_{1p}
N_2	z_{21}	z_{22}	z_{23}	...	z_{2p}
N_3	z_{31}	z_{32}	z_{33}	...	z_{3p}
⋮	⋮	⋮	⋮		⋮
N_n	z_{n1}	z_{n2}	z_{n3}	...	z_{np}

Principal Component Analysis

	Z_1	Z_2	Z_3	…	Z_p
N_1	z_{11}	z_{12}	z_{13}	…	z_{1p}
N_2	z_{21}	z_{22}	z_{23}	…	z_{2p}
N_3	z_{31}	z_{32}	z_{33}	…	z_{3p}
⋮	⋮	⋮	⋮		⋮
N_n	z_{n1}	z_{n2}	z_{n3}	…	z_{np}

$$Z_1 = e_1X = e_{11}X_1 + e_{12}X_2 + e_{13}X_3 + \cdots + e_{1p}X_p$$
$$Z_2 = e_2X = e_{21}X_1 + e_{22}X_2 + e_{23}X_3 + \cdots + e_{2p}X_p$$
$$Z_3 = e_3X = e_{31}X_1 + e_{32}X_2 + e_{33}X_3 + \cdots + e_{3p}X_p$$
$$\vdots$$
$$Z_p = e_pX = e_{p1}X_1 + e_{p2}X_2 + e_{p3}X_3 + \cdots + e_{pp}X_p$$

e_i : $ith\ eigen\ vector$, λ_i : $ith\ eigen\ value$, $\lambda_1 > \lambda_2 > \lambda_3 > \cdots > \lambda_p$

Principal Component Analysis

	Z_1	Z_2	Z_3	…	Z_p
N_1	z_{11}	z_{12}	z_{13}	…	z_{1p}
N_2	z_{21}	z_{22}	z_{23}	…	z_{2p}
N_3	z_{31}	z_{32}	z_{33}	…	z_{3p}
⋮	⋮	⋮	⋮		⋮
N_n	z_{n1}	z_{n2}	z_{n3}	…	z_{np}

$$Z_1 = e_1X = e_{11}X_1 + e_{12}X_2 + e_{13}X_3 + \cdots + e_{1p}X_p$$

$$Z_2 = e_2X = e_{21}X_1 + e_{22}X_2 + e_{23}X_3 + \cdots + e_{2p}X_p$$

$$Z_3 = e_3X = e_{31}X_1 + e_{32}X_2 + e_{33}X_3 + \cdots + e_{3p}X_p$$

$$\vdots$$

$$Z_p = e_pX = e_{p1}X_1 + e_{p2}X_2 + e_{p3}X_3 + \cdots + e_{pp}X_p$$

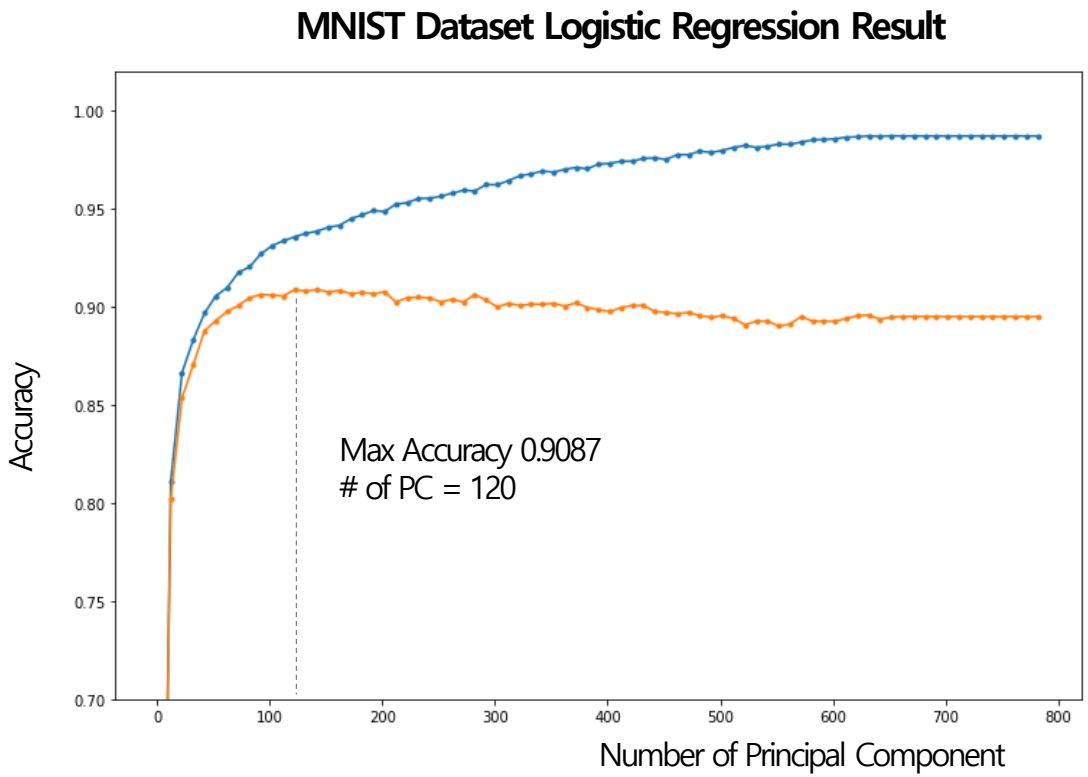
e_i : $ith\ eigen\ vector$, λ_i : $ith\ eigen\ value$, $\lambda_1 > \lambda_2 > \lambda_3 > \cdots > \lambda_p$

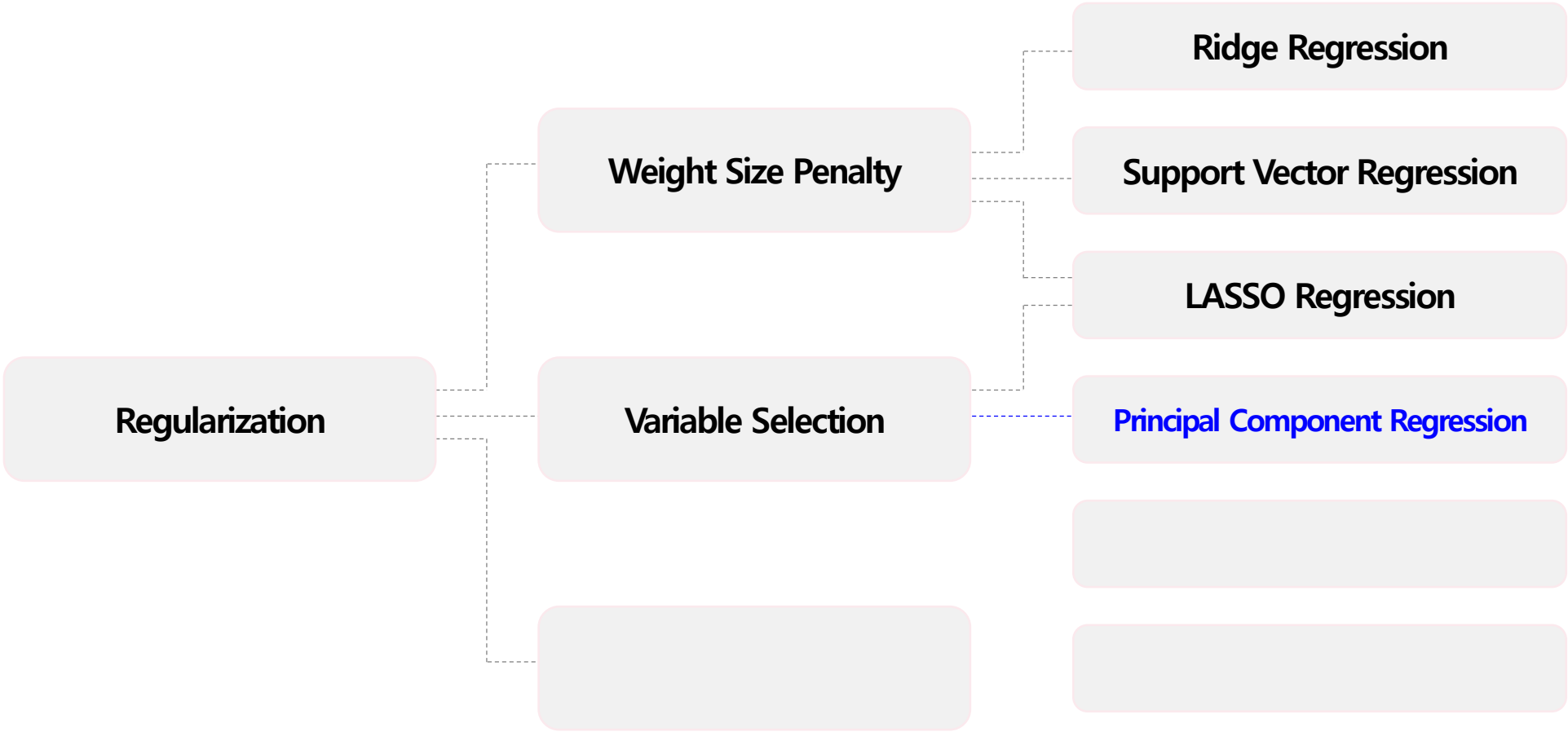
Principal Component Regression

Principal Component Regression

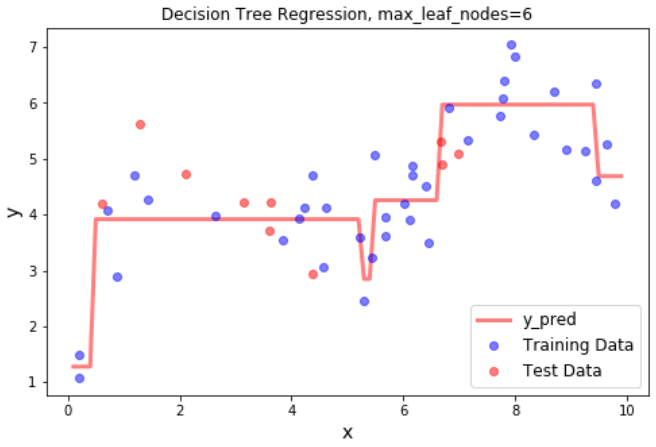
	Z_1	Z_2	Z_3	...	Z_p	Y
N_1	z_{11}	z_{12}	z_{13}	...	z_{1p}	y_1
N_2	z_{21}	z_{22}	z_{23}	...	z_{2p}	y_2
N_3	z_{31}	z_{32}	z_{33}	...	z_{3p}	y_3
⋮	⋮	⋮	⋮		⋮	⋮
N_n	z_{n1}	z_{n2}	z_{n3}	...	z_{np}	y_n

