Day 1: Machine learning

Sutharshan Rajasegarar
School of Information Technology
Deakin University

Outline

- Introduction to machine learning
- Mode evaluation
- Supervised learning
  - Decision tree
  - Random forest
  - Support vector machine
- Unsupervised learning
  - One class support vector machine for anomaly detection
  - Local outlier factor based anomaly detection

Successful ML: Board Games

AlphaGo, the board-game-playing AI from Google's DeepMind beats Korean Go Champion Lee Sedol 4-1 in March 2016.

AlphaGo uses Deep Neural Networks and Monte Carlo Tree Search.

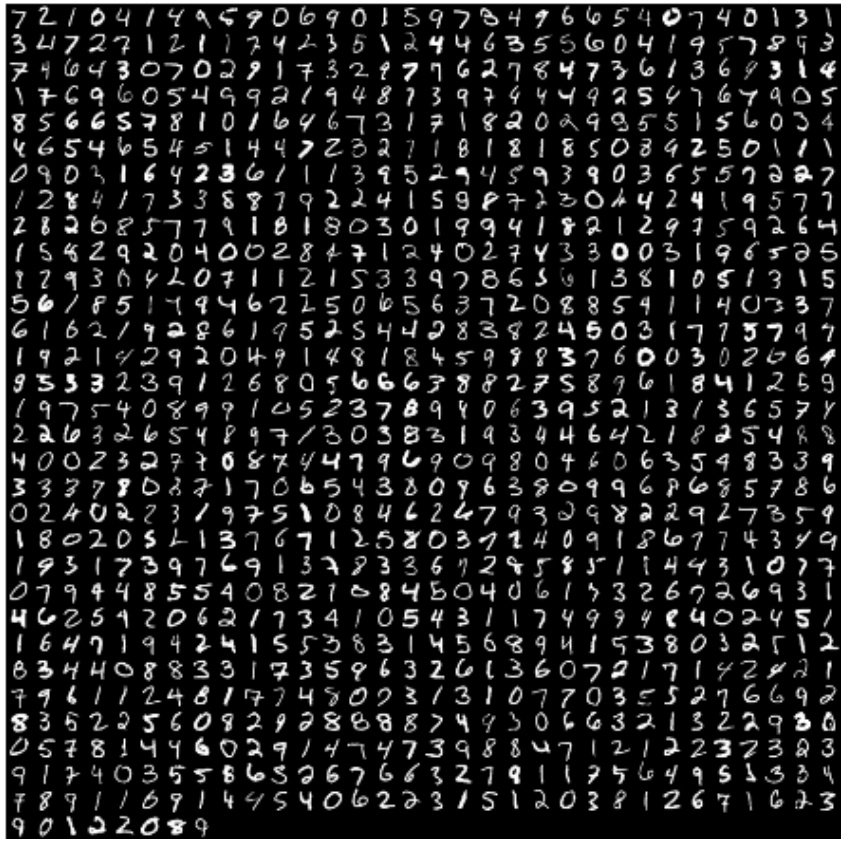
Successful ML: Voice Recognition

Technologies such as Siri, Google Now

Siri uses Speech recognizer, Natural Language Processing and Text-to-speech techniques.

Successful ML: Digit Recognition

Handwritten Digit Recognition



Machine Learning methods (SVM and Deep Learning) have hit >99% accuracy for this task.

Successful ML: Other applications

- **Healthcare Analytics: Diagnosis and Prognosis**
- **Stock Market Prediction**
- **Business Analytics**
- **Face Recognition**
- **And many others …**

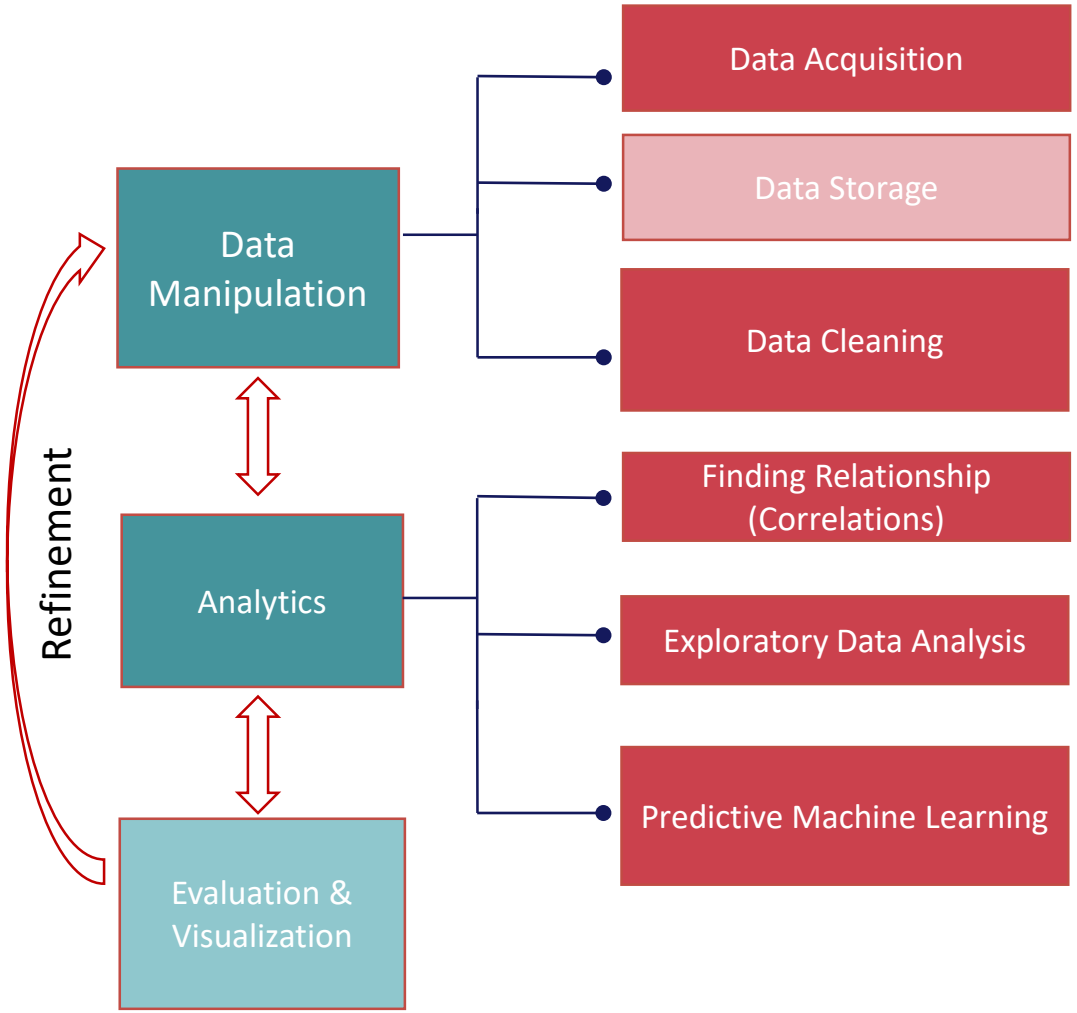
What is Machine Learning?

