Reviewing the learning models

ML: Main components

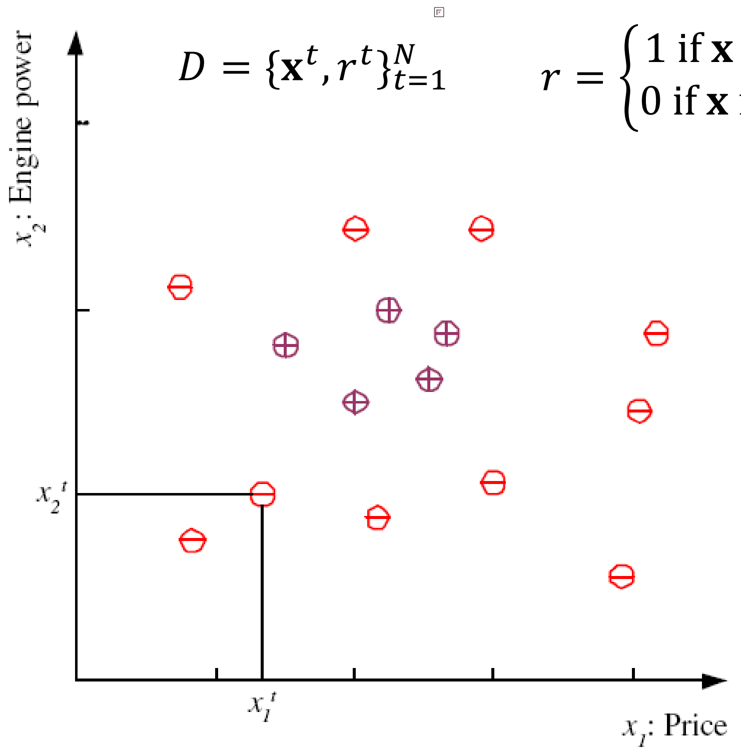
- Model representation (hypothesis space, aka learning model):
  - Structure of the functional form of the knowledge to be extracted (Trees, partition, graph,…)

- Search method (learning algorithm):
  - Strategy used to explore the search space and find the optimal or "good" model (backpropagation, local search, divide-and-conquer, greedy search, …)

- Objectif function (cost function):
  - Measure the quality of the model (Gini, Entropy, RMSE, logloss, …)

Hypothesis space (class) $\mathcal{H}$

- Let start with a sample problem of binary classification



$$D = \{\mathbf{x}^t, r^t\}_{t=1}^N \qquad r = \begin{cases} 1 \text{ if } \mathbf{x} \text{ is positive} \\ 0 \text{ if } \mathbf{x} \text{ is negative} \end{cases}$$

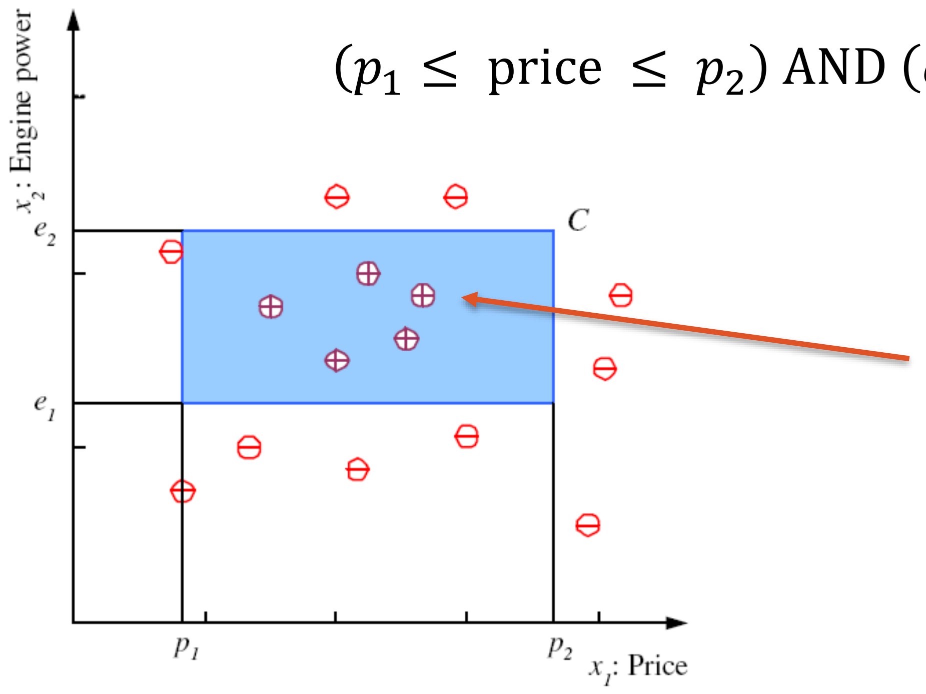
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- Prediction of "family car"
  - Prediction: Is car x a family car?
- Knowledge extraction:
  - What do people expect from a family car?
- Output:
  - Positive (+) and negative (–) examples
- Input representation:
  - x1: price, x2 : engine power

Hypothesis space $\mathcal{H}$

- Rectangle learning:
  - Hypothesis is any axis aligned Rectangle. Inside rectangle is positive

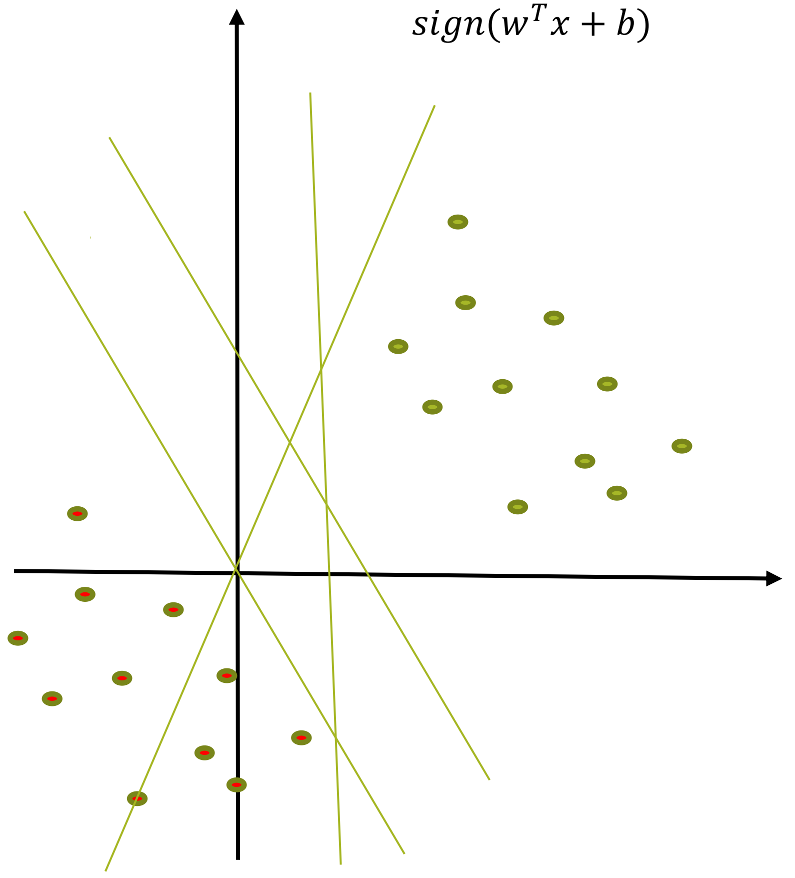
$$(p_1 \leq \text{ price } \leq p_2) \text{ AND } (e_1 \leq \text{ engine power } \leq e_2)$$

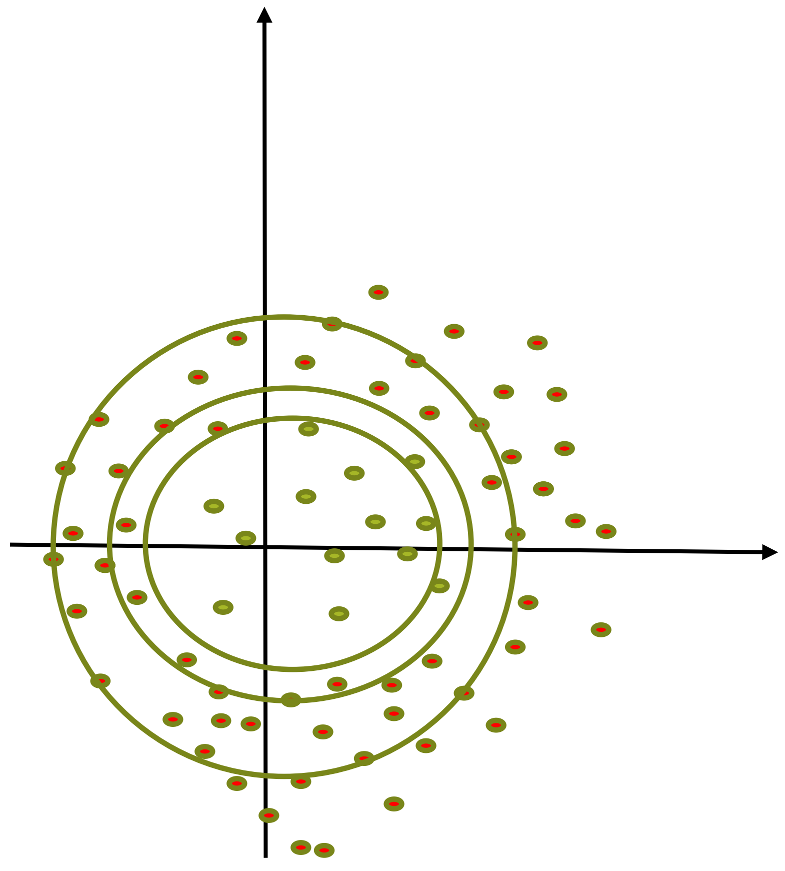


$$h(\mathbf{x}) = \begin{cases} 1 \text{ if } h \text{ says } \mathbf{x} \text{ is positive} \\ 0 \text{ if } h \text{ says } \mathbf{x} \text{ is negative} \end{cases}$$