$$Y = Zq + \varepsilon$$

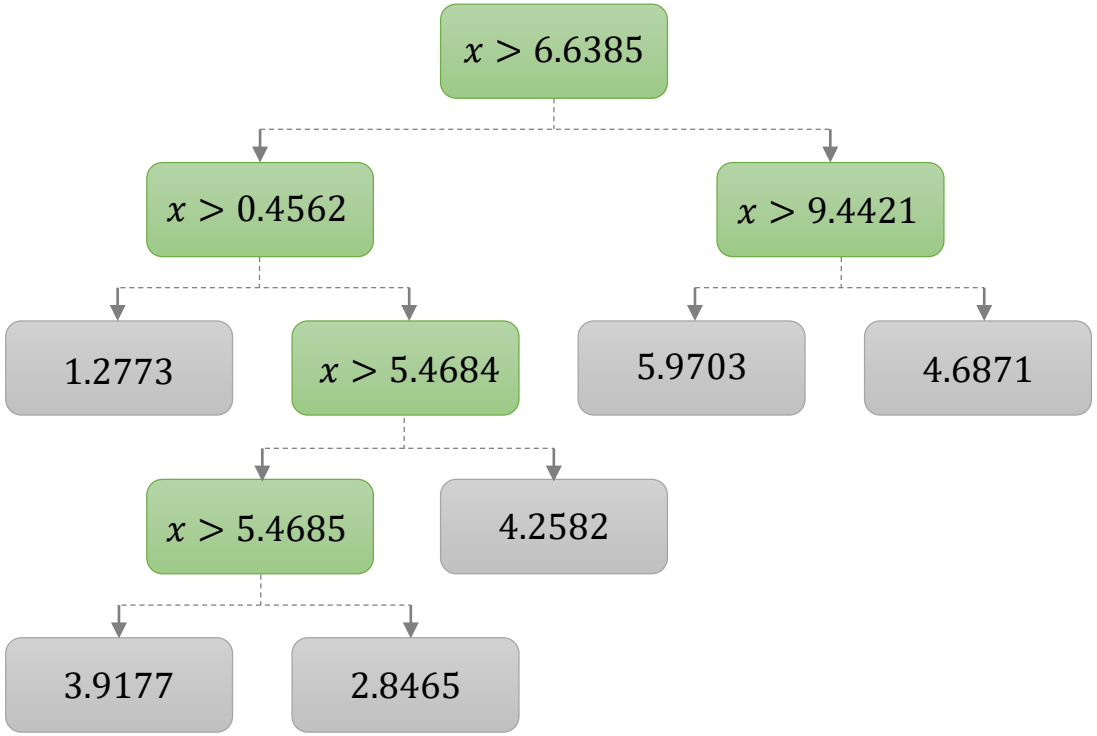



Regularization
Weight Size Penalty
Variable Selection
Ridge Regression
Support Vector Regression
LASSO Regression
Principal Component Regression

Decision Tree Regression

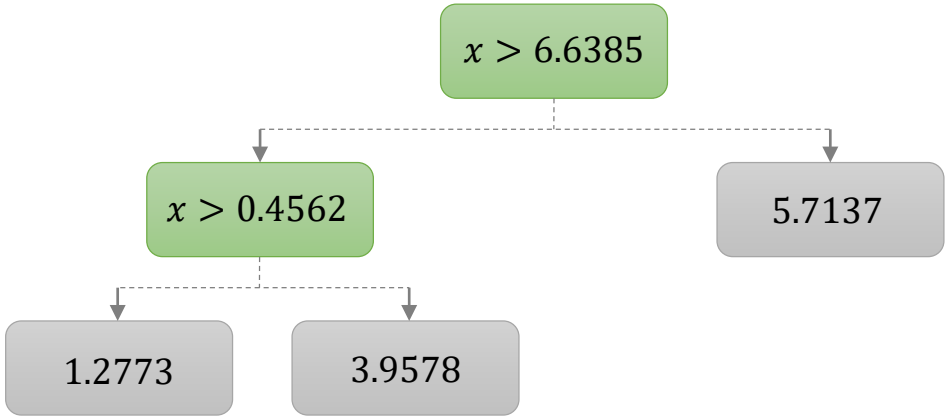


Training RMSE : 0.5319
Test RMSE : 0.8482



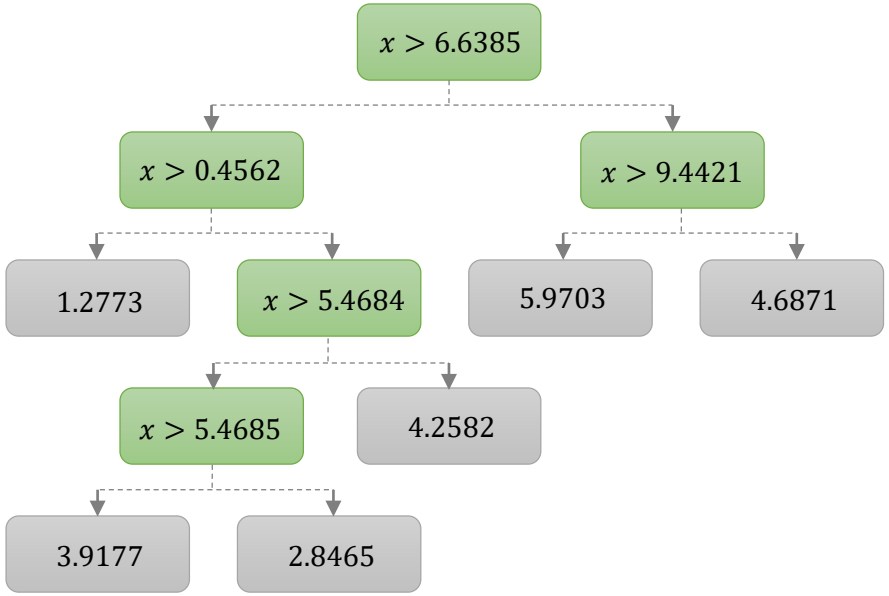
Decision Tree Regression

Max_leaf_node	Training RMSE	Test RMSE
2	0.8915	0.8381
3	0.6813	**0.7665**
4	0.6045	0.8383
5	0.5762	0.8963
6	0.5319	0.8482



Decision Tree Pruning

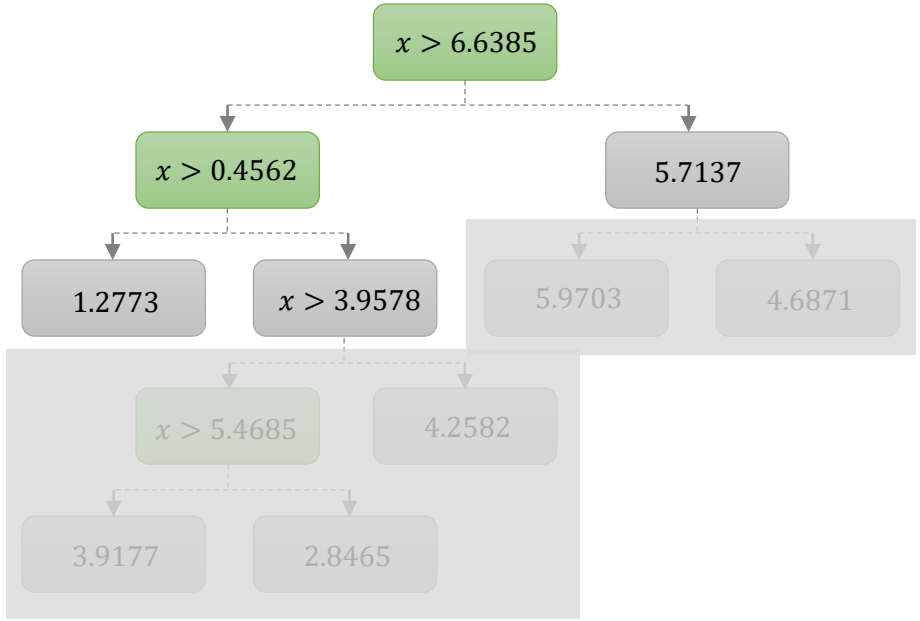
Decision Tree Regression



Decision Tree Space Embedding X → T

	T_1	T_2	T_3	T_3	T_3
	$x > 0.4562$	$x > 5.4684$	$x > 5.4685$	$x > 6.6385$	$x > 9.4421$
N_1	1	1	0	0	0
N_2	1	1	1	0	0
N_3	1	0	0	0	0
⋮					
N_n	1	1	1	1	0