"Field of study that gives computers the ability to learn without being explicitly programmed", Arthur Samuel

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E", Tom Mitchell

Tom Mitchell, Professor at CMU

Steps Involved in Machine Learning

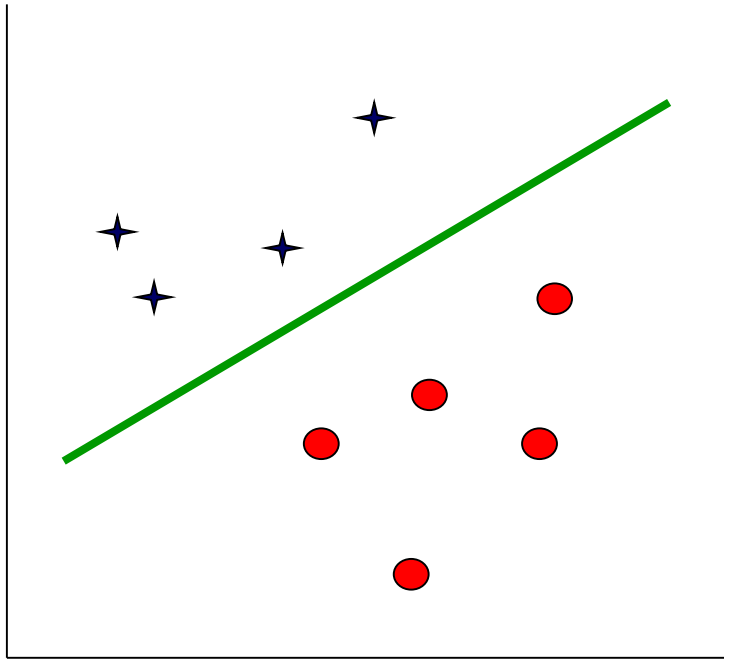


Types of Machine Learning

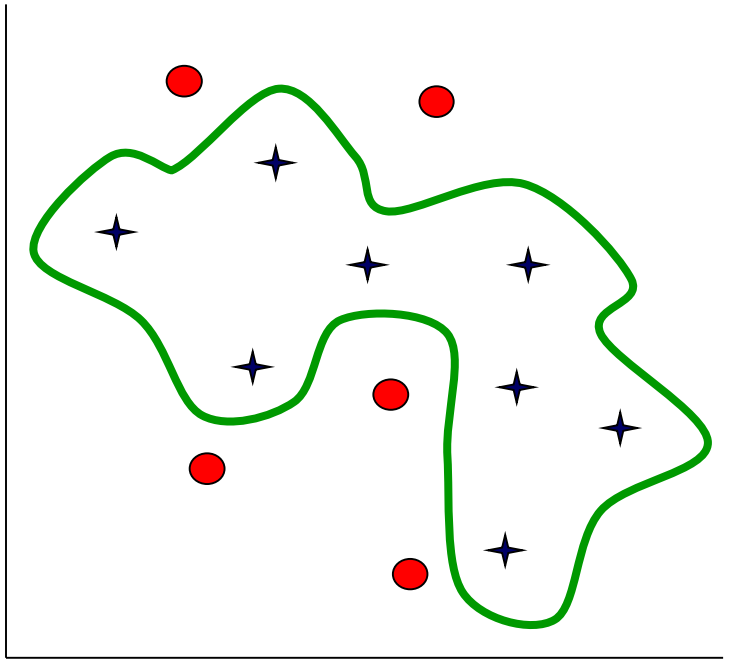
- **Supervised learning**
- **Unsupervised learning**

Classification

Consider classification problem in 2 dimensions



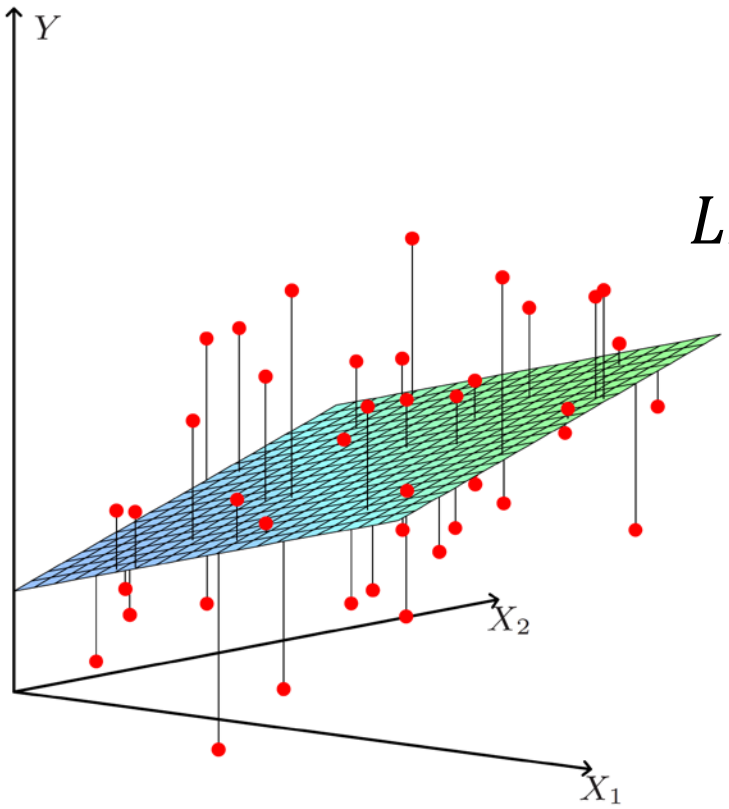
Linear decision boundary



Nonlinear decision boundary

Regression

Consider regression problem in 2 dimensions



Linear hyperplane

Unsupervised Learning

Learn patterns from (unlabeled) data $(x_1, \ldots, x_n)$

Popular Approaches

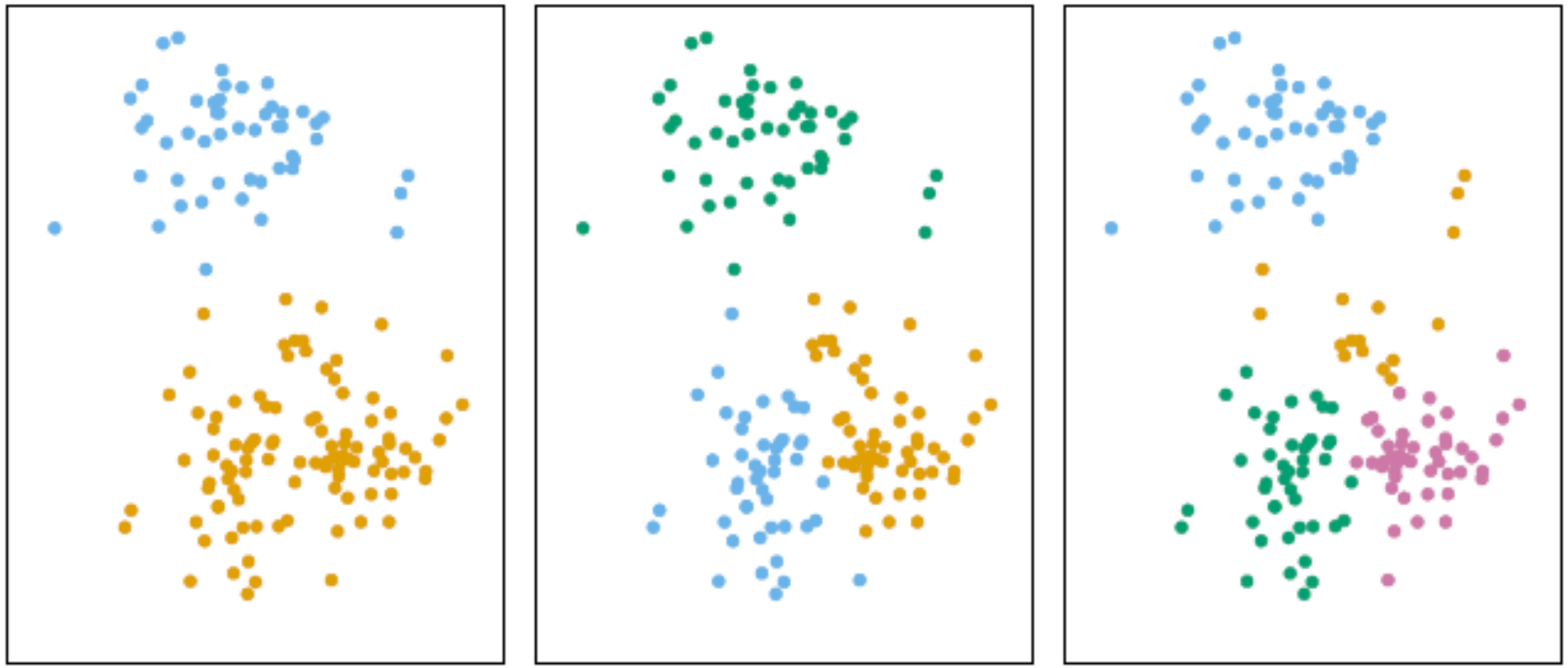
- clustering (similarity-based), density estimation