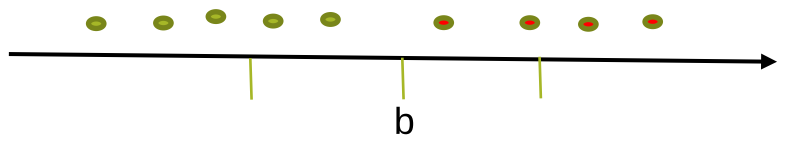
- $\mathcal{H}$ is the set of all possible rectangle called the hypothesis space

Hypothesis space (class) $\mathcal{H}$





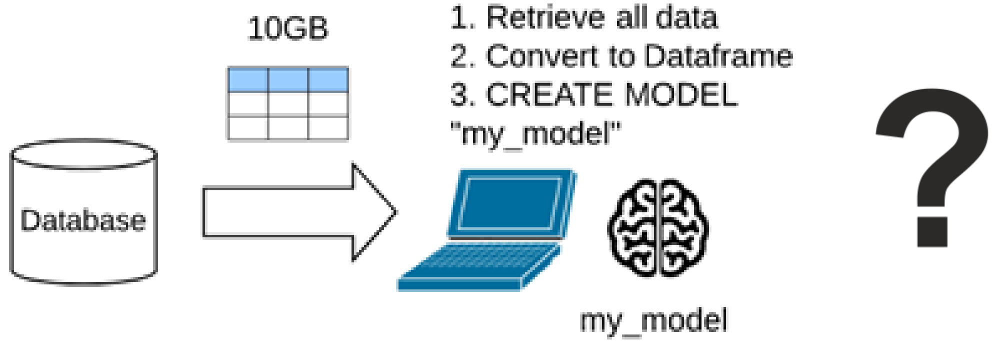
$sign(x^2 - b)$

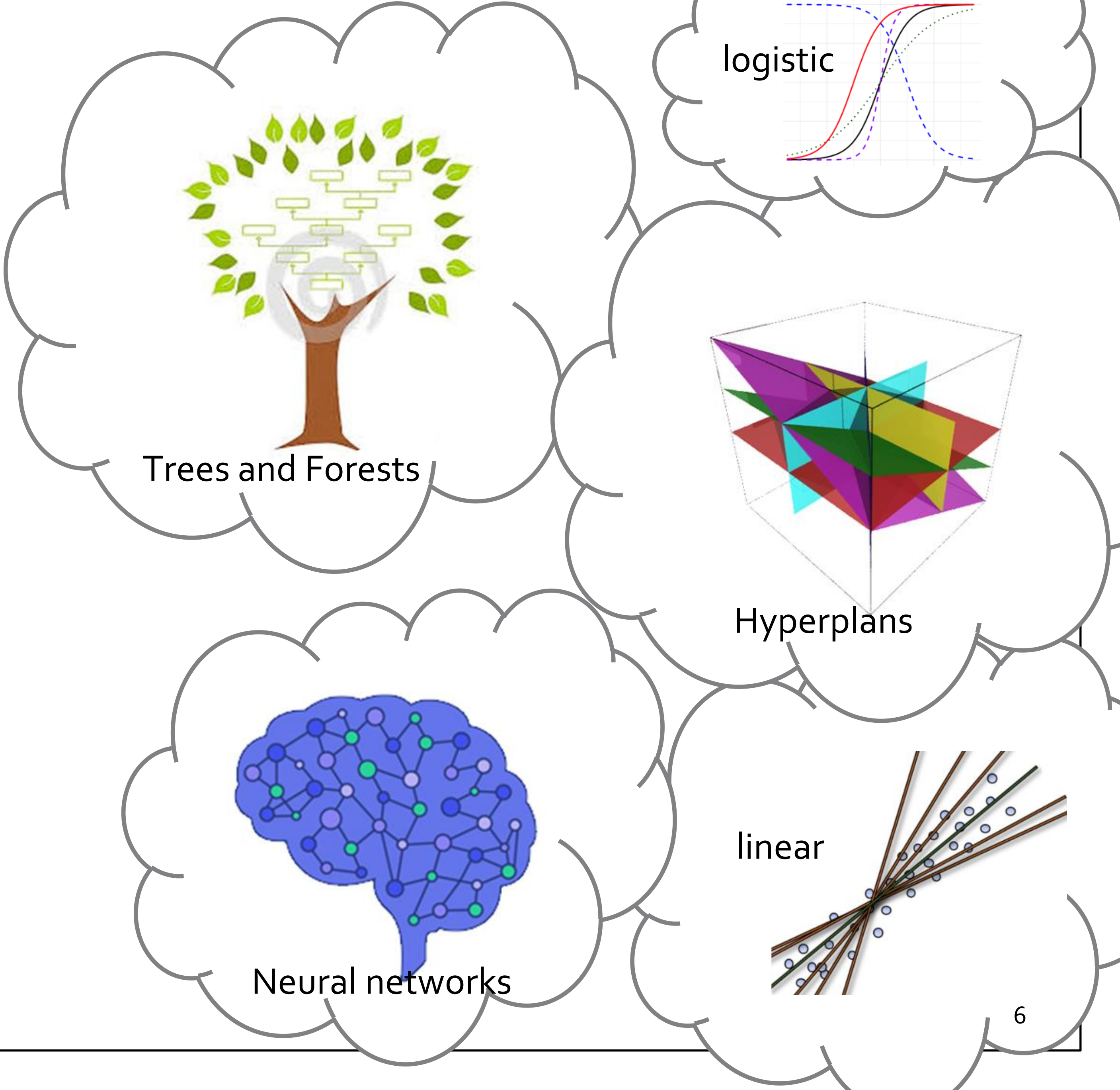


$sign(x - b)$

Hypothesis space $\mathcal{H}$

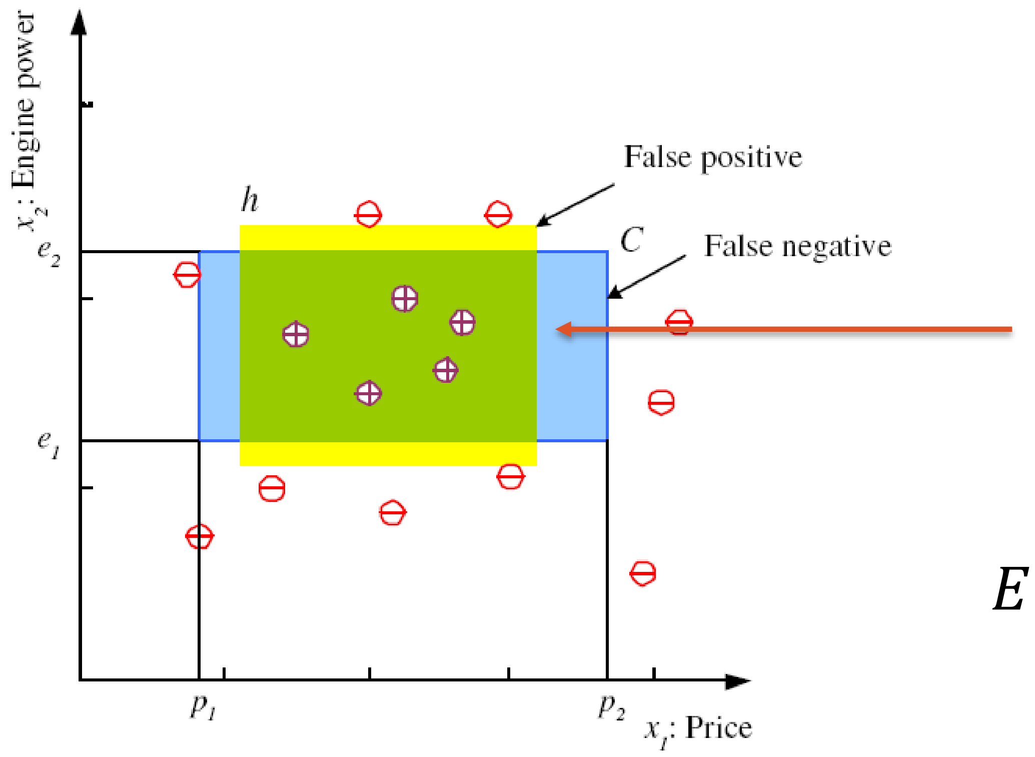
Which hypothesis space ?