Decision Tree Regression

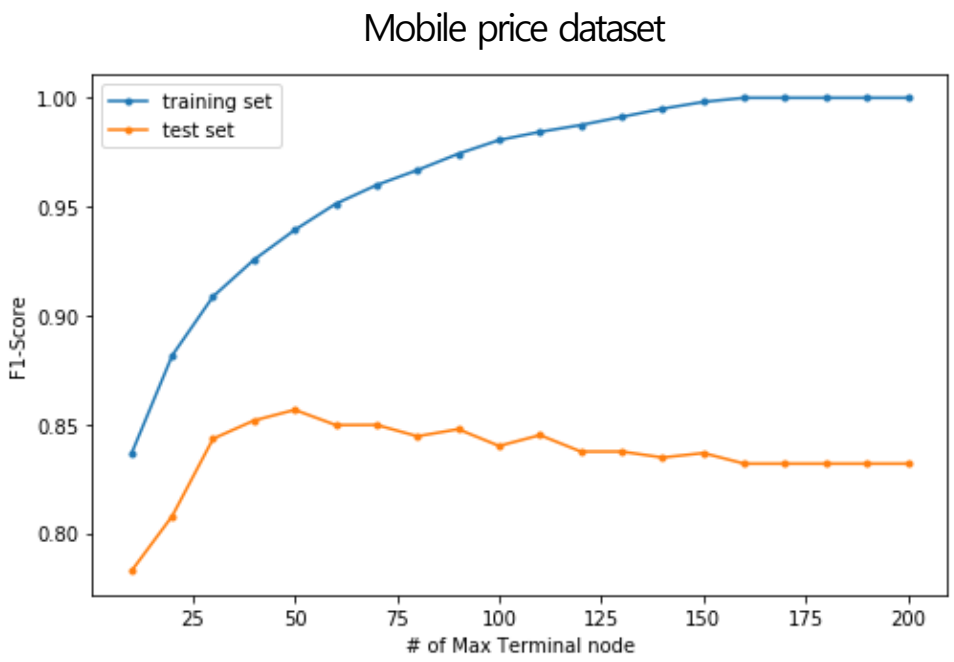


Decision Tree Space Embedding X → T

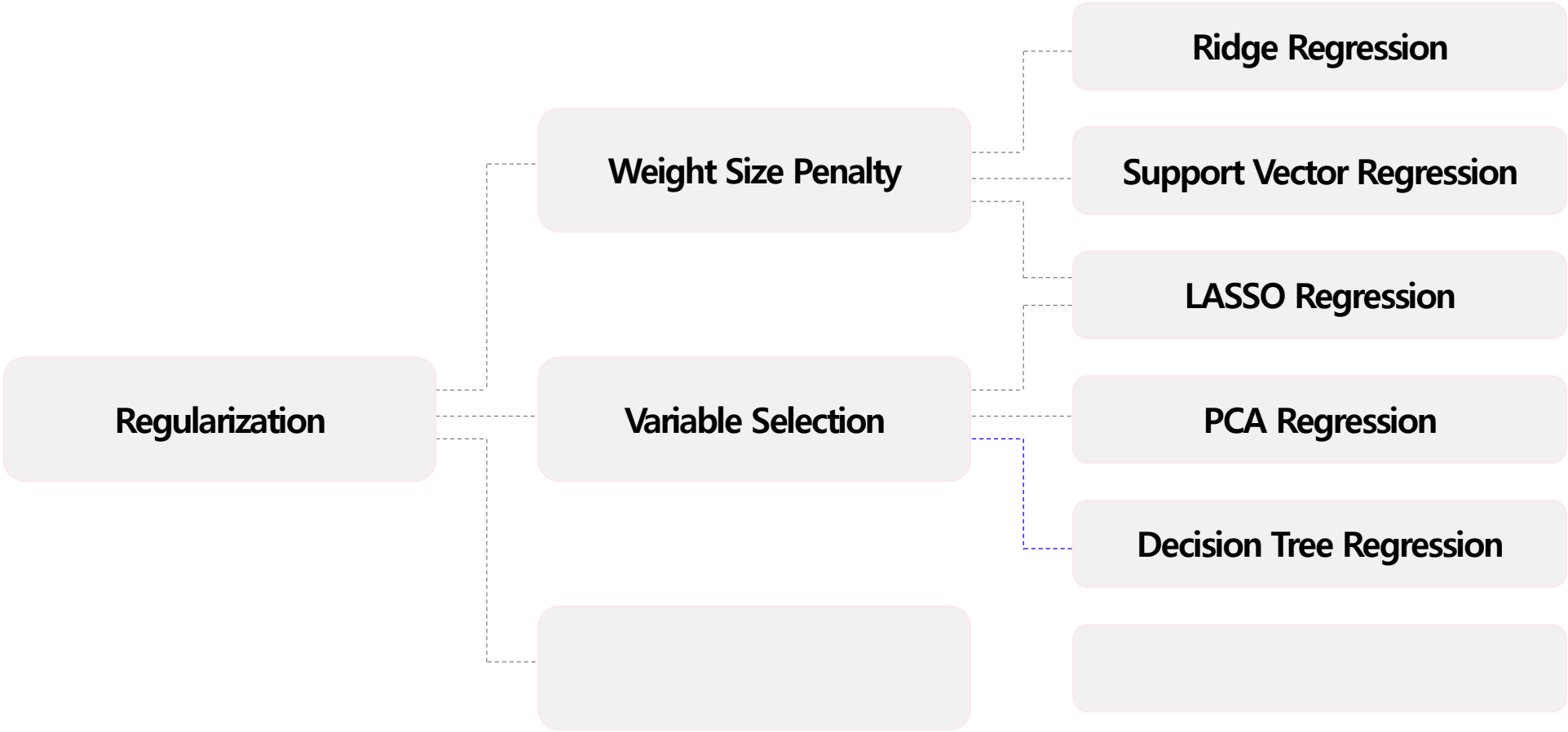
	T_1	T_2	T_3	T_4	T_5
	$x > 0.4562$	$x > 5.4684$	$x > 5.4685$	$x > 6.6385$	$x > 9.4421$
N_1	1	1	0	0	0
N_2	1	1	1	0	0
N_3	1	0	0	0	0
⋮					
N_n	1	1	1	1	0

Decision Tree Pruning = Variable Selection

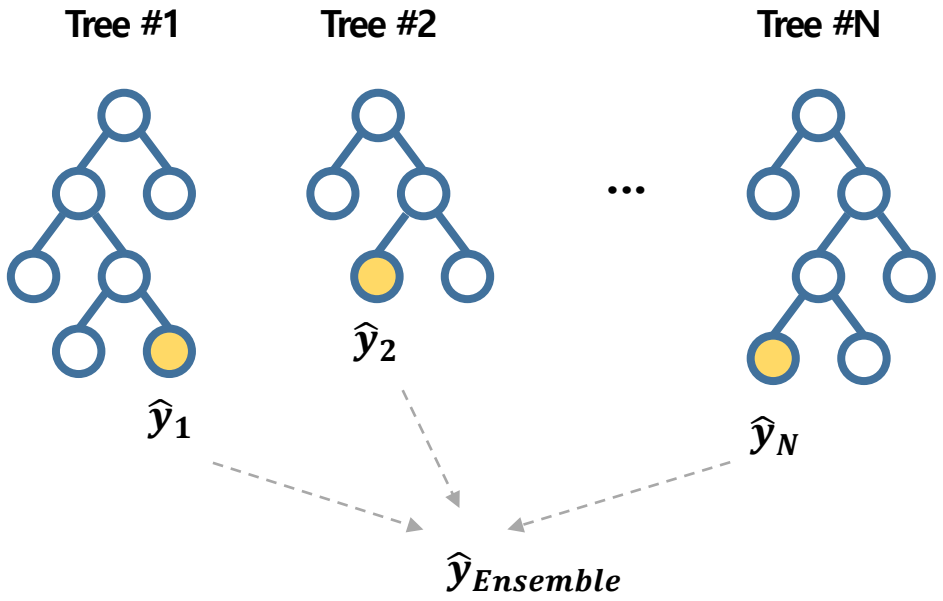
Decision Tree Regression



Dataset	끝노드 수 제한 없음				끝노드 수 = 50			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Training	1.000	1.000	1.000	1.000	0.939	0.939	0.939	0.939
Test	0.843	0.843	0.843	0.843	**0.858**	**0.855**	**0.858**	**0.857**


Regularization
Weight Size Penalty
Variable Selection
Ridge Regression
Support Vector Regression
LASSO Regression
PCA Regression
Decision Tree Regression

Ensemble = Collective Intelligence



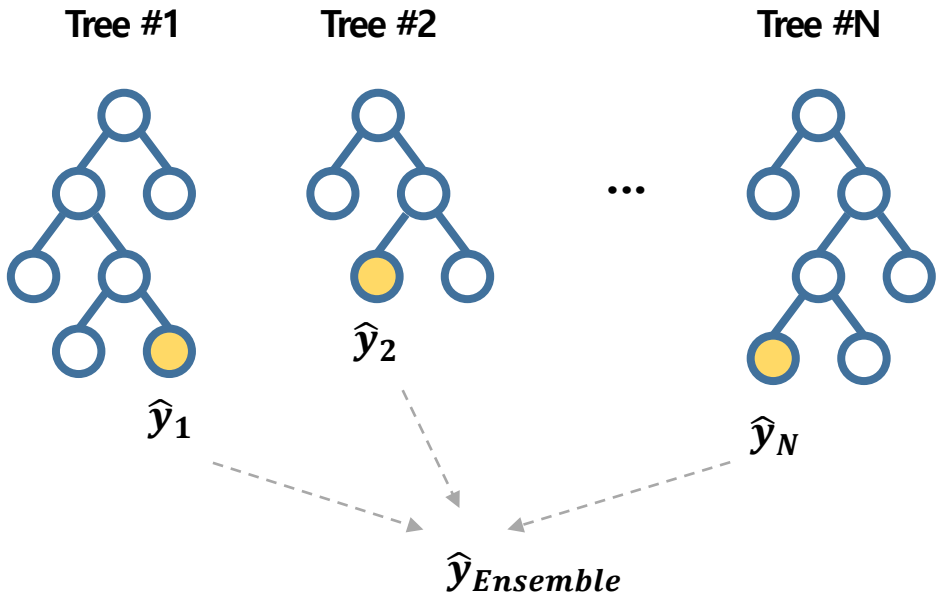
Majority voting

$$\hat{y}_{Ensemble} = \underset{i}{\text{argmax}} \sum_{j=1}^{N} \delta(\hat{y}_j = i), i \in \{0,1\}$$

Weighted voting

$$\hat{y}_{Ensemble} = \underset{i}{\text{argmax}} \frac{\sum_{j=1}^{N} TrnAcc_j \cdot \delta(\hat{y}_j = i)}{\sum_{j=1}^{N} TrnAcc_j}, i \in \{0,1\}$$

Ensemble = Collective Intelligence



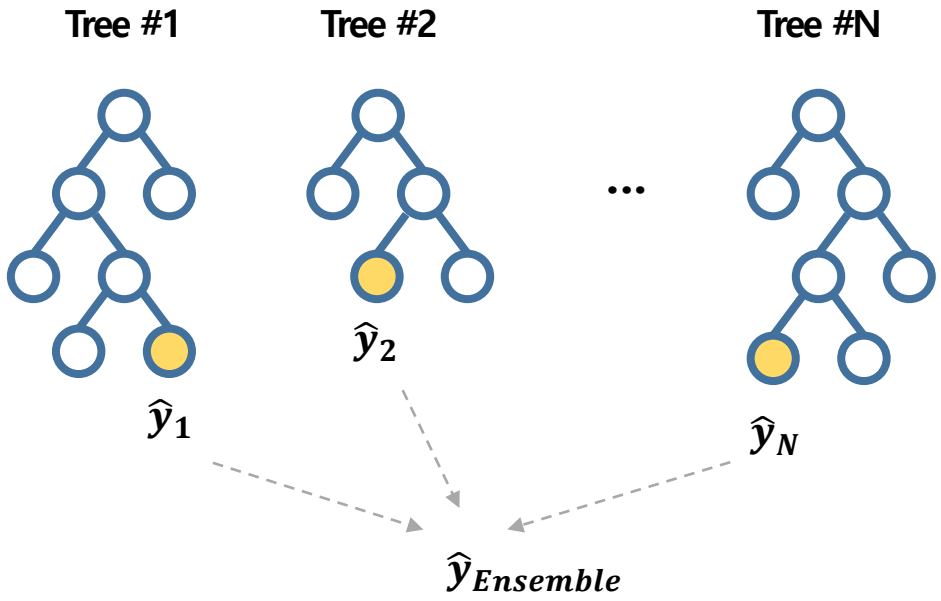
Averaging

$$\hat{y}_{Ensemble} = \frac{1}{N}\sum_{j=1}^{N}\hat{y}_j$$

Weighted Averaging

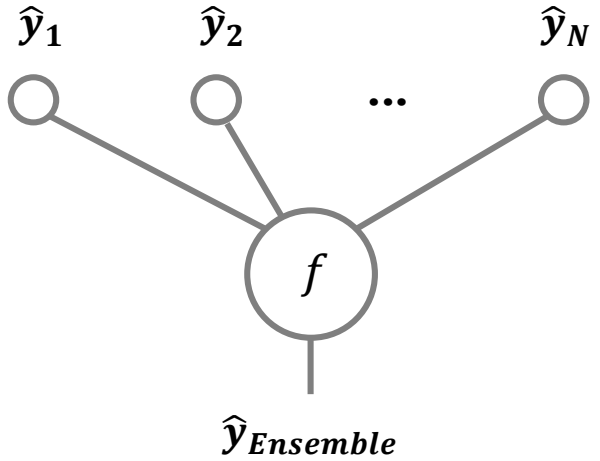
$$\hat{y}_{Ensemble} = \frac{1}{N}\frac{\sum_{j=1}^{N} TrnAcc_j \cdot \hat{y}_j}{\sum_{j=1}^{N} TrnAcc_j}$$

Ensemble = Collective Intelligence



Blending

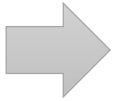
$$\hat{y}_{Ensemble} = f_{Blening}(y_1, y_2, \cdots, y_N)$$



Bootstrap

Original Dataset

	X1	X2	X3	X4	X5	Y
1	0.88	0.94	0.32	0.30	0.27	0.90
2	0.81	0.34	0.36	0.26	0.37	0.86
3	0.86	0.89	0.30	0.28	0.41	0.89
4	0.79	0.14	0.48	0.22	0.10	0.81
5	0.82	0.52	0.26	0.17	0.27	0.85
6	0.75	0.94	0.70	0.52	0.61	0.82
7	0.68	0.95	0.37	0.65	0.44	0.76
8	0.69	0.95	0.55	0.73	0.63	0.79
9	0.73	0.97	0.54	0.73	0.68	0.78



Bootstrap Dataset1

	X1	X2	X3	X4	X5	Y
2	0.81	0.34	0.36	0.26	0.37	0.86
5	0.82	0.52	0.26	0.17	0.27	0.85
8	0.69	0.95	0.55	0.73	0.63	0.79
5	0.82	0.52	0.26	0.17	0.27	0.85
5	0.82	0.52	0.26	0.17	0.27	0.85
4	0.79	0.14	0.48	0.22	0.10	0.81
1	0.88	0.94	0.32	0.30	0.27	0.90
4	0.79	0.14	0.48	0.22	0.10	0.81
3	0.86	0.89	0.30	0.28	0.41	0.89

Bootstrap Dataset2

	X1	X2	X3	X4	X5	Y
5	0.82	0.52	0.26	0.17	0.27	0.85
6	0.75	0.94	0.70	0.52	0.61	0.82
3	0.86	0.89	0.30	0.28	0.41	0.89
3	0.86	0.89	0.30	0.28	0.41	0.89
9	0.73	0.97	0.54	0.73	0.68	0.78
1	0.88	0.94	0.32	0.30	0.27	0.90
4	0.79	0.14	0.48	0.22	0.10	0.81
7	0.68	0.95	0.37	0.65	0.44	0.76
1	0.88	0.94	0.32	0.30	0.27	0.90

From the total N data,
choosing N data uniformly at random with replacement.