Tasks

- Data understanding and visualization
- anomaly detection
- information retrieval
- data compression (reduction)

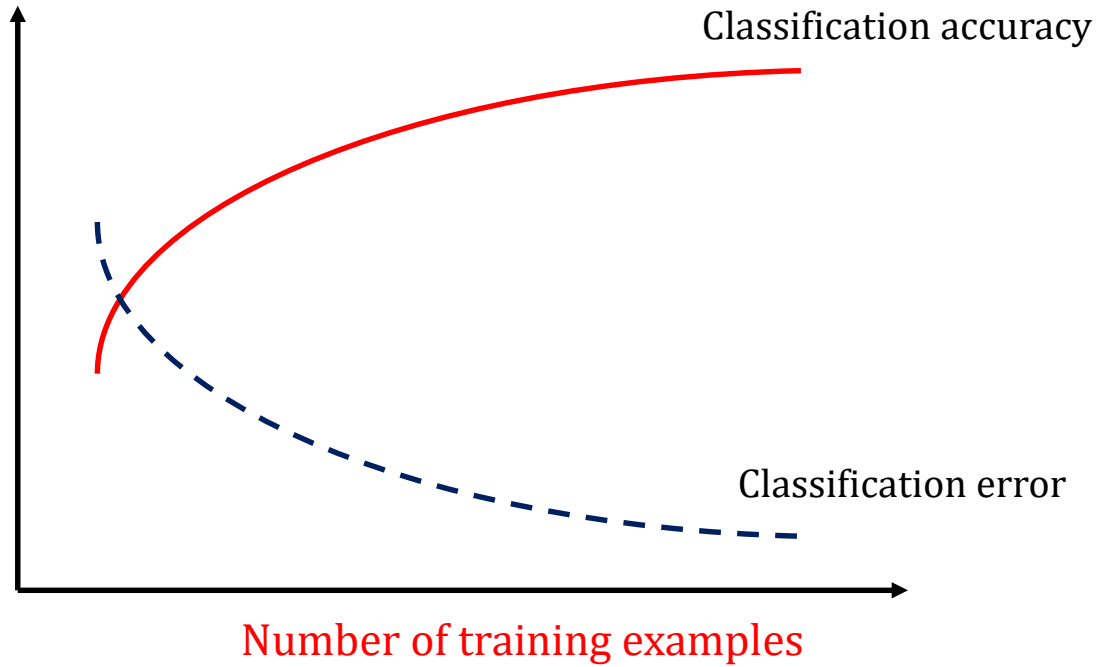
Unsupervised Learning (Clustering)

- No label is given
- Find structure and meaning in the data using similarity and correlations



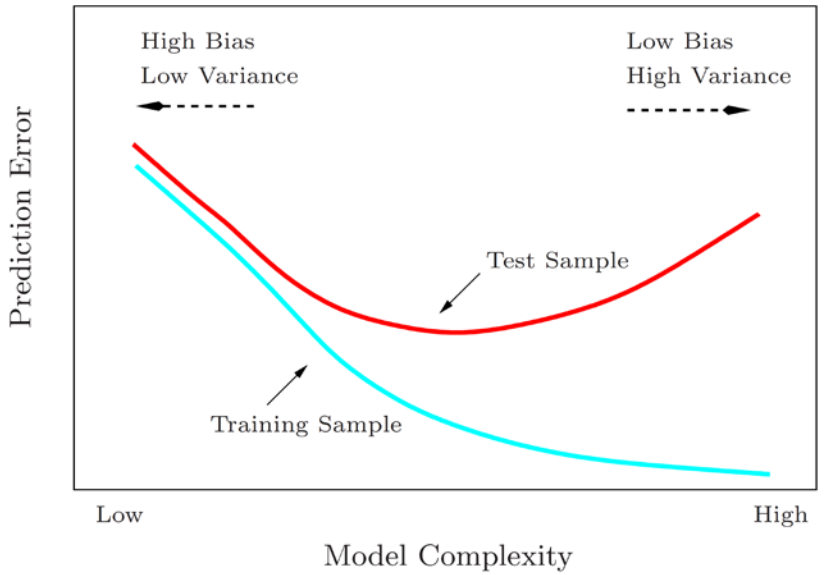
Model Evaluation

- Randomly split examples into *training set* and *test set.*
- Use training set to learn a model.
- Evaluation: Measure % of correctly classified examples *using test set.*
- Repeat for different random splits and average results.



Model Assessment and Selection

- In our model (hypothesis), often there are many knobs (parameters and hyper-parameters) that we can use to vary its fitness to the data.
- There is no easy way to know if certain fitness is the best.



[Source: ESL book, Friedman, Hastie and Tibshirani (2001)]

- There are many effective ways in which people approach this problem:
  - Look at averaged evaluation score on many random test sets.
  - Cross-validation (train using one set and test on the other, rotate them), AIC, BIC etc.

Next section…

- ✓ Linear regression
- ✓ Under-fitting
- ✓ Over-fitting
- ✓ Bias of the model
- ✓ Variance of the model
- ✓ Bias-Variance Trade-off
- ✓ Getting the right fit with Regularization
- ✓ L_2 norm Regularization
- ✓ L_1 norm Regularization
- ✓ L_2/L_1 regularization
- ✓ Most of the above concepts apply to all supervised learning models (except where mentioned otherwise)

Linear Regression

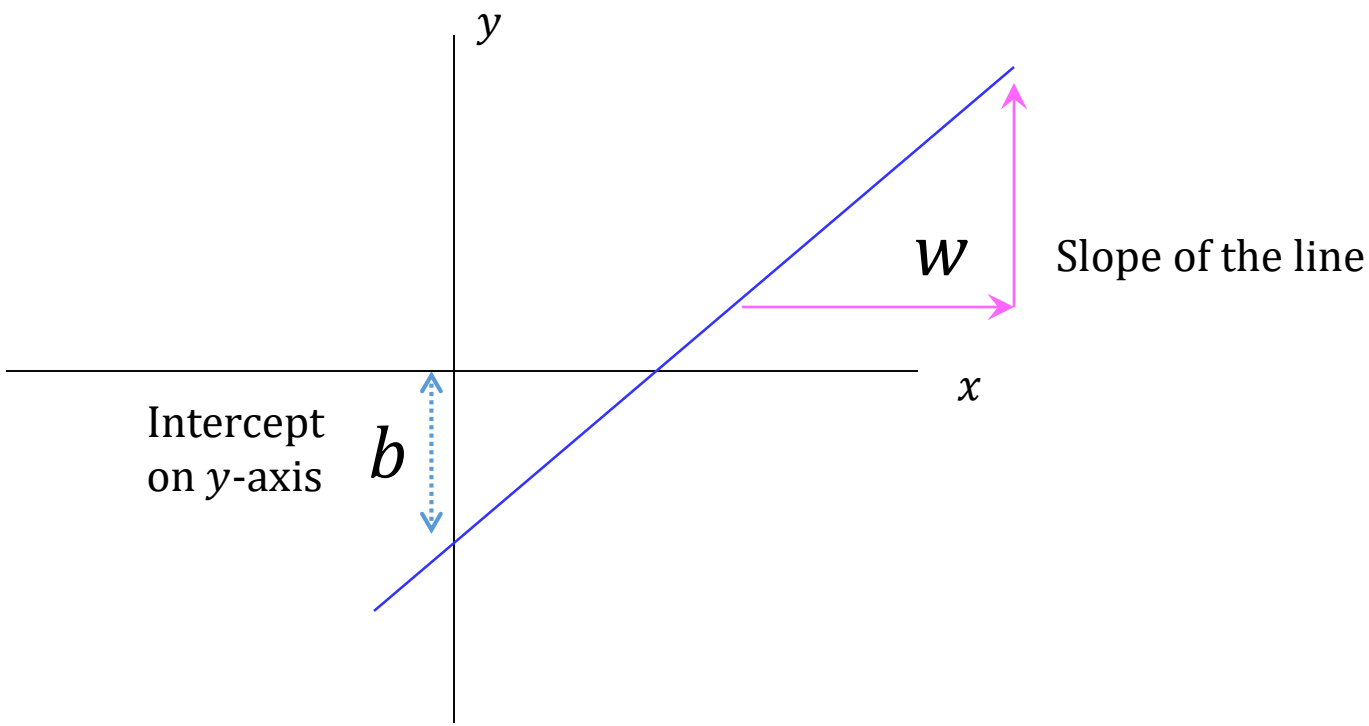
- Training data is of the form: $\{x_i, y_i\}, i = 1, \ldots, n$.
- For each data point x_i, there is an output y_i, which can be any real-valued number.
- We want to explain data using a line.