Hypothesis space $\mathcal{H}$

- Assume that *h* is given: what is the Error of *h* on $\mathcal{H}$?



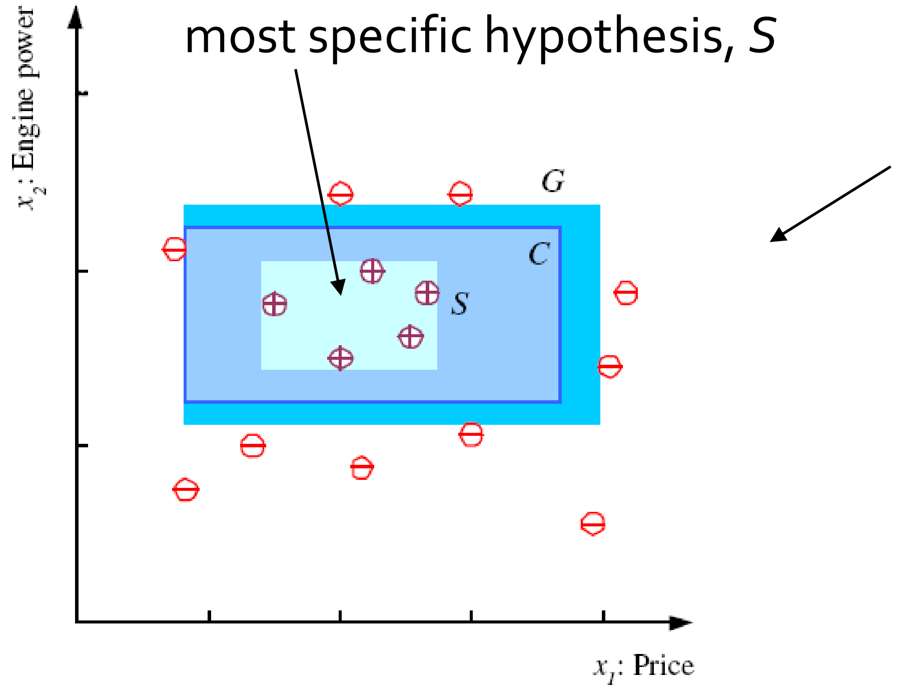
$$D = \{\mathbf{x}^t, r^t\}_{t=1}^N \subset X$$

$$h(\mathbf{x}) = \begin{cases} 1 \text{ if } h \text{ says } \mathbf{x} \text{ is positive} \\ 0 \text{ if } h \text{ says } \mathbf{x} \text{ is negative} \end{cases}$$

$$E(h|D) = \sum_{t=1}^{N} 1(h(\mathbf{x}^t) \neq r^t)$$

Hypothesis space $\mathcal{H}$

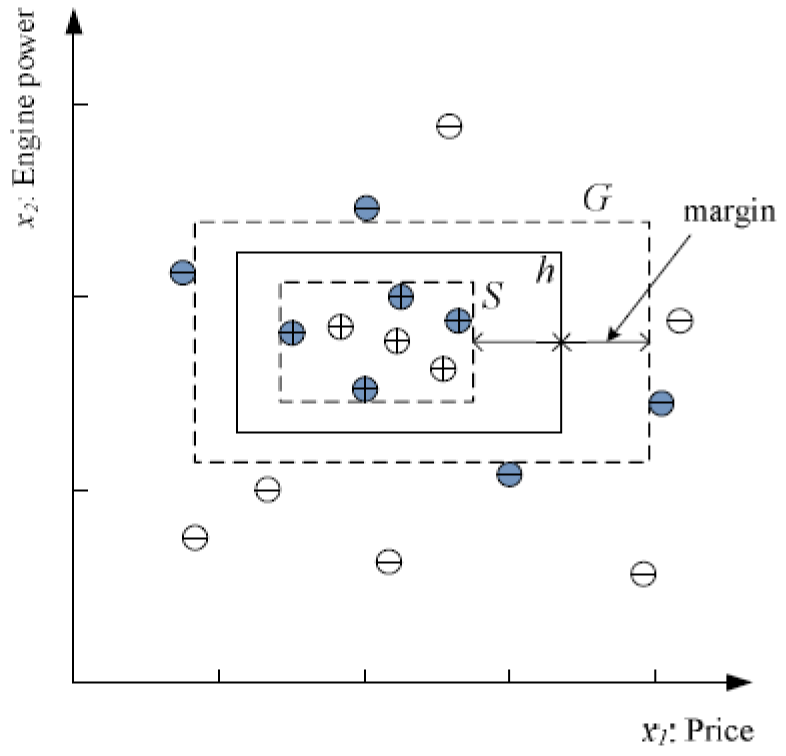
- The version space (VS) with the respect to a hypothesis space $\mathcal{H}$ and the training set D is the subset of hypothesis from $\mathcal{H}$ consistant with the training set D



$h \in \mathcal{H}$, between S and G is **Consistent ($E(h|\text{D}) = 0$)and** make up the version space (Mitchell, 1997)

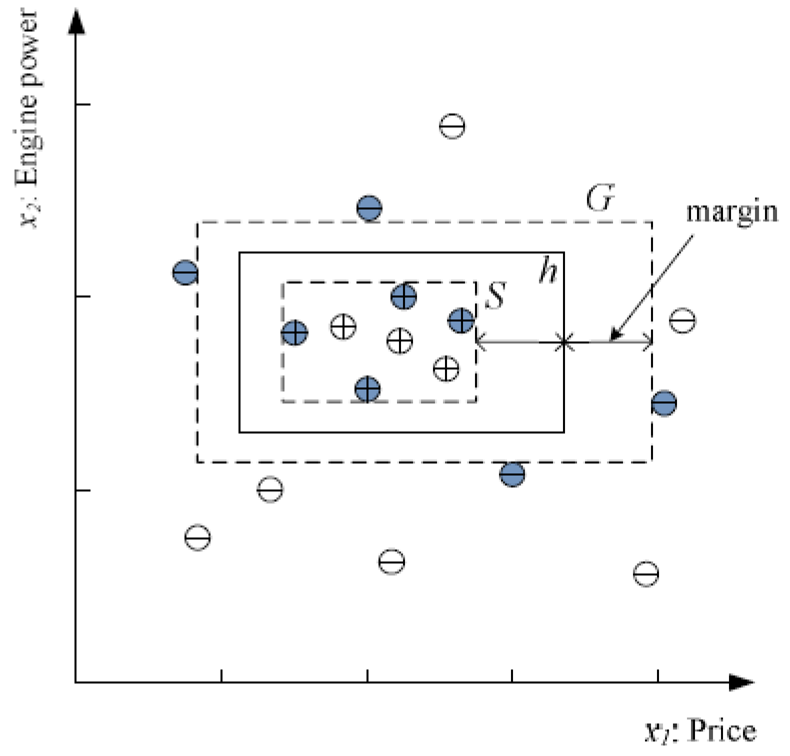
Hypothesis space $\mathcal{H}$

- To generalize better, we need to choose *h* with largest margin



Learning algorithm (or learner)

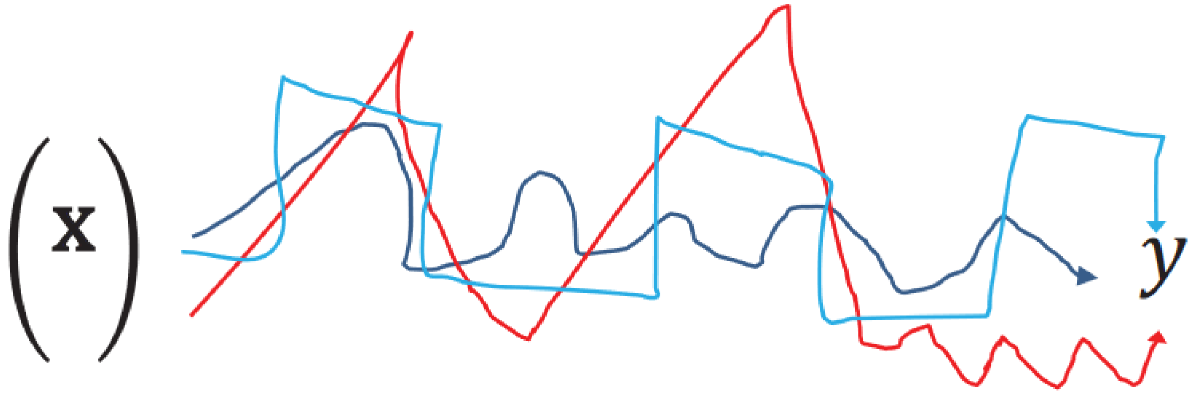
- But for a given dataset X, how to compute h?