Random subspace

Original Dataset

	X1	X2	X3	X4	X5	Y
1	0.88	0.94	0.32	0.30	0.27	0.90
2	0.81	0.34	0.36	0.26	0.37	0.86
3	0.86	0.89	0.30	0.28	0.41	0.89
4	0.79	0.14	0.48	0.22	0.10	0.81
5	0.82	0.52	0.26	0.17	0.27	0.85
6	0.75	0.94	0.70	0.52	0.61	0.82
7	0.68	0.95	0.37	0.65	0.44	0.76
8	0.69	0.95	0.55	0.73	0.63	0.79
9	0.73	0.97	0.54	0.73	0.68	0.78

Random subspace Dataset1

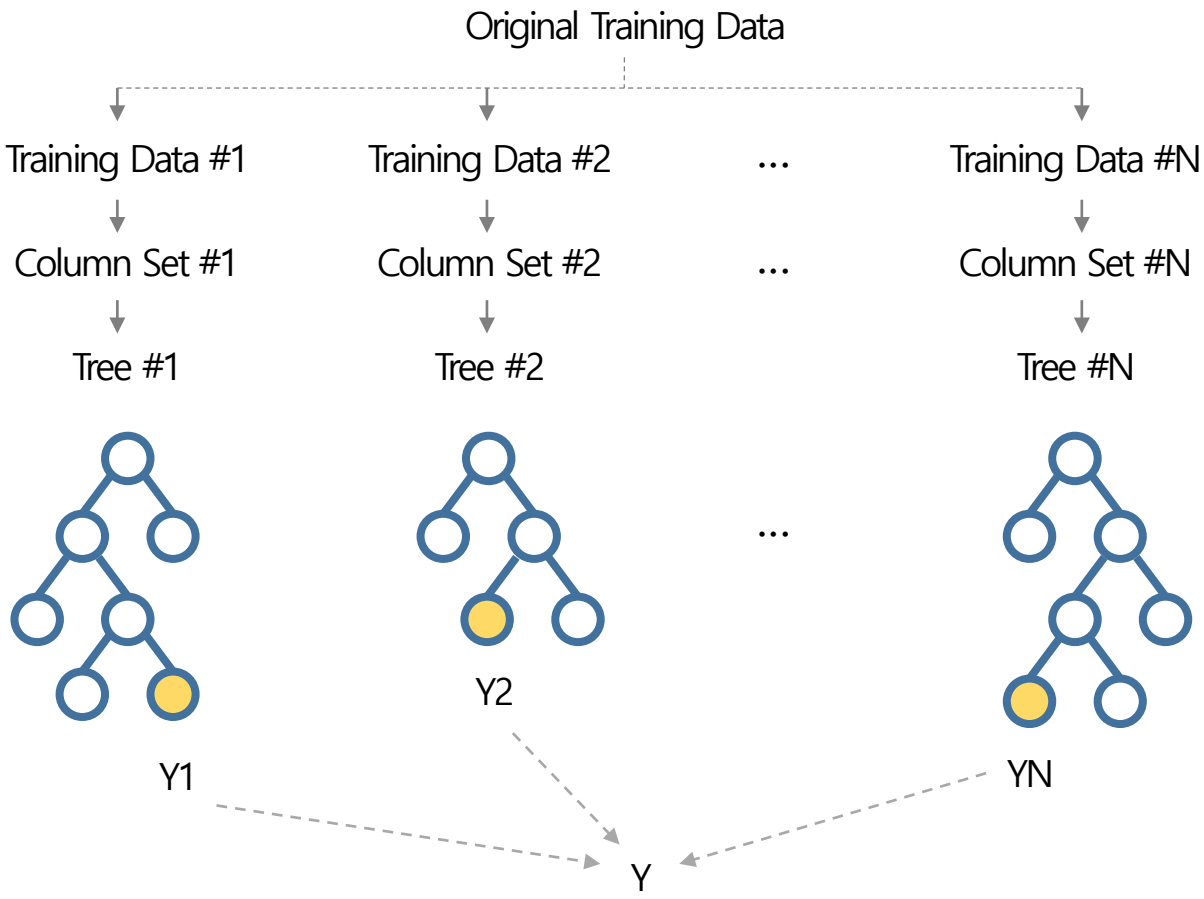
	X1	X2	X3	X4	X5	Y
1	0.88	0.94	0.32	0.30	0.27	0.90
2	0.81	0.34	0.36	0.26	0.37	0.86
3	0.86	0.89	0.30	0.28	0.41	0.89
4	0.79	0.14	0.48	0.22	0.10	0.81
5	0.82	0.52	0.26	0.17	0.27	0.85
6	0.75	0.94	0.70	0.52	0.61	0.82
7	0.68	0.95	0.37	0.65	0.44	0.76
8	0.69	0.95	0.55	0.73	0.63	0.79
9	0.73	0.97	0.54	0.73	0.68	0.78

Random subspace Dataset2

	X1	X2	X3	X4	X5	Y
1	0.88	0.94	0.32	0.30	0.27	0.90
2	0.81	0.34	0.36	0.26	0.37	0.86
3	0.86	0.89	0.30	0.28	0.41	0.89
4	0.79	0.14	0.48	0.22	0.10	0.81
5	0.82	0.52	0.26	0.17	0.27	0.85
6	0.75	0.94	0.70	0.52	0.61	0.82
7	0.68	0.95	0.37	0.65	0.44	0.76
8	0.69	0.95	0.55	0.73	0.63	0.79
9	0.73	0.97	0.54	0.73	0.68	0.78

From the total p variables,
choosing m variables uniformly at random.

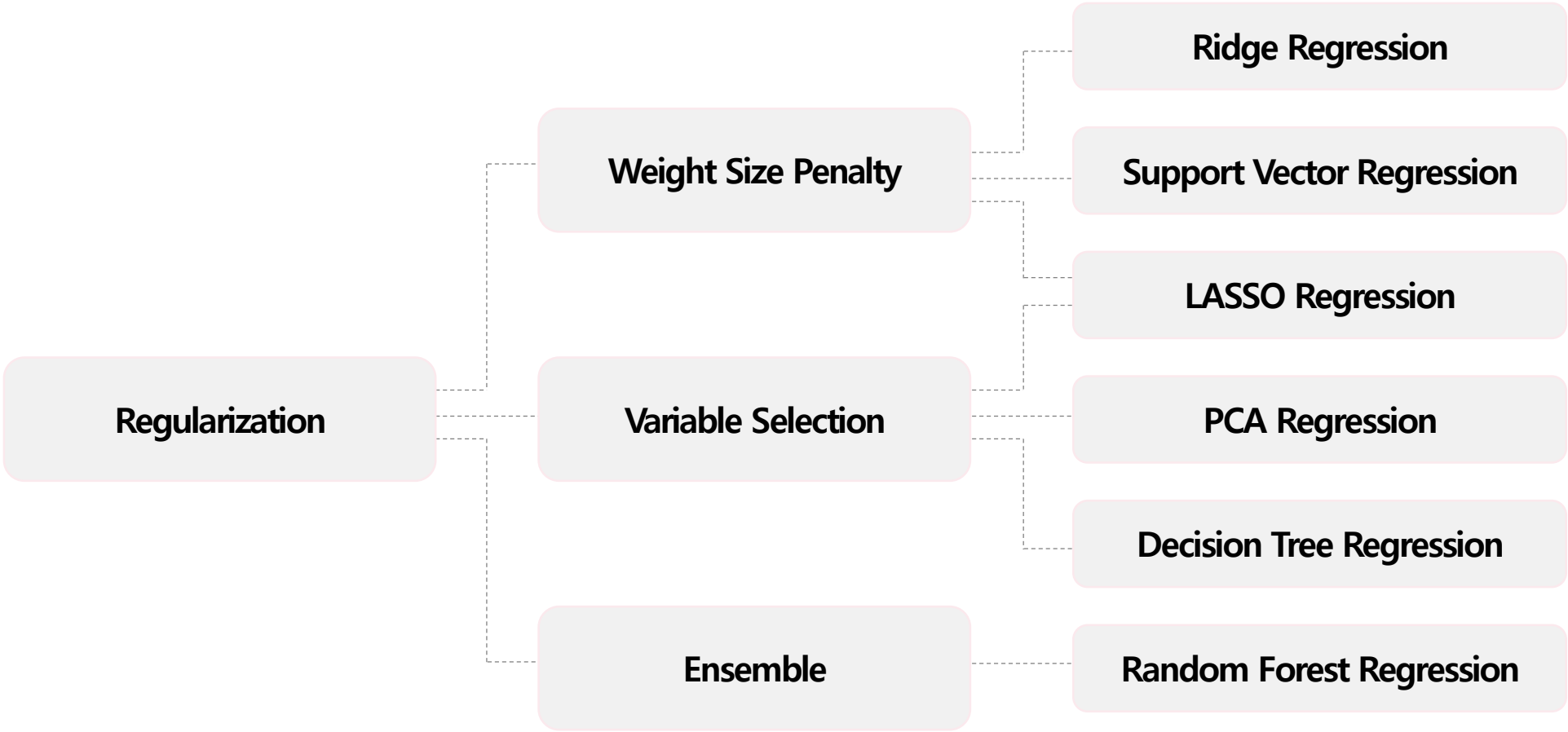
Random forest Regression



Bootstrap

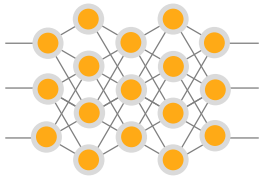
Random Subspace

Ensemble


Regularization
Weight Size Penalty
Variable Selection
Ensemble
Ridge Regression
Support Vector Regression
LASSO Regression
PCA Regression
Decision Tree Regression
Random Forest Regression

Dropout

A Simple Way to Prevent Neural Networks from Overfitting



[PDF] **Dropout**: a simple way to prevent neural networks from overfitting

N Srivastava, G Hinton, A Krizhevsky... - The journal of machine ..., 2014 - jmlr.org

Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of ...

☆ 99 Cited by 14569 Related articles All 25 versions Web of Science: 5045 »

[PDF] **Improving neural networks** with **dropout**

N Srivastava - University of Toronto, 2013 - cs.toronto.edu

Neural networks are powerful computational models that are being used extensively for solving problems in vision, speech, natural language processing and many other areas. In spite of many successes, neural networks still suffer from a major weakness. The presence

☆ 99 Cited by 168 Related articles All 7 versions »

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Nitish Srivastava NITISH@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
Alex Krizhevsky KRIZ@CS.TORONTO.EDU
Ilya Sutskever ILYA@CS.TORONTO.EDU
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
10 Kings College Road, Rm 3302
Toronto, Ontario, M5S 3G4, Canada.