What is “Linear” hypothesis?

- Linear hypothesis:

$$y = h(x) = wx + b$$

- For data point x_i, we have $\hat{y}_i = wx_i + b$



- Line predicts $\hat{y}_i$ for x_i.

Linear Regression Formulation

- In the previous slide, we talked about linear regression in 1-dimension.
- In d-dimension, we can write linear regression as

$$\hat{y}_i = b + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id}$$

- We can rewrite the above by introducing an extra feature x_{i0}=1 as

  $\hat{y}_i = w_0 x_{i0} + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id}$, where $w_0 = b$ and $x_{i0} = 1$.

- Using the vector notation, we can write the above as $\hat{y}_i = x_i^T w$ using $(d + 1)$ dimensional vectors.

Linear Regression Formulation

- For $i = 1, \ldots, n$, we have

$$\hat{y}_1 = x_1^T w, \qquad \text{for } i = 1$$
$$\vdots$$
$$\hat{y}_n = x_n^T w, \qquad \text{for } i = n$$

- Collectively, we can write:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{w}$$

where $\hat{\boldsymbol{y}} = [\hat{y}_1 \quad \ldots \quad \hat{y}_n]^T$, $\boldsymbol{w} = [w_0 \quad \ldots \quad w_d]^T$.

Linear Regression Formulation

- Error in fitting when using line: $e_i = y_i - \hat{y}_i$.

- The best linear model minimizes the empirical risk $R(\boldsymbol{w})$ via square loss $(y_i - \hat{y}_i)^2$ as

$$\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

or $$\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - x_i^T \boldsymbol{w}\right)^2$$

The above formulation is same as minimizing sum-of-the square-errors.

Solution of the Optimization Problem

- After taking the derivative of the objective function and equating to zero, we find the optimum solution as

$$\boldsymbol{w} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-\mathbf{1}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$$

- Just for your knowledge, the matrix $\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-\mathbf{1}}\boldsymbol{X}^{\boldsymbol{T}}$ is known as Moore-Penrose pseudo-inverse of the matrix $\boldsymbol{X}$ and often denoted as $\boldsymbol{X}^{\dagger}$. This can be thought as generalization of the notion of matrix inverse to non-square matrices.

Example of Linear Regression

- Training data: Five randomly selected students took a math aptitude test before they began their statistics course. The Statistics Department wants to know:
  - What linear regression equation best predicts statistics performance, based on math aptitude scores?

- Answer: $X=\begin{bmatrix}1 & 95\\ 1 & 85\\ 1 & 80\\ 1 & 70\\ 1 & 60\end{bmatrix}$, $y=\begin{bmatrix}85\\ 95\\ 70\\ 65\\ 70\end{bmatrix}$, using $\boldsymbol{w}=\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$,

we get $\boldsymbol{w}=\begin{bmatrix}w_0\\ w_1\end{bmatrix}=\begin{bmatrix}26.781\\ 0.644\end{bmatrix}$.

Student Data

Student	x_i	y_i
1	95	85
2	85	95
3	80	70
4	70	65
5	60	70

Example of Linear Regression

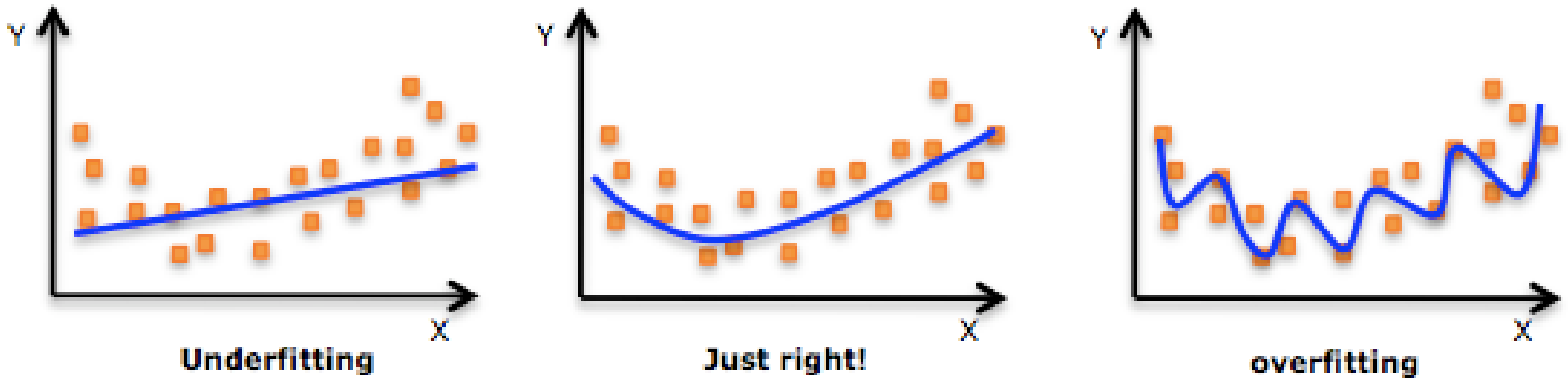
- Prediction: If a student made an 80 on the aptitude test, what grade would we expect her to make in statistics?

- Answer: Predicted Statistics Grade is computed as

$$\text{Statistics grade} = [1\ 80]\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = 1 \times 26.781 + 80 \times 0.644 = 78.288$$

- R example on linear regression
- linearRegEx.R

Complexity of Model



How well you are going to predict the future data from the same distribution

Under-fitting

- Under-fitting occurs if the complexity of the model is lower than necessary.
  - Scenario-1: We may be using a linear model, whereas the data requires a nonlinear model.
  - Scenario-2: We may be using right hypothesis (linear or nonlinear) but the number of variables might be falling short of what is required. For example, to predict the income of a person, “age” alone may not be sufficient.

- We can detect it by checking if the model fitting error on the training data is high.

Example-1:Underfitting/Overfitting

- For the income prediction of a person, say “age” alone is not sufficient.
- Assuming our dataset has information about “age”, “sex”, “education”, we could add them as explaining variables. Our model becomes more interesting and more complex.
- We train a model and find that it is explaining the data better now but still not good enough.
- So we add even more variables: location, profession of parents, social background, number of children, weight, preferred colour, best meal, last holidays destination,… and so on.
- Our model will do good but it is probably over-fitting, i.e. it will probably have poor prediction on unseen data: it has learnt too much specifics of training data and has probably learnt the background noise.

(source: https://www.quora.com/What-is-regularization-in-machine-learning)

Example-2: Overfitting

- Let's say you attend a symphony and want to get the clearest sound possible!

- So you buy a super-sensitive microphone and hearing aid to pick up all the sounds in the auditorium.

- Then you start to overfit:
  - you hear your neighbours shuffling in their seats;
  - the musicians turning their pages and;
  - even swishing of the conductor's coat jacket.

- Fitting a perfect model is only listening to Symphony (signal) and not to the background noise.