Learning algorithm (or learner)

Which learning algorithms? Approaches to learn classifiers/predictors

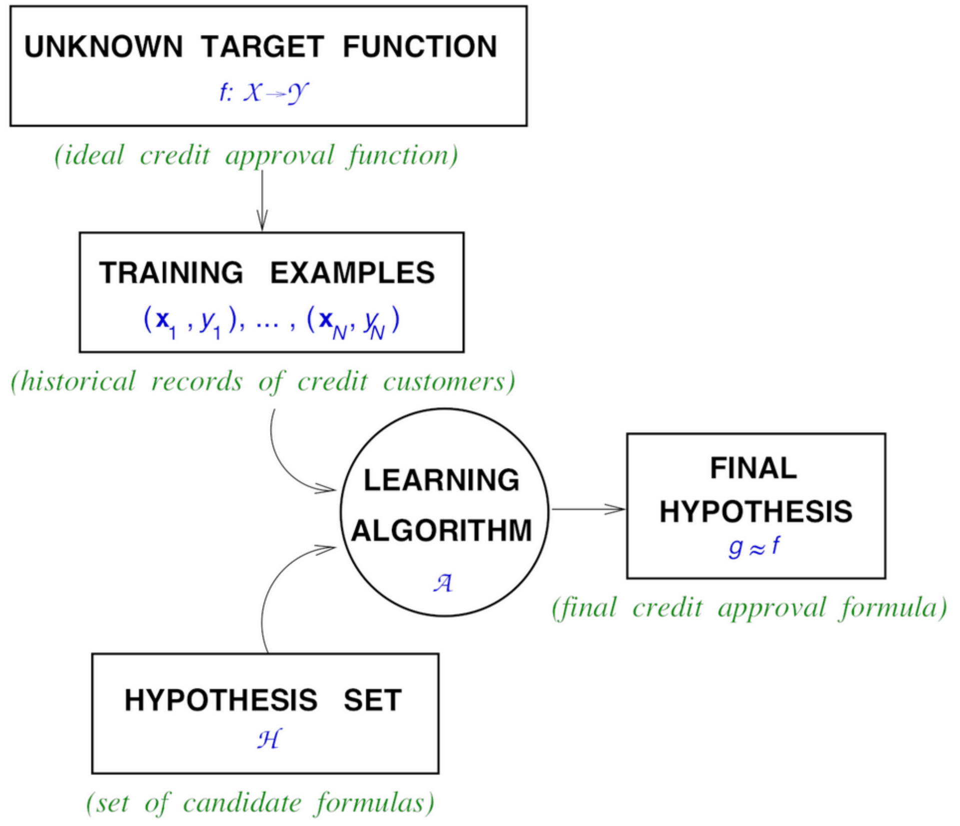
$$\begin{pmatrix}\mathbf{X}\end{pmatrix} \xrightarrow{f(\mathbf{X},\alpha)\,?} y$$

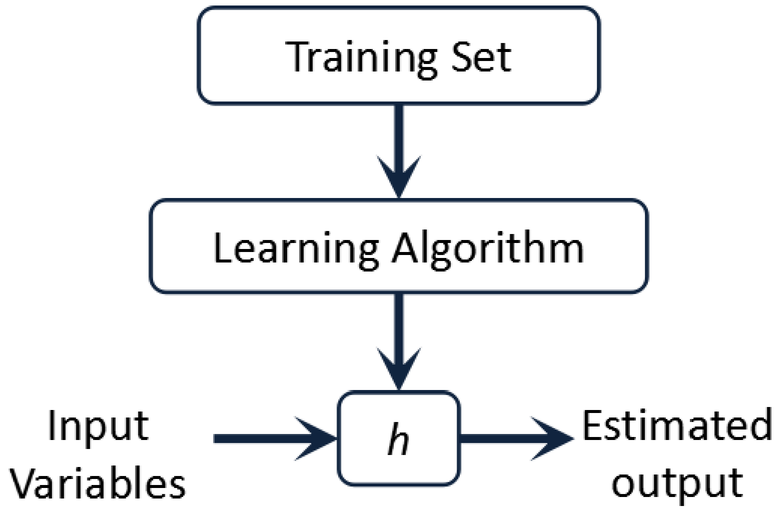


Support Vector Machines *Random Forest* *Artificial Neural Networks*

Hypothesis space $\mathcal{H}$

- Each hypothesis space has it learning algorithm. The hypothesis set and learning algorithm together, is called a **Learning Model**


UNKNOWN TARGET FUNCTION
$f: \mathcal{X} \to \mathcal{Y}$
(ideal credit approval function)
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
(historical records of credit customers)
LEARNING ALGORITHM
$\mathcal{A}$
FINAL HYPOTHESIS
$g \approx f$
(final credit approval formula)
HYPOTHESIS SET
$\mathcal{H}$
(set of candidate formulas)



Learning Model

- The hypothesis set and learning algorithm together, is called a **Learning Model**
  - **Question 1: what is the best learning model?**
  - Question2: how good is a learning model, given dataset ?
  - Question 3: what is the capacity (complexity, expressive power, richness, or flexibility) of a learning model ?

Learning Model

- Decision trees are not always most accurate on test error.

- What is the “best” machine learning model?

- An alternative measure of performance is the generalization error:
  - Average error over all x_i vectors that are not seen in the training set.
  - “How well we expect to do for a *completely unseen* feature vector”.

- No free lunch theorem (proof ??):

Q1: No Free Launch (NFL) theorem

In his 1996 paper, “**The Lack of A Priori Distinctions Between Learning Algorithms”**, Wolpert introduced the NFL theorem for supervised machine learning.

“ The theorem states that given a noise-free dataset, for any two machine learning algorithms A and B (learning model), the average performance of A and B will be the same across all possible problem instances drawn from a uniform probability distribution”

- There is **no** “best” model achieving the best generalization error for every problem.
- If model A generalizes better to new data than model B on one dataset, there is another dataset where model B works better.

Q1: No Free Launch (NFL) theorem

- Does it mean that all algorithms are equal? No, of course not.
  - Some algorithms may generally perform better than others on certain types of problems, but every algorithm has disadvantages and advantages due to the prior assumptions that come with that algorithm.
  - XGBoost may win hundreds of Kaggle competitions yet fail miserably at forecasting tasks because of the limiting assumptions involved in tree-based models.
  - Neural networks may perform really well when it comes to complex tasks like image classification and speech detection, yet suffer from overfitting due to their complexity if not trained properly.

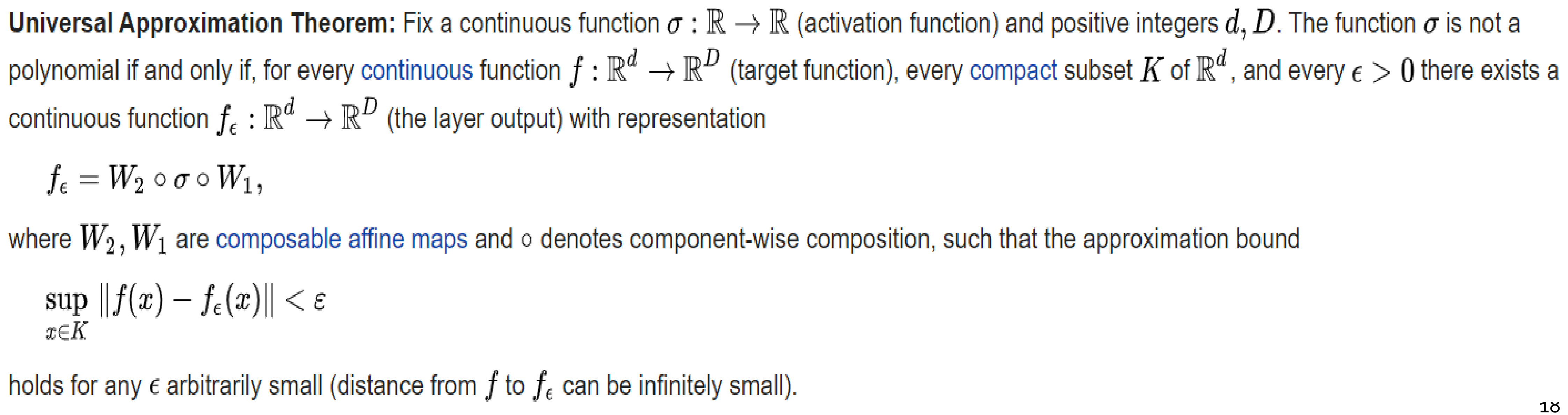
- Machine learning research:
  - Large focus on models that are useful across many applications.

Q1: No Free Launch (NFL) theorem

- In practice, this is what "no free lunch" means for you:
  - No single algorithm will solve all your machine learning problems better than every other algorithm.
  - Make sure you completely understand a machine learning problem and the data involved before selecting an algorithm to use.
  - All models are only as good as the assumptions that they were created with and the data that was used to train them.
  - Simpler models like logistic regression have more bias and tend to underfit, while more complex models like neural networks have more variance and tend to overfit.
  - The best models for a given problem are somewhere in the middle of the two bias-variance extremes.
  - To find a good model for a problem, you may have to try different models and compare them using a robust cross-validation strategy.

Q1: Universal Approximation Theorem

- A feedforward network with a single layer is sufficient to represent any function, but the layer may be infeasibly large and may fail to learn and generalize correctly.