Editor: Yoshua Bengio

Abstract

Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods. We show that dropout improves the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark data sets.

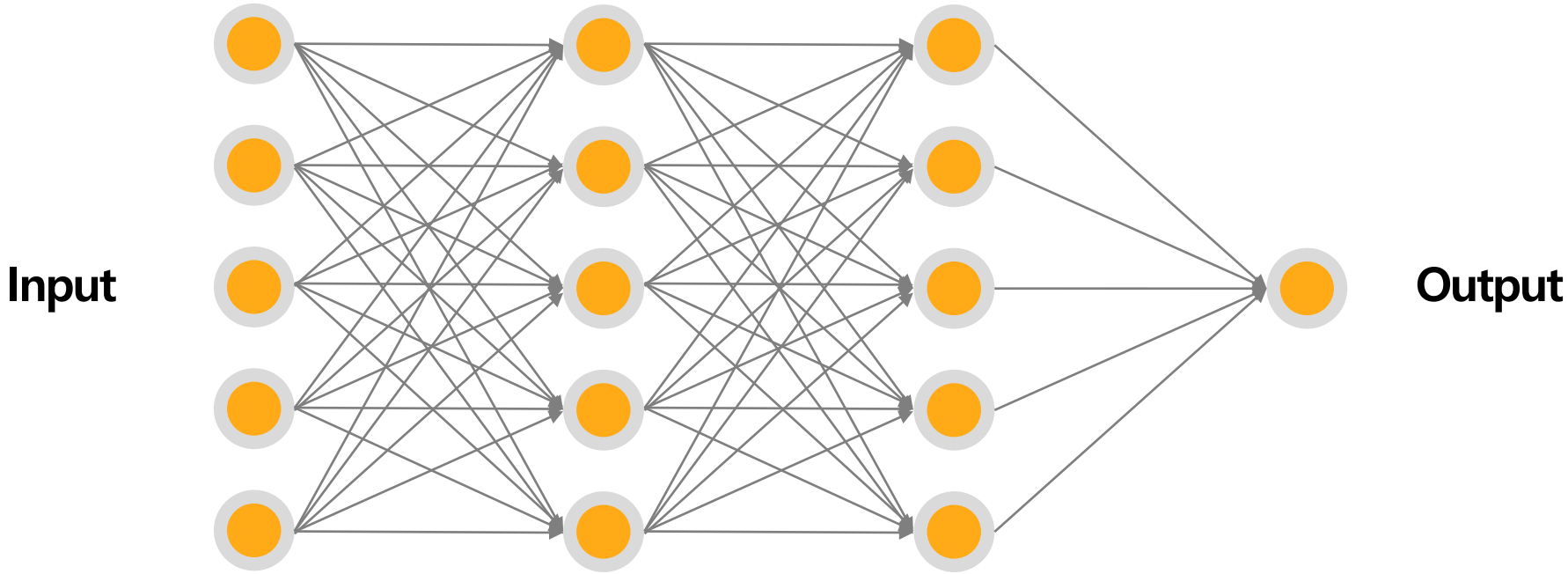
Keywords: neural networks, regularization, model combination, deep learning

Dropout : A Simple Way to Prevent Neural Networks from Overfitting

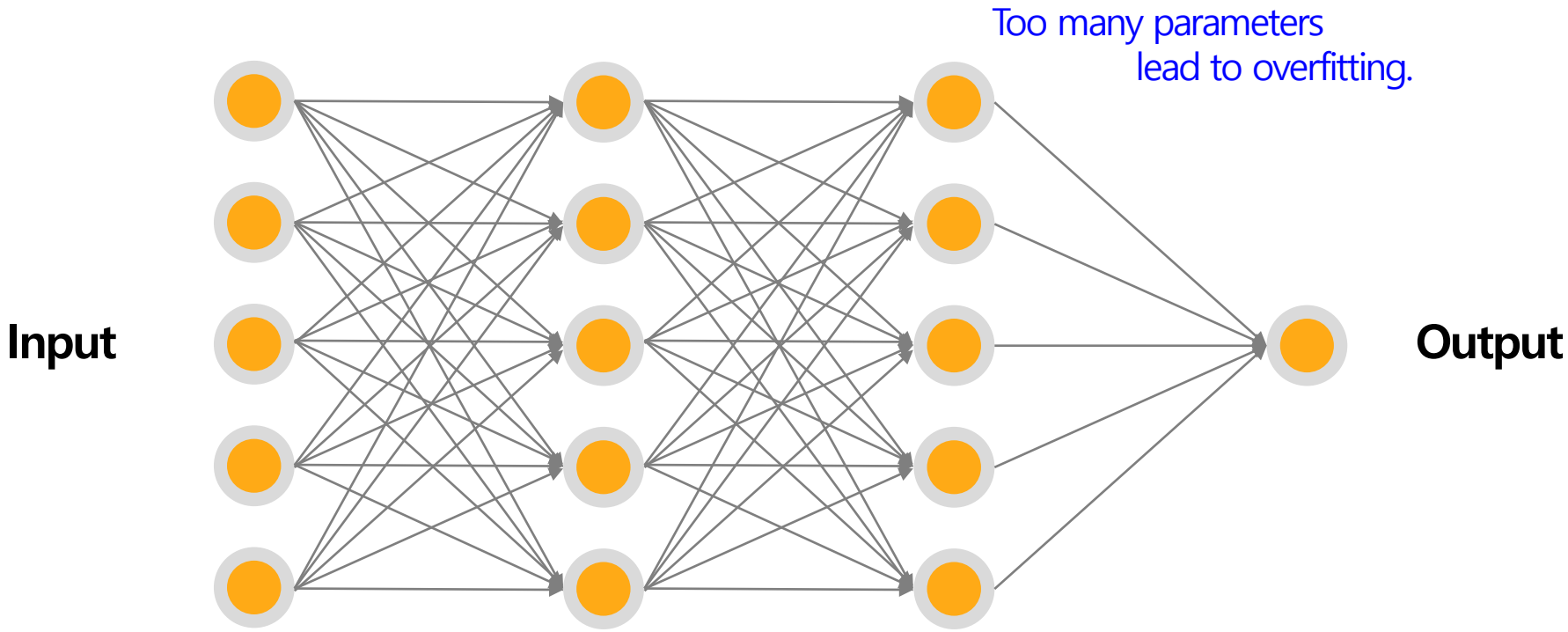
Abstract

Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single "unthinned" network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods. We show that dropout improves the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark data sets.