(source: https://www.quora.com/What-is-an-intuitive-explanation-of-overfitting)

Bias Variance Decomposition

- Let us assume our data (x, y) has the true relation $y = f(x) + \epsilon$, where ϵ is measurement noise in y with mean zero and variance σ_ϵ^2.

- Also assume that we are fitting a hypothesis function (or model) $h_D(x)$ using dataset D. Then the expected loss (or risk) has three components.

Risk =

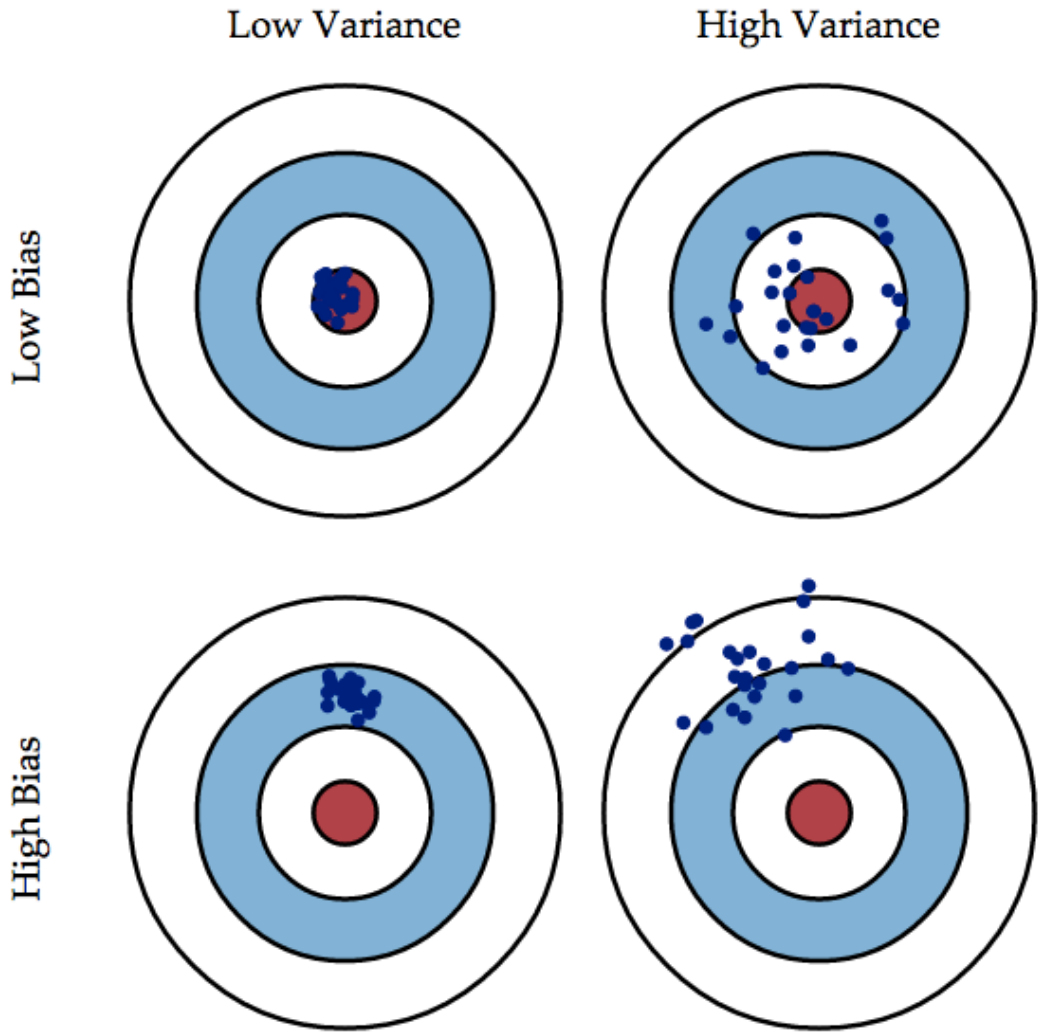
$$\underbrace{\{E_D[h_D(x) - f(x)]\}^2}_{\text{(bias)}^2} + \underbrace{E_D[\{h_D(x) - E_D[h_D(x)]\}^2]}_{\text{variance}} + \sigma_\epsilon^2$$

noise

We can minimize only bias and variance

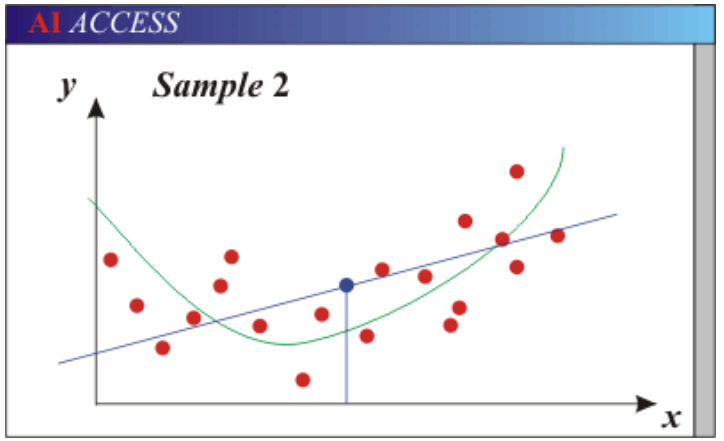
Irreducible error

Bias and Variance

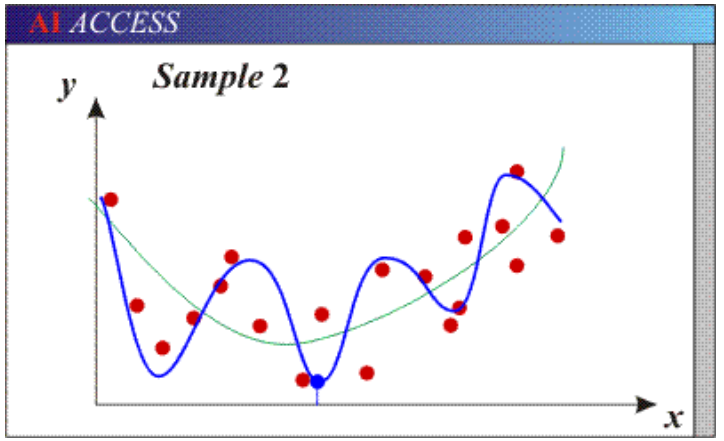


Bias and Variance

- Models with too few parameters are inaccurate because of a large bias (not enough flexibility): the case of under-fitting.

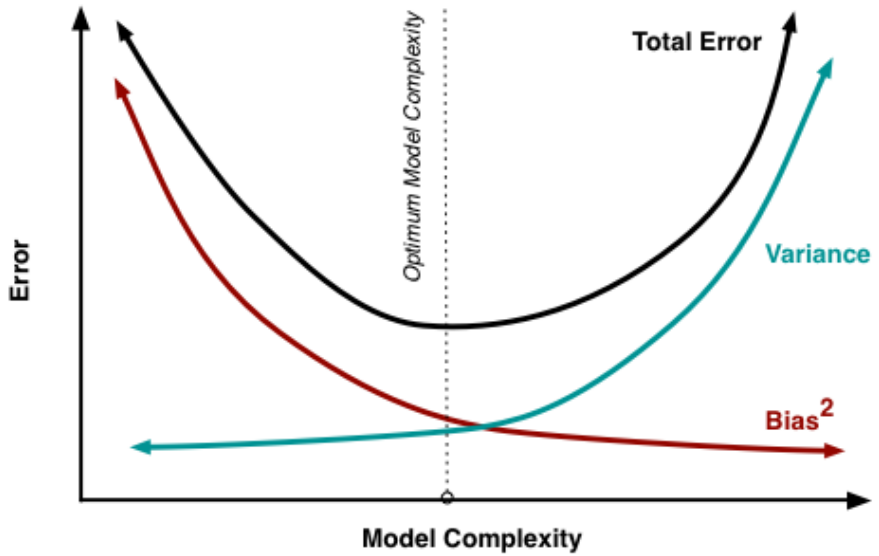


- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample): the case of over-fitting.