Universal Approximation Theorem: Fix a continuous function $\sigma : \mathbb{R} \to \mathbb{R}$ (activation function) and positive integers d, D. The function σ is not a polynomial if and only if, for every continuous function $f : \mathbb{R}^d \to \mathbb{R}^D$ (target function), every compact subset K of $\mathbb{R}^d$, and every $\epsilon > 0$ there exists a continuous function $f_\epsilon : \mathbb{R}^d \to \mathbb{R}^D$ (the layer output) with representation

$$f_\epsilon = W_2 \circ \sigma \circ W_1,$$

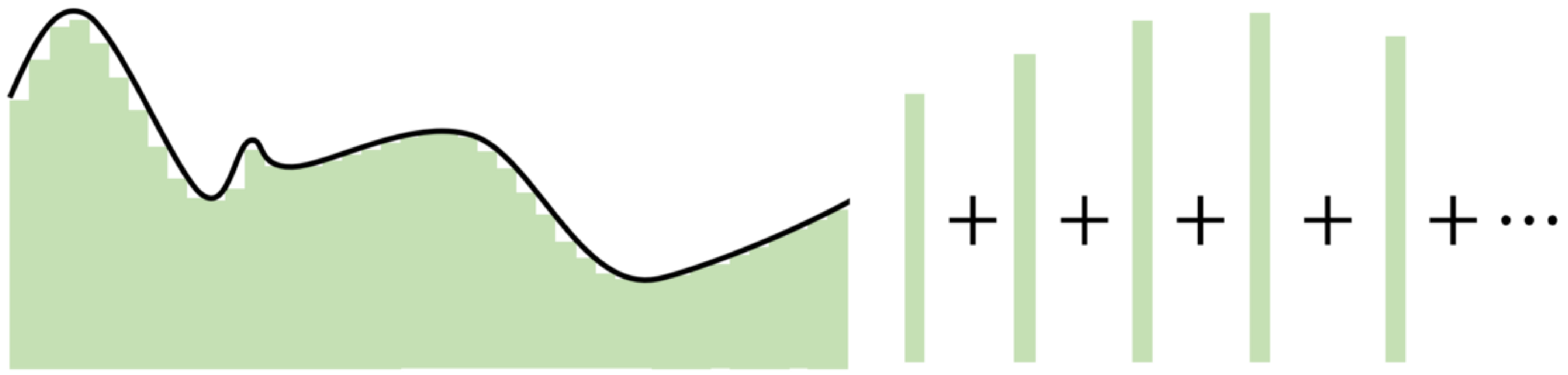
where W_2, W_1 are composable affine maps and $\circ$ denotes component-wise composition, such that the approximation bound

$$\sup_{x \in K} \|f(x) - f_\epsilon(x)\| < \varepsilon$$

holds for any ϵ arbitrarily small (distance from f to f_ϵ can be infinitely small).

Q1: Universal Approximation Theorem

- The key point in the Universal Approximation Theorem is that instead of creating complex mathematical relationships between the input and output, it uses simple linear manipulations to divvy up the complicated function into many small, less complicated pieces, each of which are taken by one neuron.



Q1: Universal Approximation Theorem.

- A model with too little capacity cannot learn the problem, whereas a model with too much capacity can learn it too well and overfit the training dataset. Both cases result in a model that does not generalize well.
  - The capacity of a neural network model is defined by both it’s structure in terms of nodes and layers and the parameters in terms of its weights. Therefore, we can reduce the complexity of a neural network to reduce overfitting in one of two ways:
    - Change network complexity by changing the network structure (number of weights): grid search until a suitable number of nodes and/or layers
    - Change network complexity by changing the network parameters (values of weights): keeping network weights small

Q1: Universal Approximation Theorem.

- The number of hidden neurons should be between the size of the input layer and the output layer.
  - The most appropriate number of hidden neurons is sqrt(input layer nodes * output layer nodes)

- Machine learning research:
  - Network Architecture Search (NAS)
  - AutoML ??

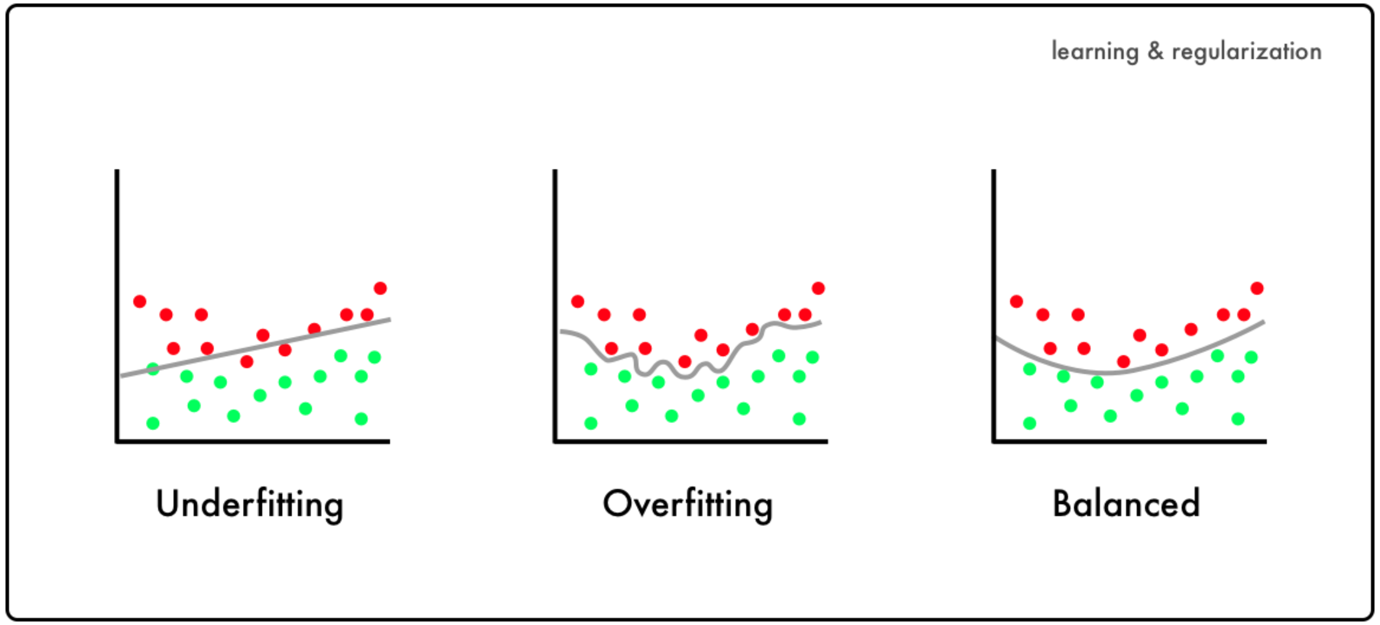
Learning Model

- The hypothesis set and learning algorithm together, is called a **Learning Model**
  - Question 1: what is the best learning model?
  - **Question 2: how good is a learning model, given a dataset ?**
  - Question 3: what is the capacity (complexity, expressive power, richness, or flexibility) of a learning model ?

Objective function

- At the end of the day, the learning algorithm in a given hypothesis space will give me a candidate model
- If you look too hard at a dataset, you will find something, but it might not generalize beyond the data you're looking at (unseen data)

But
HOW GOOD IS YOUR MODEL



Back to the real world

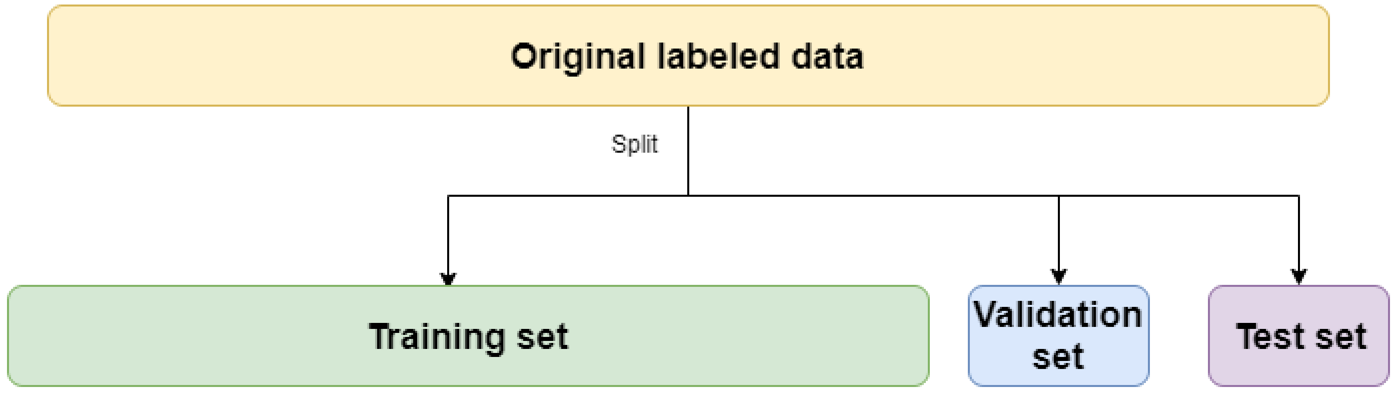
- We aspire to understand the workings of an event A, by
    - Collecting a set S of data (X,Y)
    - Formulating hypotheses, and
    - Designing models

If (X,Y) follows a probability distribution D, our objective is to find D which is generally impossible, however we can estimate D with some errors using the set S of collected data.