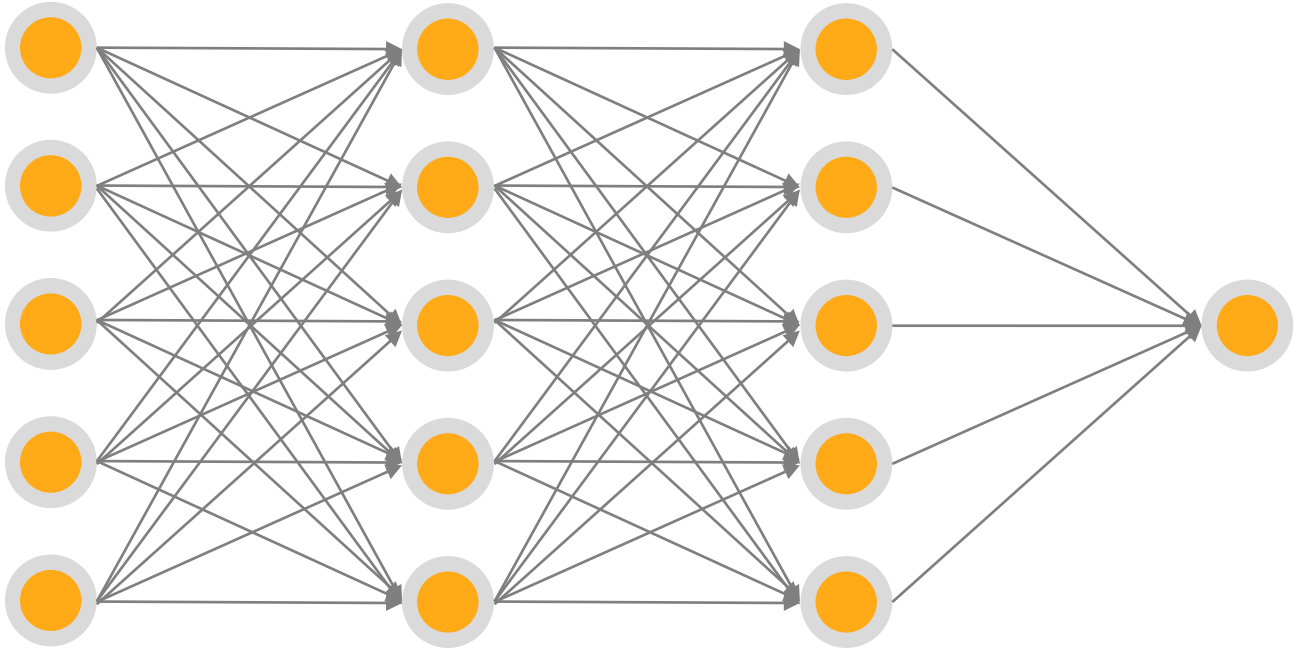
Neural Network

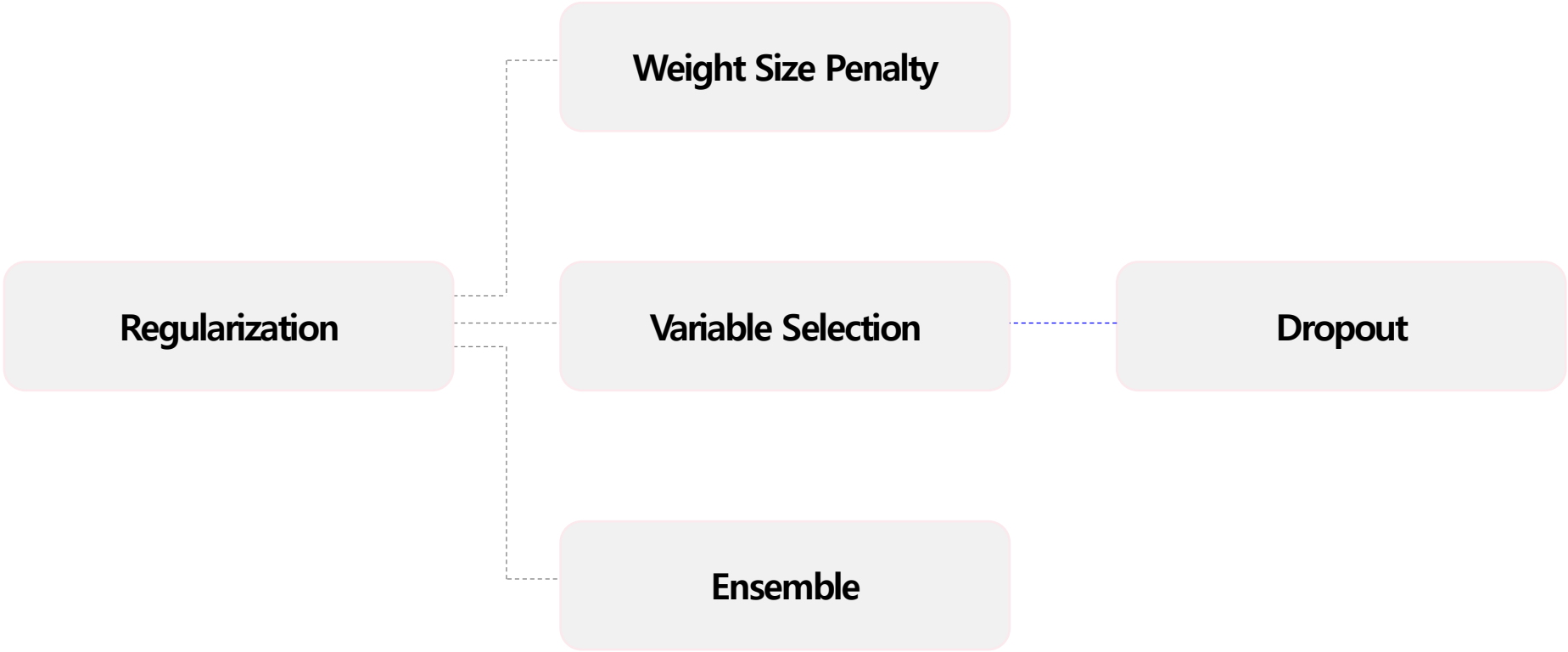


Neural Network

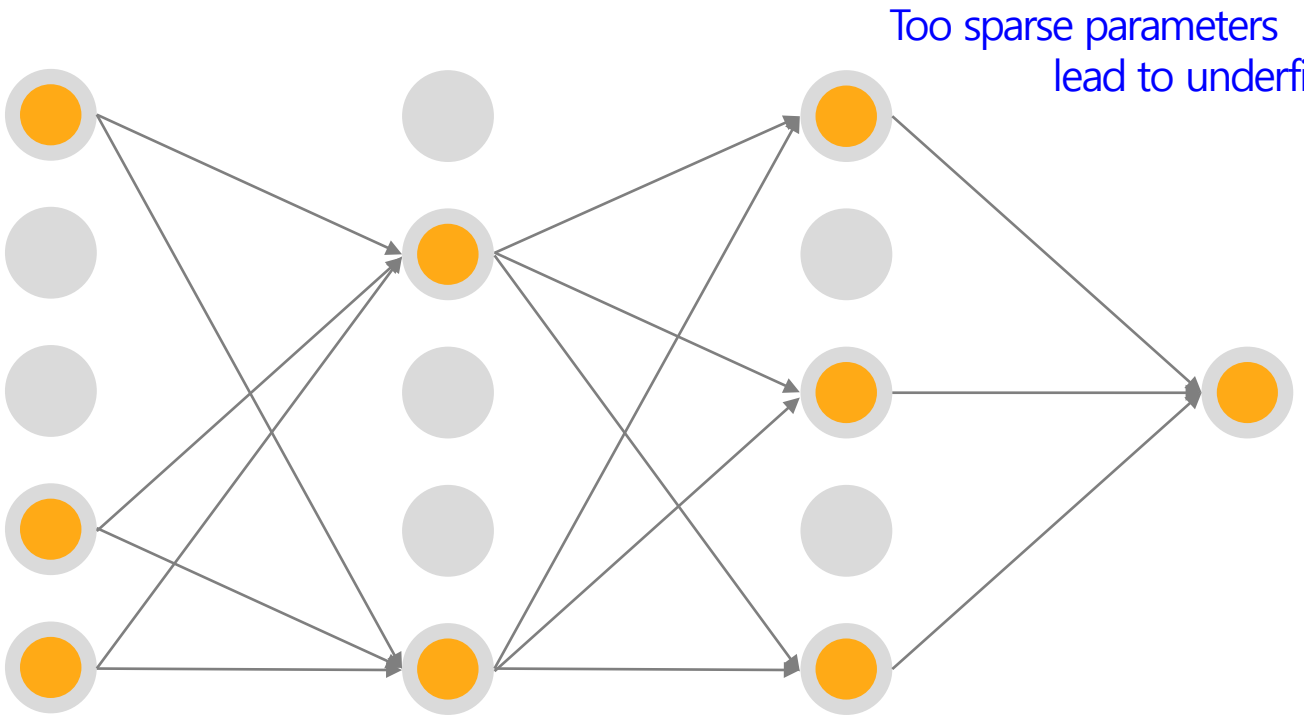


Dropout = Variable selection

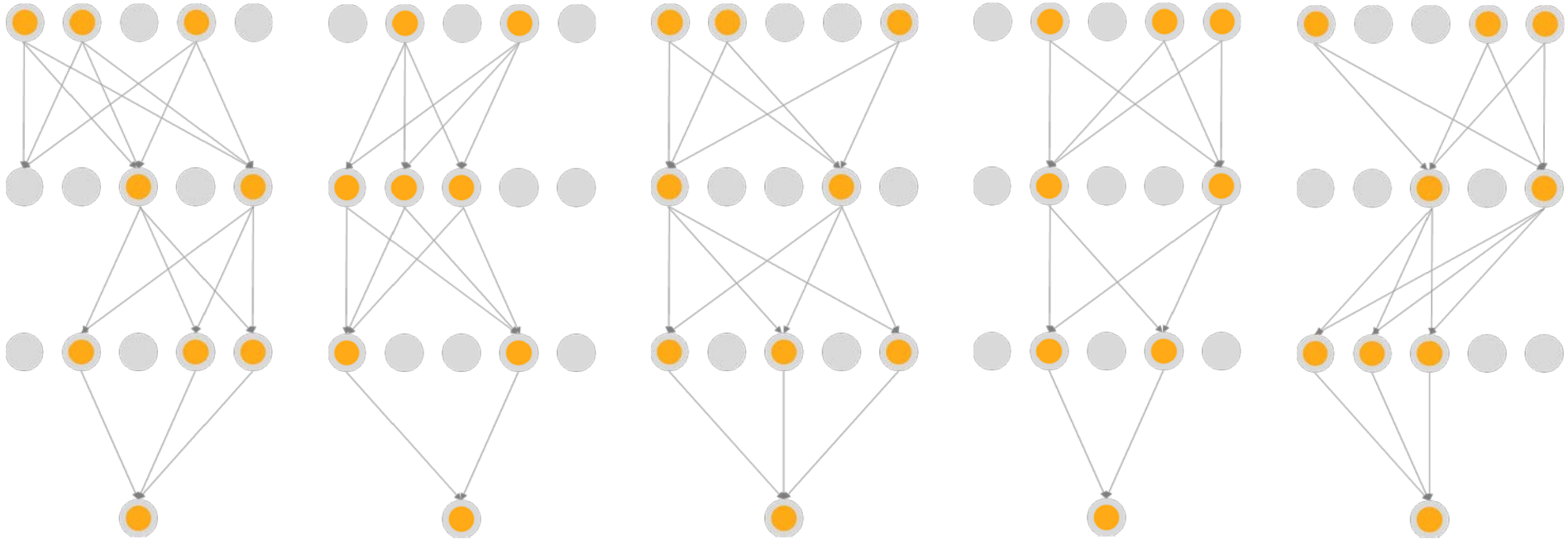


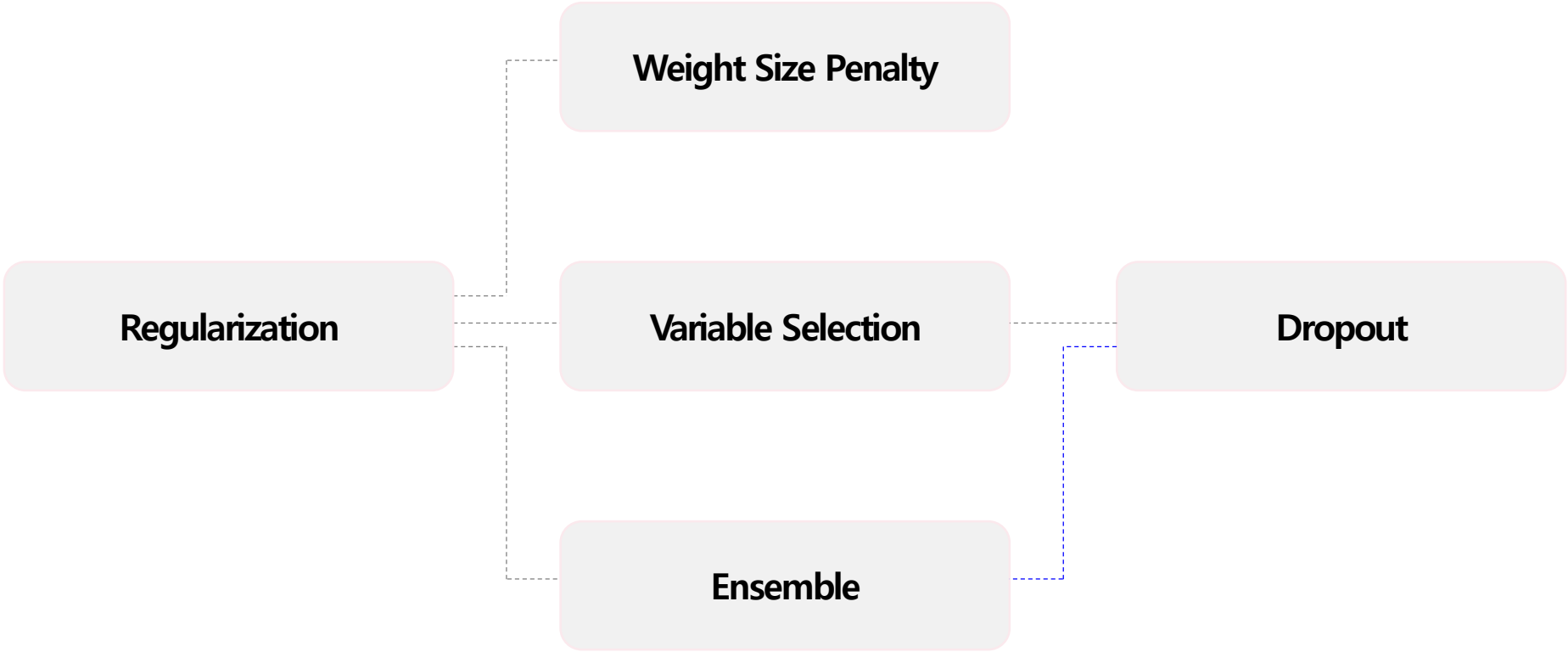

Weight Size Penalty
Regularization
Variable Selection
Dropout
Ensemble

Dropout = Variable selection

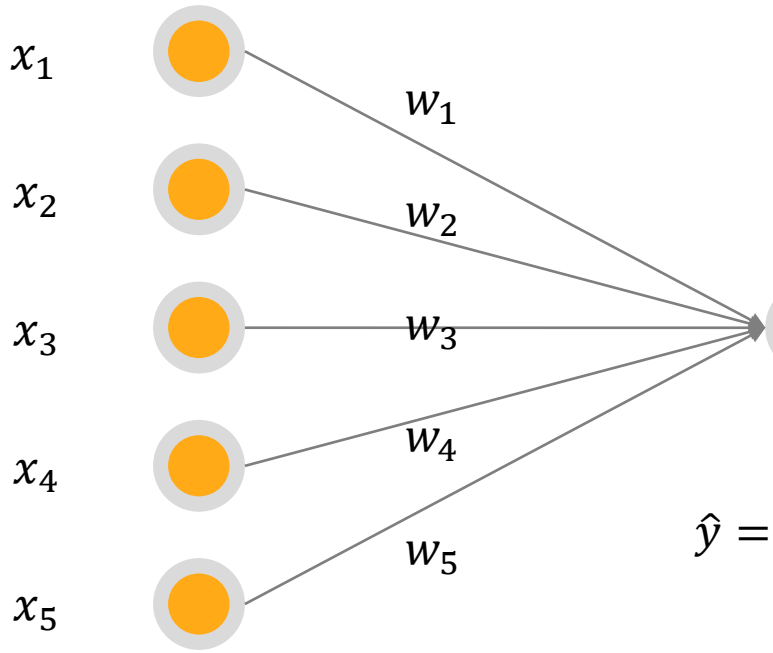


Dropout = Ensemble



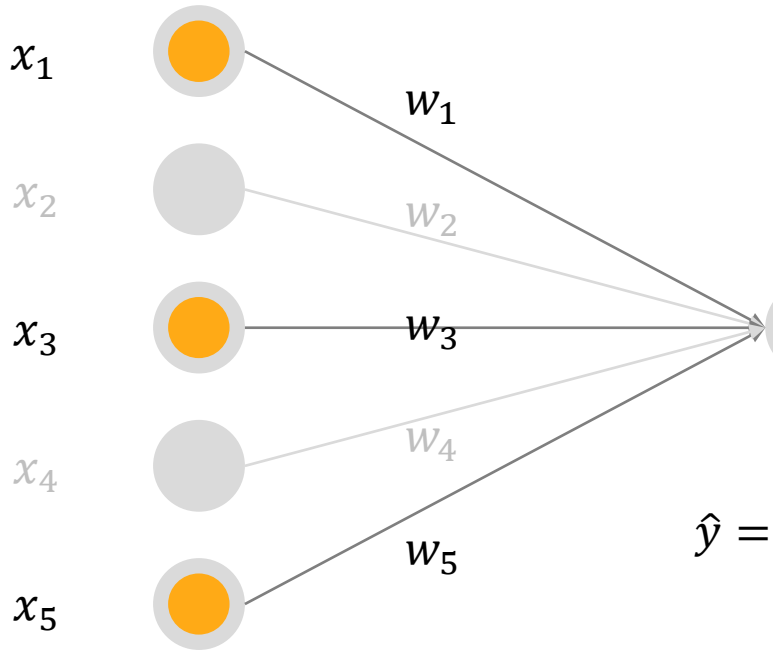