Bias-Variance Trade-off

- Low bias implies high variance, and high bias implies low variance
- We need to find the "sweet spot" where $Risk = bias^2 + variance + noise$ is the minimum.
- The minimum error is at the right model complexity.



Overfitting in Linear Models

- When using Linear Models, can we still overfit?
- Depends! Depends on what's our model complexity.
- In linear models, the model complexity grows with the number of features.
- Using all data dimensions as features may fit the model on not only true patterns (signal) but also on backgound noise.
- Regularization is a technique used to control the model complexity.

Regularization

- A "regularizer" is an additional term in the loss function to avoid overfitting.
- It is called regularizer since it tries to keep the parameters more normal or regular.
- In other words, it does not allow regression coefficients (or weights) to take excessively large value.
- It is a way to guide the training process to prefer certain types of weights over others.

$$\underset{\boldsymbol{w}}{\text{minimize}} \ \frac{1}{n}\sum_{i} L\left(y_i, \boldsymbol{x}_i^T \boldsymbol{w}\right) + \lambda \, Regularizer(\boldsymbol{w})$$

Regularized Linear Models

- Linear model: $y = w_0 + \sum_{j=1}^{d} w_j x_j$
- Should we allow all possible weights?
- Or impose any preferences?
- What makes a simpler linear model?

Regularized Linear Models

- Linear model: $y = w_0 + \sum_{j=1}^{d} w_j x_j$
- Our preference: we do not want huge weights (i.e. do not want to over-rely on any one feature).
- If weights are huge, a small change in a feature would result in a large change in the prediction!
- In fact, since we may even have irrelevant features, we want some of the weights to be zero to discard some features.

Regularized Linear Regression

- In the following formulation

$$\underset{\boldsymbol{w}}{\text{minimize}} \frac{1}{n}\sum_{i} L\left(y_i, \boldsymbol{x}_i^T \boldsymbol{w}\right) + \lambda\, Regularizer(\boldsymbol{w})$$

how do we penalize large weights or encourage small/zero weights ?

- What should be our "regularizer function"?

- Popular regularizer functions:
  - $Regularizer(\boldsymbol{w}) = \sum_j |w_j| = \|\boldsymbol{w}\|_1$ (l_1-norm) --- encourages 0 weights (sparsity)
  - $Regularizer(\boldsymbol{w}) = \sum_j |w_j|^2 = \|\boldsymbol{w}\|_2$ (l_2-norm) --- penalizes large weights

L_1 Regularization (LASSO)

- It is therefore quite common to use the following formulation

$$\underset{\boldsymbol{w}}{\text{minimize}} \frac{1}{n}\sum_i L\left(y_i, \boldsymbol{x}_i^T \boldsymbol{w}\right) + \lambda_1 \|\boldsymbol{w}\|_1$$

- It is also known as LASSO.

- LASSO stands for **Least Absolute Shrinkage and Selection Operator**.

L_2 Regularization (Ridge)

- Another common formulation is

$$\underset{\boldsymbol{w}}{\text{minimize}} \frac{1}{n}\sum_i L\left(y_i, \boldsymbol{x}_i^T \boldsymbol{w}\right) + \lambda_2 \|\boldsymbol{w}\|_2^2$$

- It is also known as Ridge Regularization.

Combination of L_1 and L_2 Regularizations

- A combination of L_2 and L_1 Regularization is also used

$$\underset{\boldsymbol{w}}{\text{minimize}} \frac{1}{n}\sum_i L\left(y_i, \boldsymbol{x}_i^T \boldsymbol{w}\right) + \lambda_1 \|\boldsymbol{w}\|_1 + \lambda_2 \|\boldsymbol{w}\|_2^2$$

- This is known as Elastic Net.

- LASSO and ridge regularization are special cases of Elastic net for $\lambda_2 = 0$ and $\lambda_1 = 0$.

- Elastic net overcomes a limitation of LASSO: When presented with few samples in high dimension spaces ($d > n$ case), LASSO selects at most n variables before it saturates.

Linear and Logistic Regression

(Loss functions)

- For Linear regression: we use square loss function, i.e.

$$L\left(y_i, \boldsymbol{x}_i^T \boldsymbol{w}\right) = \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right)^2$$

Regularized Linear Regression

- We solve the following optimization

$$\underset{\boldsymbol{w}}{\text{minimize}} \sum_{i} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right)^2 + \lambda_1 \|\boldsymbol{w}\|_1 + \lambda_2 \|\boldsymbol{w}\|_2^2$$

- For ridge regularization (when $\lambda_1 = 0$), the solution is closed form and is given as

$$\boldsymbol{w} = \left(\boldsymbol{X}^T \boldsymbol{X} + \lambda_2 \boldsymbol{I}\right)^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

- For LASSO and elastic net, we have to perform iterative optimization.
- Since L_1 norm is non-differentiable, promixal gradient is used.

Regularized Logistic Regression

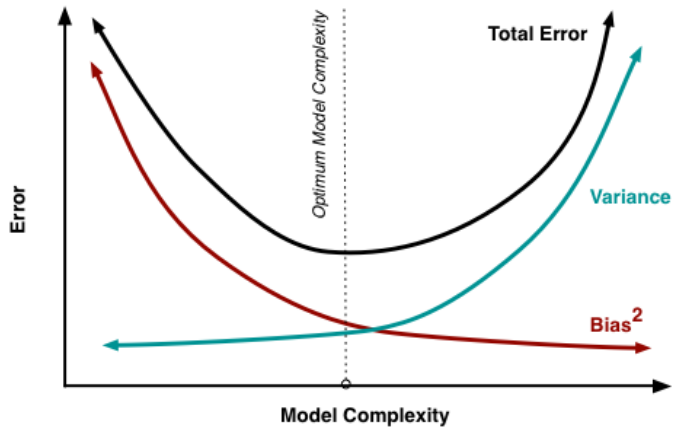
- We solve the following optimization

$$\min_{\boldsymbol{w}} \sum_{i} \log\left(1 + \exp\left(-y_i \boldsymbol{x}_i^T \boldsymbol{w}\right)\right) + \lambda_1 \|\boldsymbol{w}\|_1 + \lambda_2 \|\boldsymbol{w}\|_2^2$$