- Is it the write way to do the learning ?

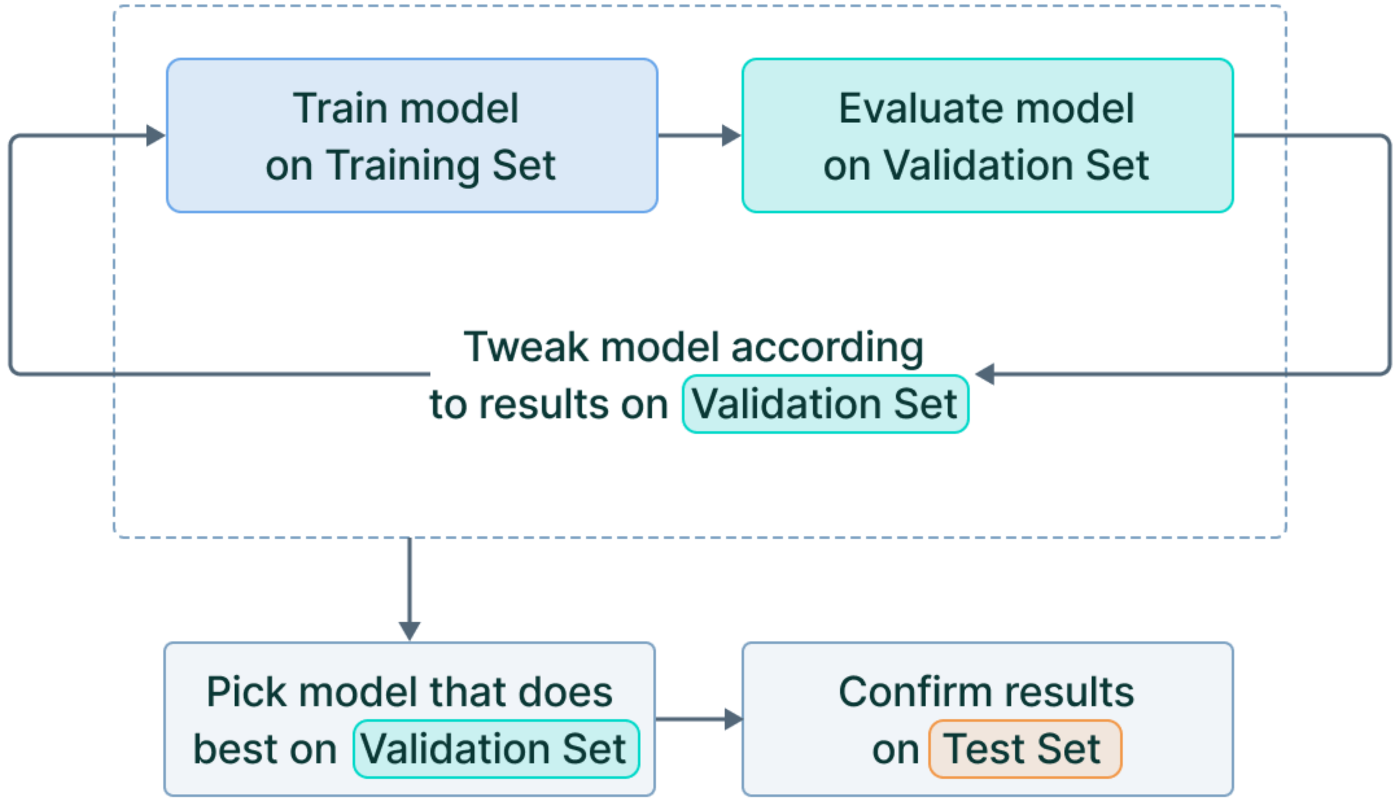
Training, validation and Testing sets

- Splitting the original data



Training, validation and Testing sets

- Evaluation workflow


Train model on Training Set
Evaluate model on Validation Set
Tweak model according to results on Validation Set
Pick model that does best on Validation Set
Confirm results on Test Set

Goal of Machine Learning

- In machine learning:
  - What we care about is the test error!

- Memorization vs learning:
  - Can do well on training data by memorizing it.
  - You've only learned if you can do well in new situations.

Golden Rule of Machine Learning

- Even though what we care about is test error:
  - THE TEST DATA CANNOT INFLUENCE THE TRAINING PHASE IN ANY WAY.

- We're measuring test error to see how well we do on new data:
  - If used during training, doesn't measure this.
  - You can start to overfit if you use it during training.

– You also shouldn't change the test set to get the result you want.

Golden Rule of Machine Learning

- Note the golden rule applies to hypothesis testing in scientific studies.
  - Data that you collect can't influence the hypotheses that you test.

- EXTREMELY COMMON and a MAJOR PROBLEM, coming in many forms:
  - Collect more data until you coincidentally get significance level you want.
  - Try different ways to measure performance, choose the one that looks best.
  - Choose a different type of model/hypothesis after looking at the test data.

- If you want to modify your hypotheses, you need to test on new data.
  - Or at least be aware and honest about this issue when reporting results.

Is Learning Possible?

- Does training error say anything about test error?
  - In general, NO: Test data might have nothing to do with training data.
  - E.g., "adversary" takes training data and flips all labels.

$X =$

Egg	Milk	Fish
0	0.7	0
0.3	0.7	1
0.3	0	0

$y =$

Sick?
1
1
0

$\tilde{X} =$

Egg	Milk	Fish
0	0.7	0
0.3	0.7	1
0.3	0	0

$\tilde{y} =$

Sick?
0
0
1

- In order to learn, we need assumptions:
  - The training and test data need to be related in some way.
  - Most common assumption: independent and identically distributed (IID).

IID Assumption

- Training/test data is independent and identically distributed (IID) if:
  - All examples come from the same distribution (identically distributed).
  - The example are sampled independently (order doesn't matter).

Age	Job?	City	Rating	Income
23	Yes	Van	A	22,000.00
23	Yes	Bur	BBB	21,000.00
22	No	Van	CC	0.00
25	Yes	Sur	AAA	57,000.00

Row 1 comes from same distribution as rows 2-3.

Row 4 does not depend on values in rows 1-3.

- Examples in terms of cards:
  - Pick a card, put it back in the deck, re-shuffle, repeat. → IID
  - Pick a card, put it back in the deck, repeat.
  - Pick a card, don't put it back, re-shuffle, repeat.

} Not IID

Learning Theory

- Why does the IID assumption make learning possible?
  - Patterns in training examples are likely to be the same in test examples.
- The IID assumption is rarely true:
  - But it is often a good approximation.
  - There are other possible assumptions.
- Also, we're assuming IID across examples but not across features.

- Learning theory explores how training error is related to test error.
- ~~We'll look at a simple example, using this notation:~~
  - ~~E_{train} is the error on training data.~~
  - ~~E_{test} is the error on testing data.~~

Learning Theory

HOW GOOD IS YOUR MODEL

- Given:
  - Distribution D over (x,y) pairs
  - A hypothesis $h \in H$ from hypothesis class H
  - Loss function L
- We are interested in Expected Risk:

$$R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}} L(h(x), y)$$

- Given training set S, and a particular hypothesis h, Empirical Risk:

$$\hat{R}(S,h) = \frac{1}{|S|} \sum_{(x,y)\in S} L(h(x), y)$$

Learning Theory

- Important remarks
  - The Expected Risk is an expectation of error
  - We want to minimize the Expected Risk but we can not as D is unknown
    - The test error is impossible to be
  - So, we will try to minimize the Empirical Risk (Empirical Risk Minimization (ERM)) on the set S:
    - S: the training set
    - If the error is measured on S: Training Error (not enough as model bias)
    - We need another set T: Testing set for computing the test set error (unbiased)
  - Other alternatives is the literature to ERM:
    - Heterogeneous Risk Minimization (HRM)

Example of ERM

- Fix hypothesis class (logistic regression)

$$h(x; \mathbf{w}, b) = \sigma(\mathbf{w}^T \phi(x) + b)$$

- Define loss function

$$L(h(x; \mathbf{w}, b), y) = -(y \log h(x; \mathbf{w}, b) + (1 - y) \log(1 - h(x; \mathbf{w}, b))$$

- Minimize average loss on the training set using SGD

$$\min_{\mathbf{w}, b} \frac{1}{N} \sum_{i=1}^{N} L(h(x_i, \mathbf{w}, b), y_i)$$

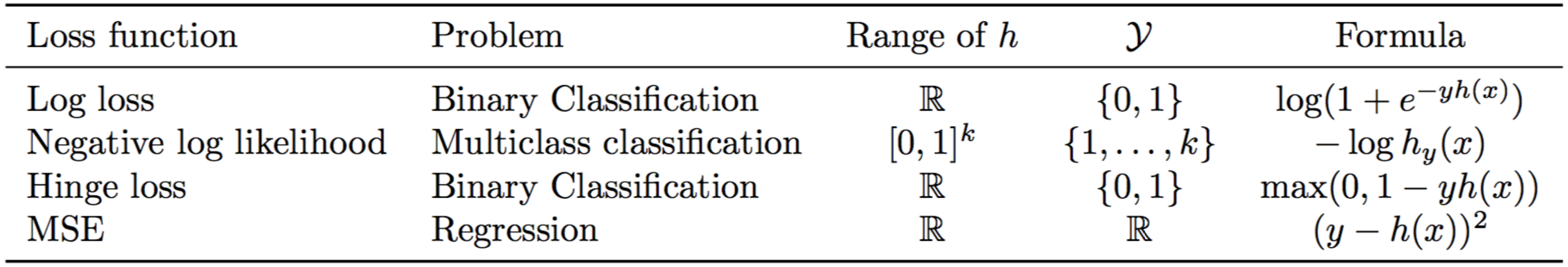
Objective: learn the best h() that minimize the loss (find the best w)

Example of loss function

- Cost (loss or objective) function:
  - quantifies the error between predicted values and expected values and **presents it in the form of a single real number**.