Weight Size Penalty
Regularization
Variable Selection
Dropout
Ensemble

Dropout at training time



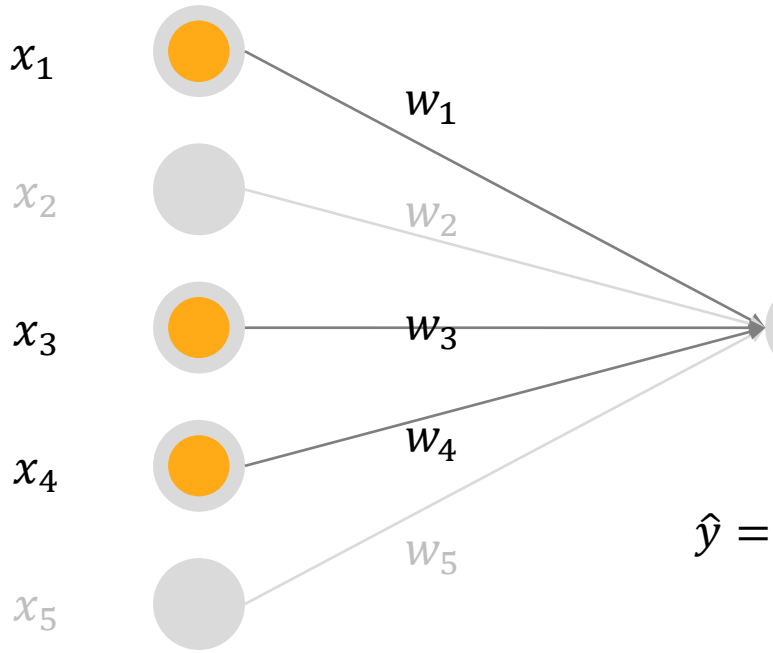
$$\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5$$

Dropout at training time



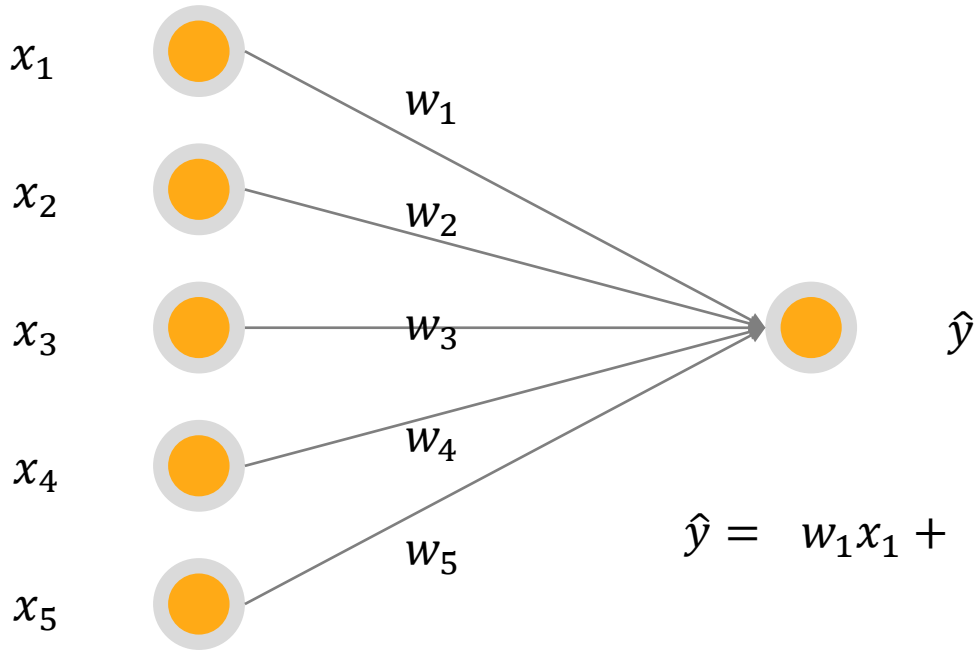
$$\hat{y} = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5$$

Dropout at training time



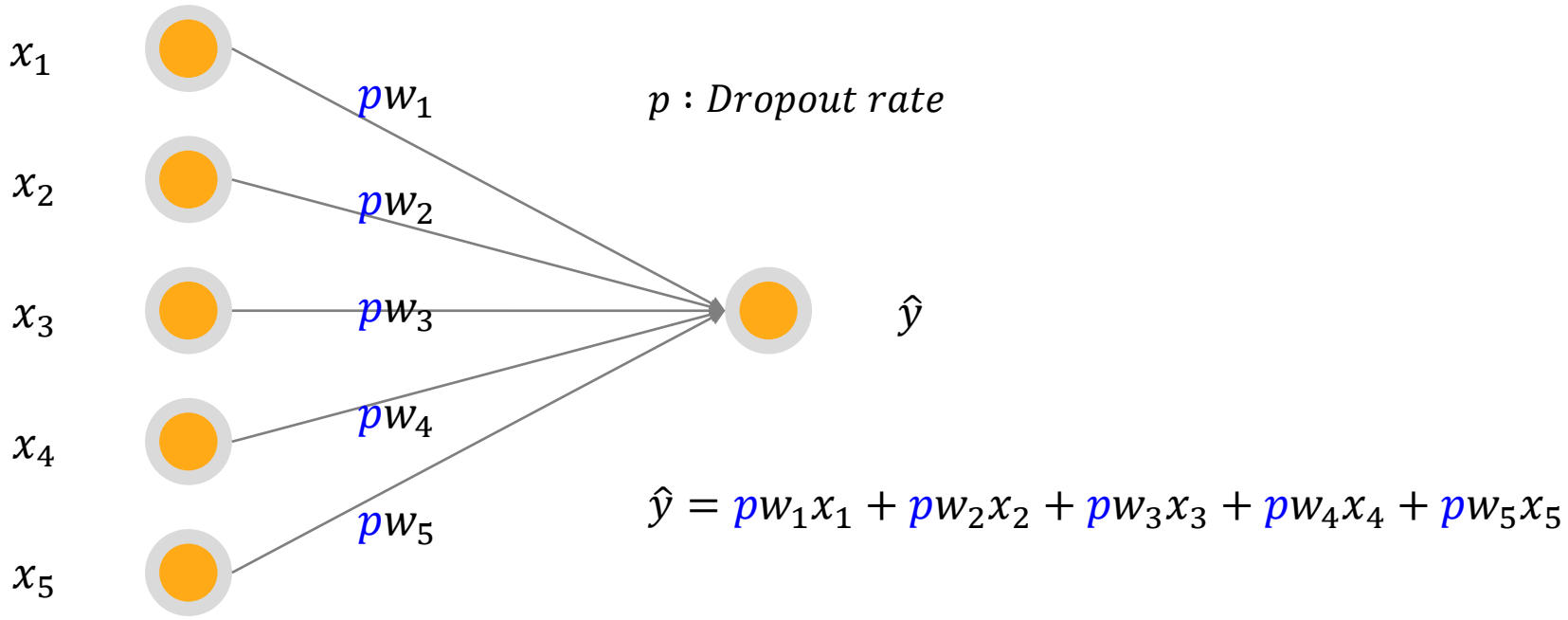
$$\hat{y} = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5$$