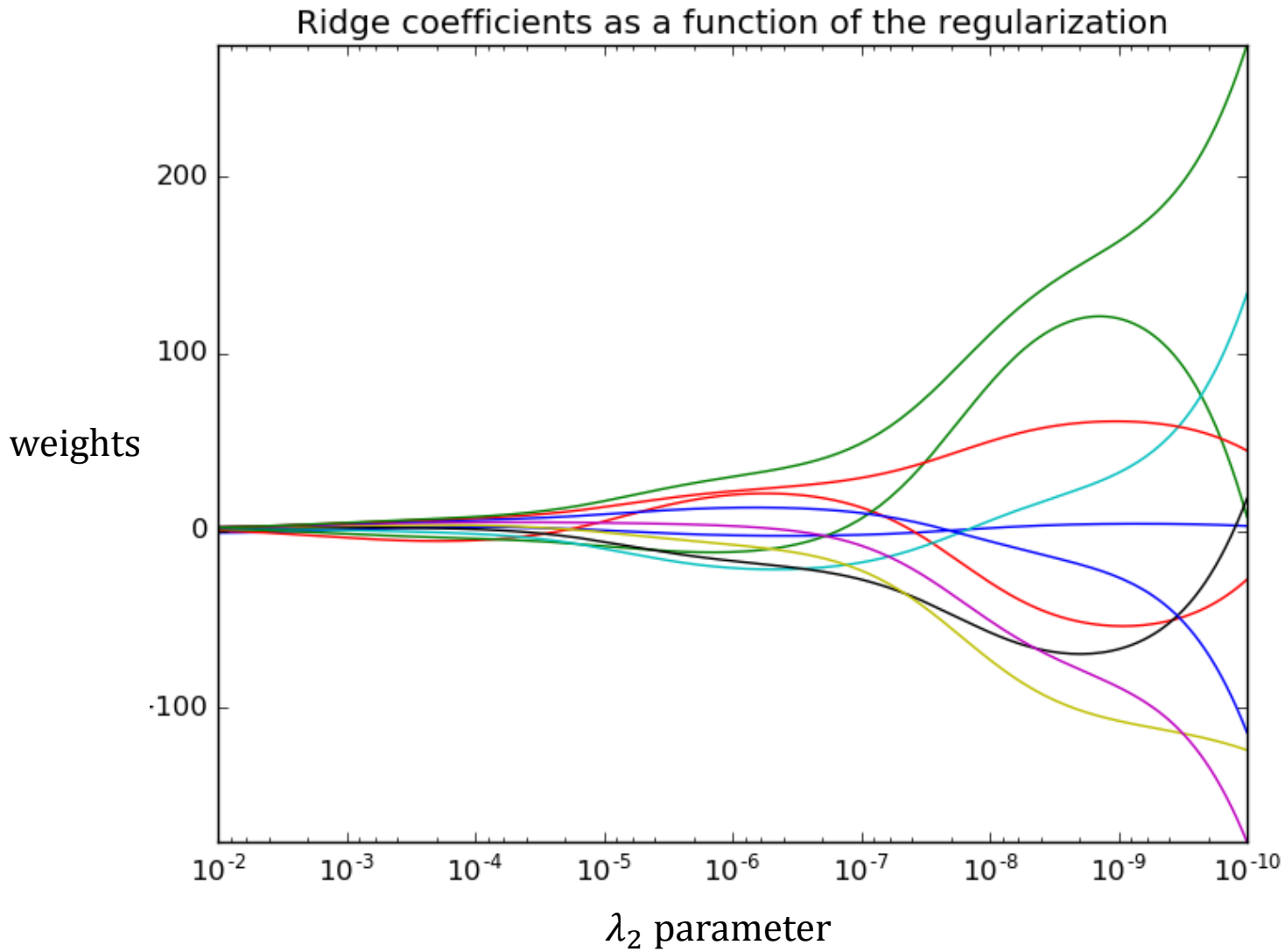
- For regularized logistic regression, we always have to perform iterative optimization.

Effect of Regularization on Bias and Variance

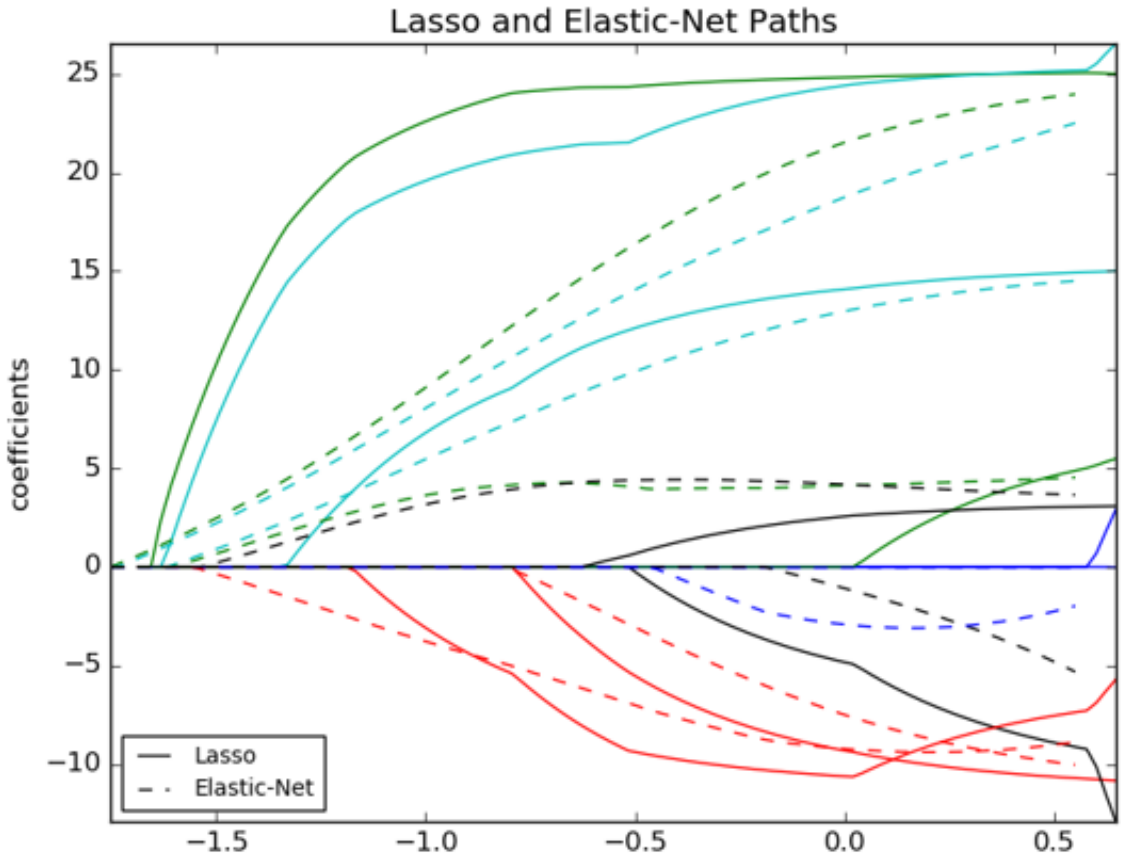
- Regularization increases bias in our model. We are only partially listening to our training data! Why it might make sense?
- However, it greatly reduces the variance.
- It is useful when the net effect (i.e. $bias^2 + variance$) reduces.



Ridge Regularization Effect



Lasso and Elastic Net Regularization Effect



$-log\lambda_1$

LASSO and Feature Selection

- Because L_1 regularization shrinks the weights of noisy dimensions to zero, these dimensions do not participate in the prediction model.

- Only those dimensions that have nonzero weights participate in the prediction.