- Various measures of model error (any norm) depending on the hypothesis space
  - Mean Absolute Error (MAE) $$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}|$$
  - Mean Square Error (MSE) $$MSE = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y})^2$$
  - Root Mean Square Error (RMSE) $$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y})^2}$$

Example of loss function



Loss function	Problem	Range of h	$\mathcal{Y}$	Formula
Log loss	Binary Classification	$\mathbb{R}$	$\{0,1\}$	$\log(1+e^{-yh(x)})$
Negative log likelihood	Multiclass classification	$[0,1]^k$	$\{1,\ldots,k\}$	$-\log h_y(x)$
Hinge loss	Binary Classification	$\mathbb{R}$	$\{0,1\}$	$\max(0,1-yh(x))$
MSE	Regression	$\mathbb{R}$	$\mathbb{R}$	$(y-h(x))^2$

Empirical Risk

$$R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}} L(h(x), y) \qquad \hat{R}(S,h) = \frac{1}{|S|} \sum_{(x,y)\in S} L(h(x), y)$$

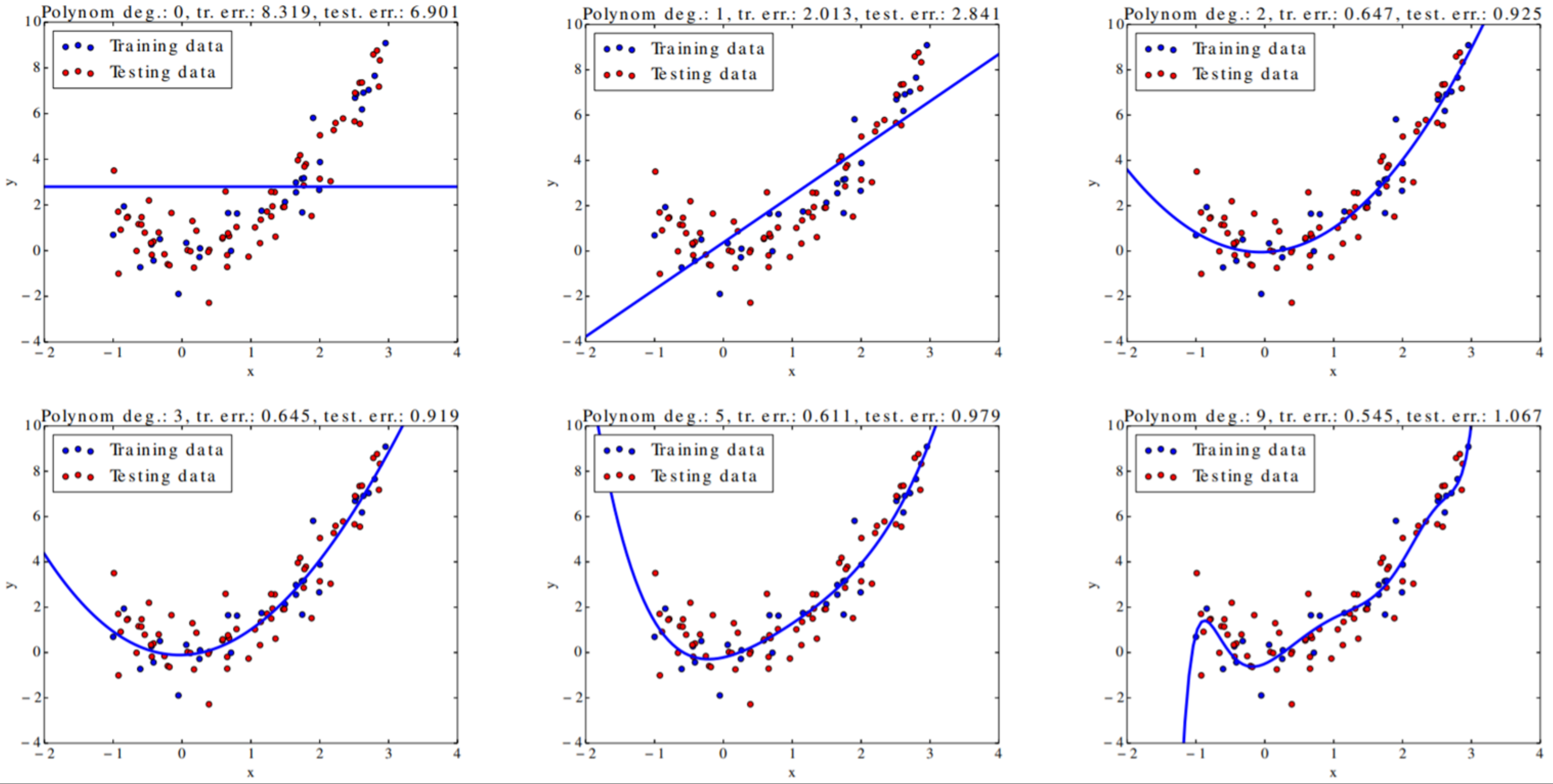
$$R(h) = \underbrace{\hat{R}(S,h)}_{\text{Training error}} + \underbrace{\left(R(h) - \hat{R}(S,h)\right)}_{\text{Generalization error}}$$