Dropout at test time

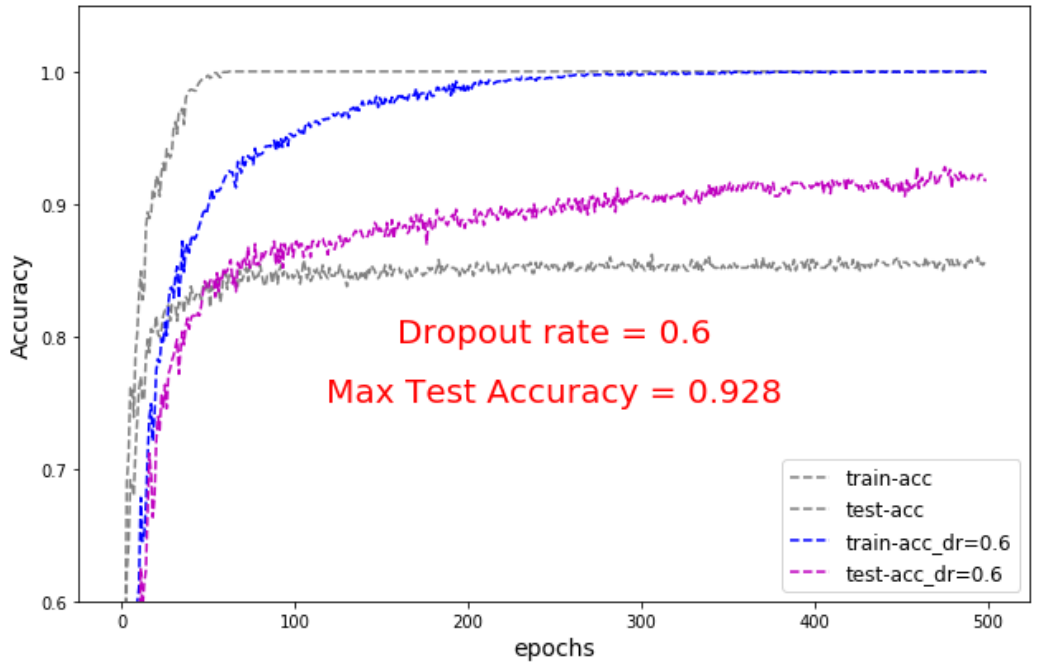


$$\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5$$

Dropout at test time



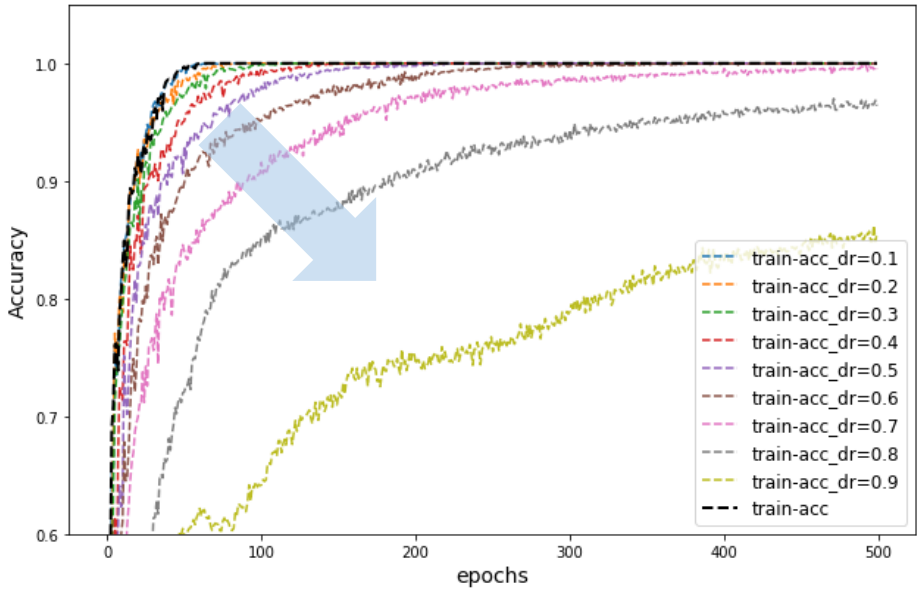
Dropout Evaluation



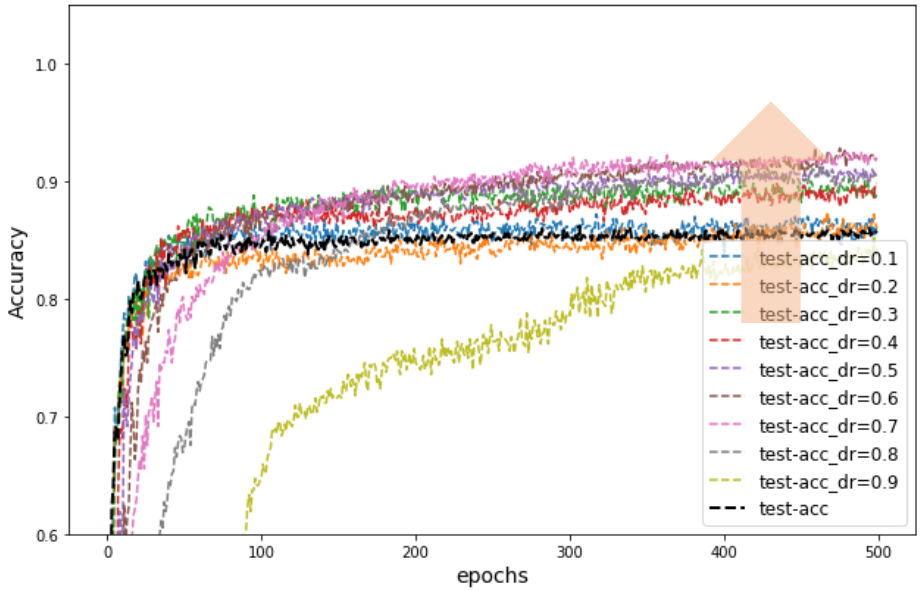
Dropout rate	Test Accuracy
0.0	0.862
0.1	0.872
0.2	0.874
0.3	0.902
0.4	0.898
0.5	0.912
0.6	0.928
0.7	0.924
0.8	0.912
0.9	0.852

Dropout Evaluation

Training Accuracy



Test Accuracy



Place of dropout layer

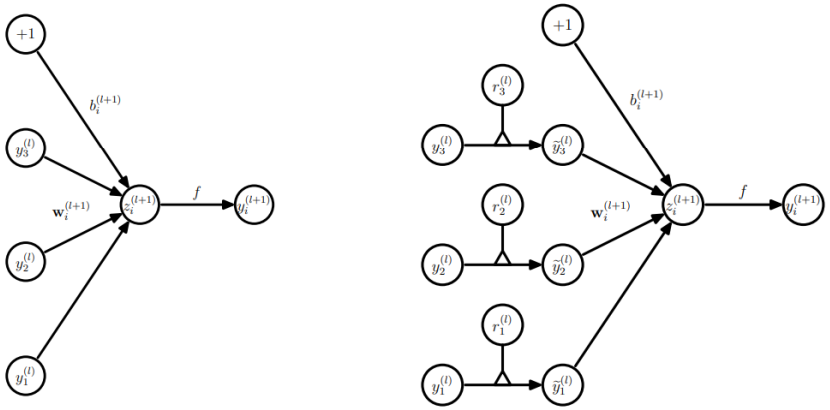
Before the activation layer

```
x = Conv2D(channel, kernal_size, strides, padding)(x)
x = Dropout(0.5)(x)
x = LeakyReLU()(x)

x = Conv2D(channel, kernal_size, strides, padding)(x)
```

After the activation layer

```
x = Conv2D(channel, kernal_size, strides, padding)(x)
x = LeakyReLU()(x)
x = Dropout(0.5)(x)

x = Conv2D(channel, kernal_size, strides, padding)(x)
```

Dropout Description 1



(a) Standard network (b) Dropout network

Figure 3: Comparison of the basic operations of a standard and dropout network.

Consider a neural network with L hidden layers.

Let $l \in \{1, \ldots, L\}$ *index the hidden layers of network.*

Let $z^{(l)}$ *the vector of inputs to layer* l.

Let $y^{(l)}$ *the vector of outputs to layer* l. $(y^{(0)} = x)$

$W^{(l)}, b^{(l)}$: *weight and biases at layer* l

Standard neural network

$$z_i^{(l+1)} = w_i^{(l+1)} y^{(l)} + b_i^{(l+1)}$$

$$y_i^{(l+1)} = f\left(z_i^{(l+1)}\right)$$

Dropout neural network

$$r_j^{(l)} \sim Bernoulli(p)$$

$$\tilde{y}^{(l)} = r^{(l)} * y^{(l)}$$

$$z_i^{(l+1)} = w_i^{(l+1)} \tilde{y}^{(l)} + b_i^{(l+1)}$$

$$y_i^{(l+1)} = f\left(z_i^{(l+1)}\right)$$

Dropout Description 2

In linear regression, $\min_w \|y - Xw\|^2$

+ dropout $\min_w \mathrm{E}_{R\sim Bernoulli(p)}[\|y - (R * X)w\|^2]$ $R_{ij}\sim Bernoulli(p)$

This reduces to $\min_w \|y - pXw\|^2 + p(1-p)\|\Gamma w\|^2$ $\Gamma = \left(diag(X^T X)\right)^{\frac{1}{2}}$

Another way $\min_w \|y - X\tilde{w}\|^2 + \frac{1-p}{p}\|\Gamma\tilde{w}\|^2$ $\tilde{w} = pw$

Dropout Description 2

Another way

$$\min_{w} \|y - X\tilde{w}\|^2 + \frac{1-p}{p}\|\Gamma\tilde{w}\|^2$$

$$0 < p \leq 1, \qquad \infty > \frac{1-p}{p} \geq 0$$

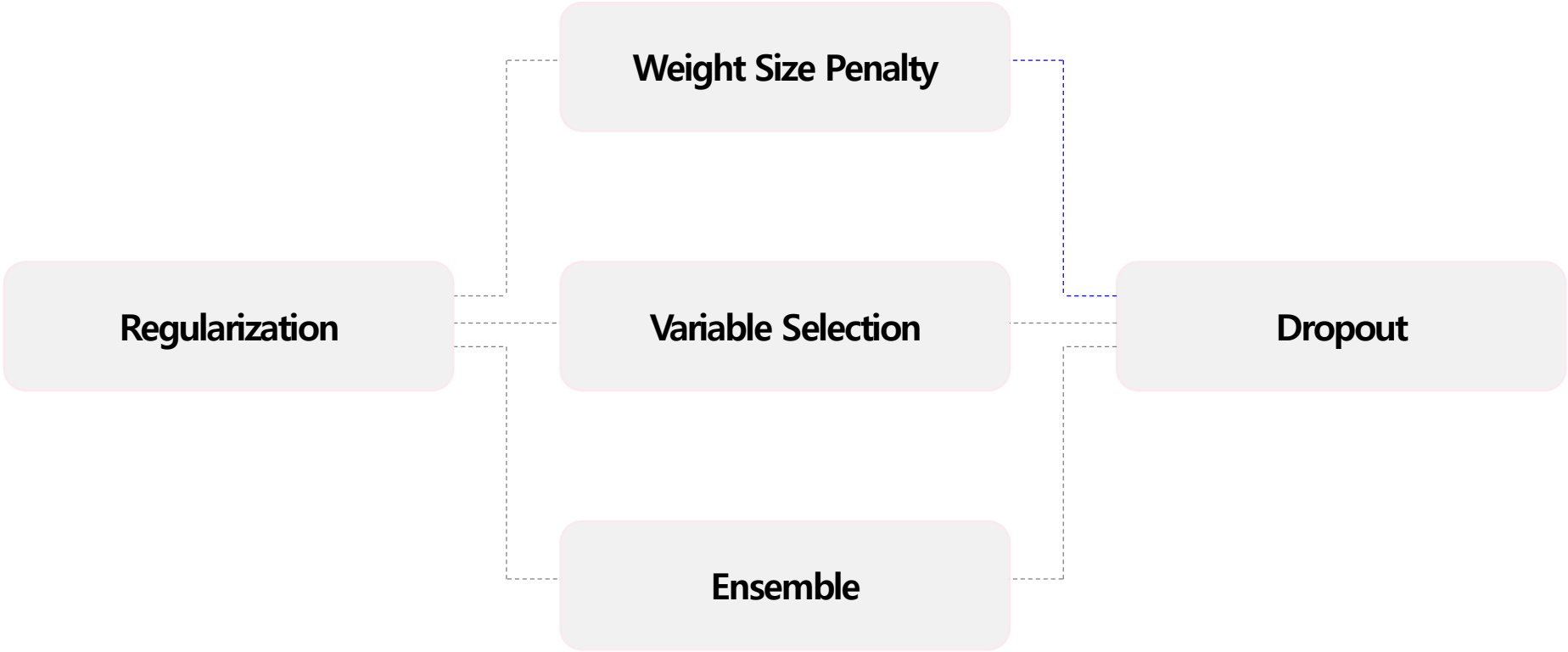
p: *the probability of retaining a unit in the network*

Ridge regression

$$\min_{\beta} \sum_{i=1} (y_i - \hat{y}_i)^2 + \lambda\|\boldsymbol{\beta}\|^2$$

$\lambda \uparrow \sim \boldsymbol{\beta} \downarrow$ $\quad p \uparrow \sim \boldsymbol{w} \uparrow$

$\lambda \downarrow \sim \boldsymbol{\beta} \uparrow$ $\quad p \downarrow \sim \boldsymbol{w} \uparrow$


Weight Size Penalty
Regularization
Variable Selection
Dropout
Ensemble

감사합니다.