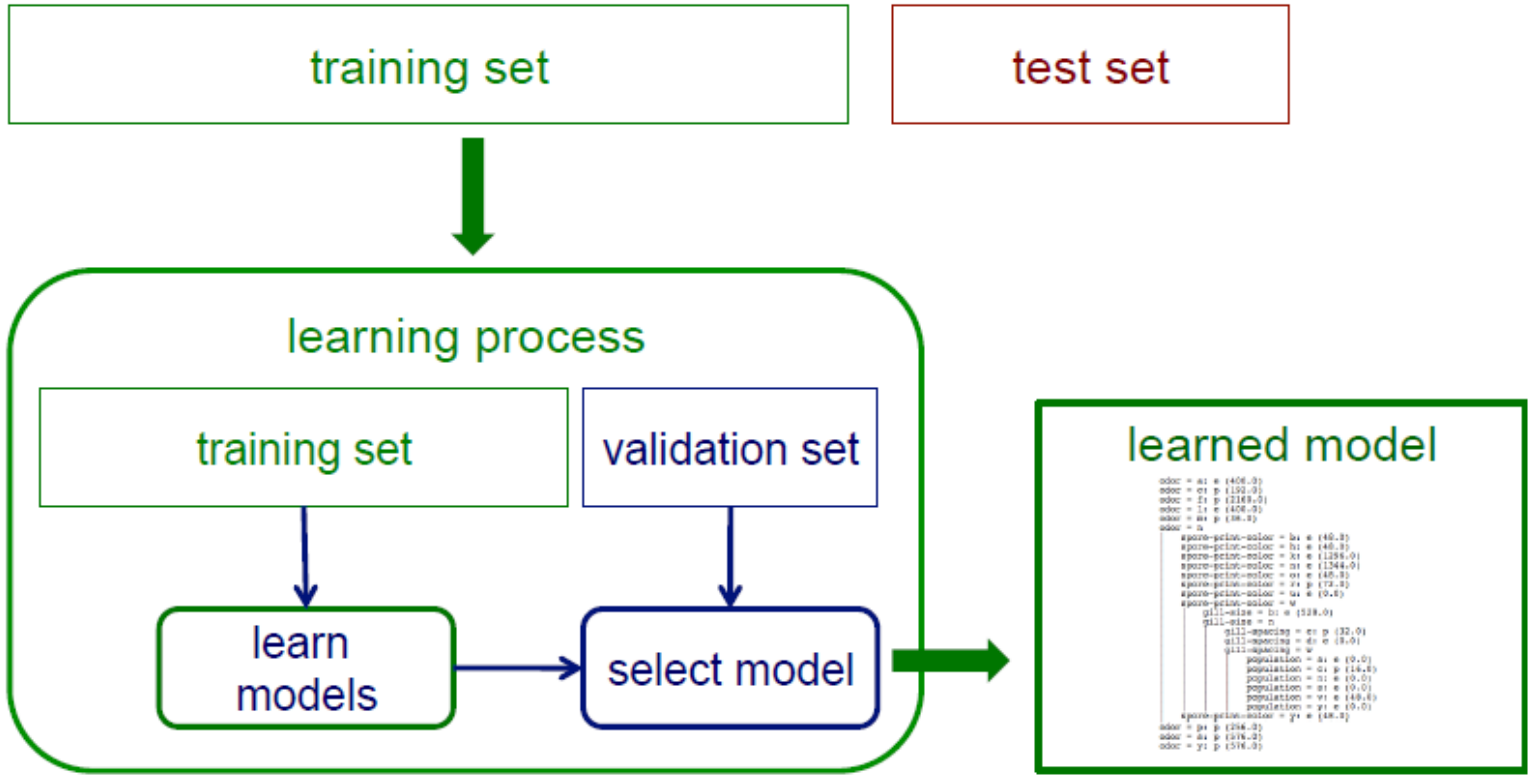
- Therefore, LASSO is also used to select predictive features among all dimensions. This is feature selection property of LASSO.

- R example on regression
- LAssoRidgeElasticRegularisationEx.R

How to set hyperparameters in the model?

Finding the Best Hyperparameter
(Model Selection)

To search for the best hyperparameters, we need to partition training data into separate training and validation sets.



Finding the Best Hyperparameter
(Model Selection)

- We first decide a possible range for each hyperparameter.
  - For example, in case of LASSO, we might assume that the regularization parameter λ_1 (or α) will be between 10^{-3} to 1.

- We then define a search grid within the specified range.
  - For Lasso example, we may have $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$.

- Next, we train a model using each hyperparameter value from the search grid and assess its performance on a validation set (taken out from training set).

- We compute the performance on the validation set for each hyperparameter value and select the one with the best performance.

Finding the Best Hyperparameter
(Model Selection)

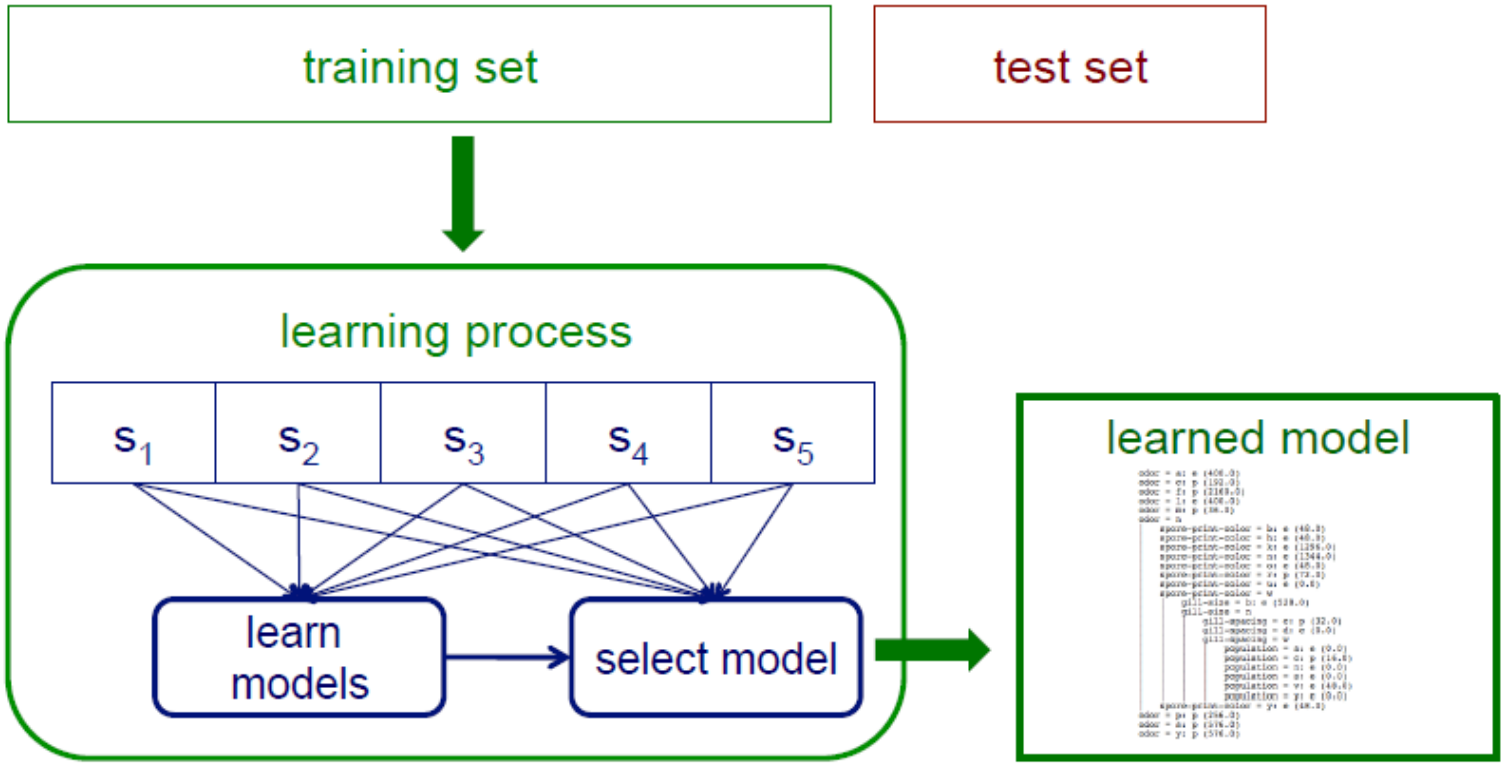
- All those techniques that we discussed for model assessment is also applicable for training/validation set splitting.
  - Random subsampling
  - Stratified subsampling
  - Cross-validation

- We are still assessing how a particular hyperparameter is doing on validation set.

- This step is internal to the learning process and different from model assessment on the test data.

Internal Cross-validation (Example)

Instead of using a single validation set, we can use cross-validation within a training set to select the best set of hyperparameters.



Example of Internal cross-validation

- Say, we want to do 10-fold Cross-validation to estimate the model performance of Elastic Net model.

- We can divide the data into 10 equal subsamples and then "train the model" using 9 subsamples and test the model using the 10^{th} subsample. We repeat this 10-times using each subsample for the test purpose and all other subsamples for the training.

- In the above "train the model" step, best hyperparameter can be selected using an internal cross-validation. Say, we want to use 5-fold cross-validation for this. Then for each possible hyperparameter set, we compute 5-fold CV accuracy and select the best hyperparameter set.

Hyperparameter Search

- We can select the best hyperparameter set by searching/or optimizing over all possible values.

- Different ways to navigate the hyperparameter space:
  - Grid-search (not so efficient) – This is what we are using!
  - Bayesian optimization (efficient in general) [Snoek et al. (2012)]

For better understanding, you can read this article (https://arimo.com/data-science/2016/bayesian-optimization-hyperparameter-tuning/).