Training and testing error

- Example: Polynomial regression with varying degree

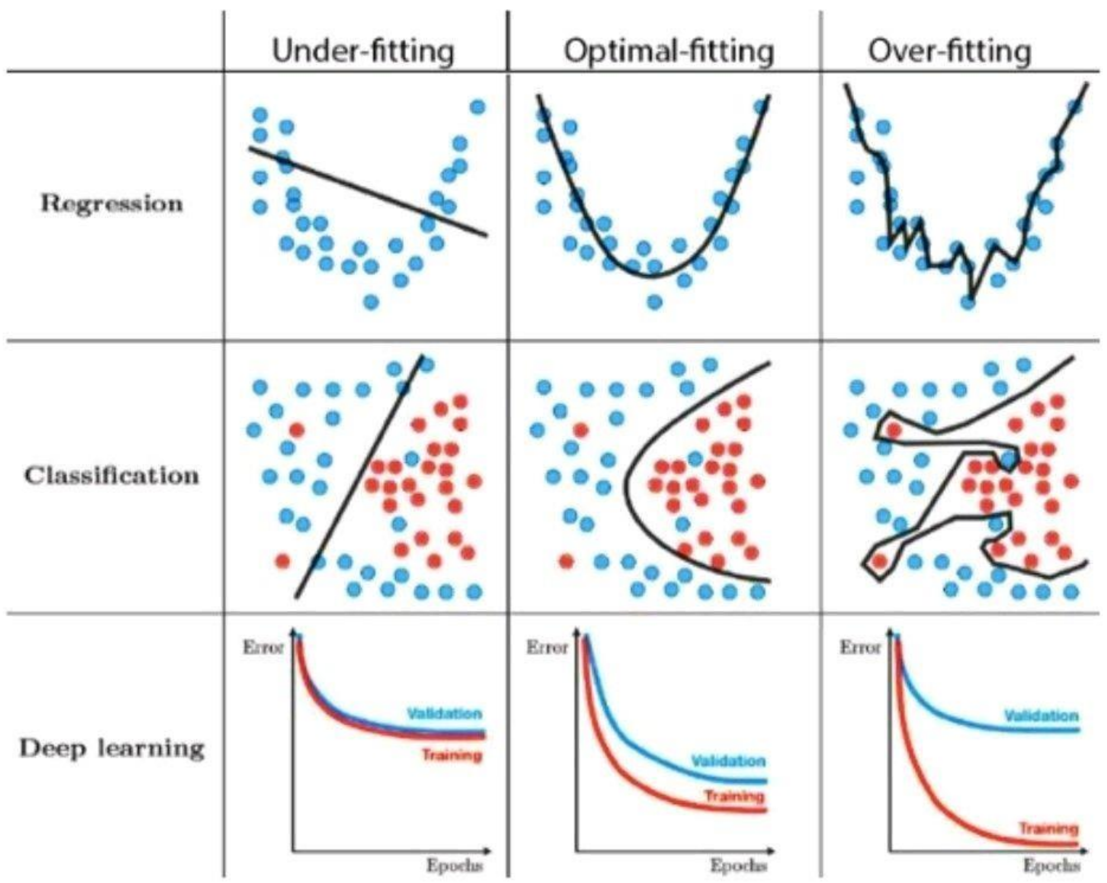
$$X \sim U(-1,3)$$

$$Y \sim X^2 + N(0,1)$$



Training and testing error

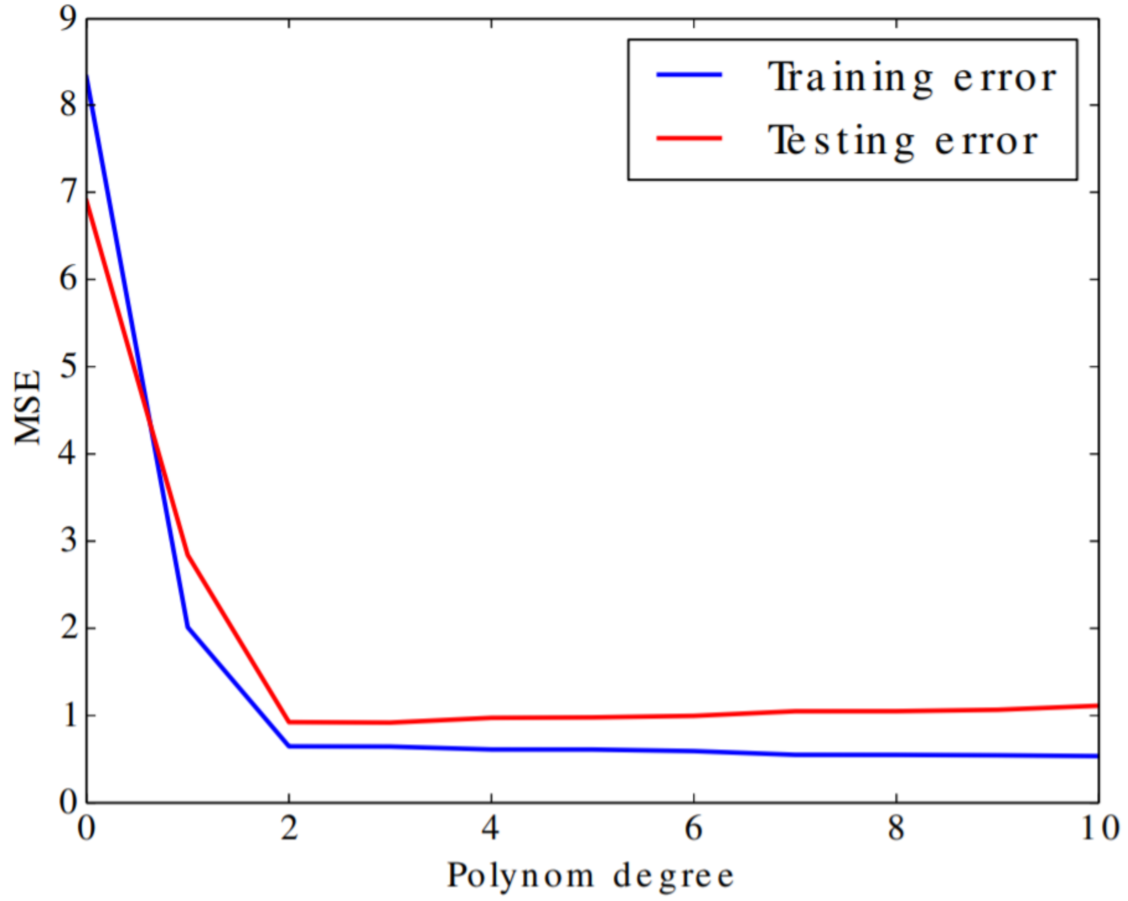
- Example: linear regression



Source: Underfitting, Optimal-fitting and Overfitting for linear regression [1]

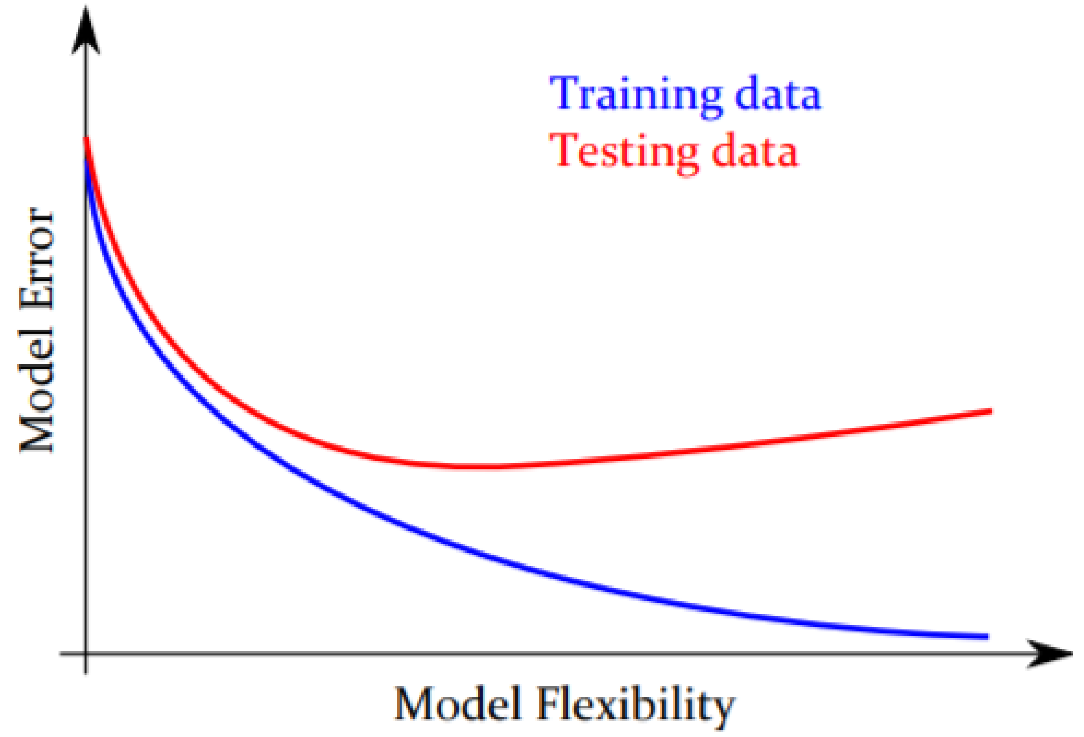
Training and testing error

- The training error decreases with increasing model flexibility (the curviness).
- The testing error is minimal for certain degree of model flexibility.



Overfitting

- Overfitting is a general phenomenon affecting all kinds of inductive learning.
- When overfitted, the model works well for the training data, but fails for new (testing) data.



Bias vs Variance

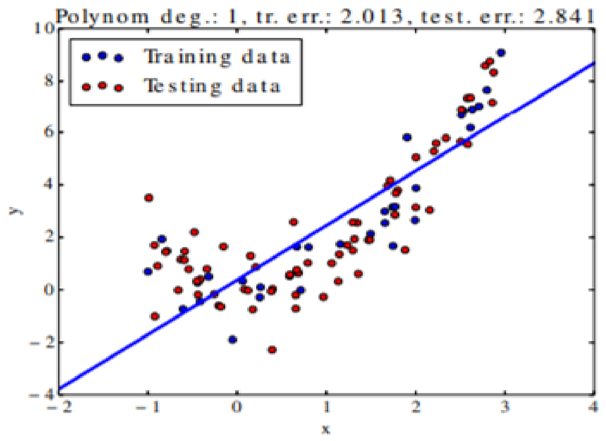
- The bias error is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

$$\mathrm{Bias} = E[\hat{\theta}] - \theta$$

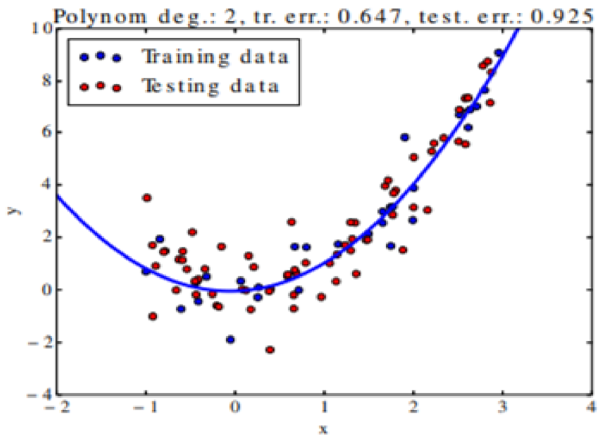
- The variance is an error from sensitivity to small fluctuations in the training set. High variance may result from an algorithm modeling the random noise in the training data (overfitting).

$$\mathrm{Var}(\hat{\theta}) = E[\hat{\theta}^2] - \left(E[\hat{\theta}]\right)^2 \qquad \mathrm{Var}(\hat{\theta}) = E[(E[\hat{\theta}] - \hat{\theta})^2]$$

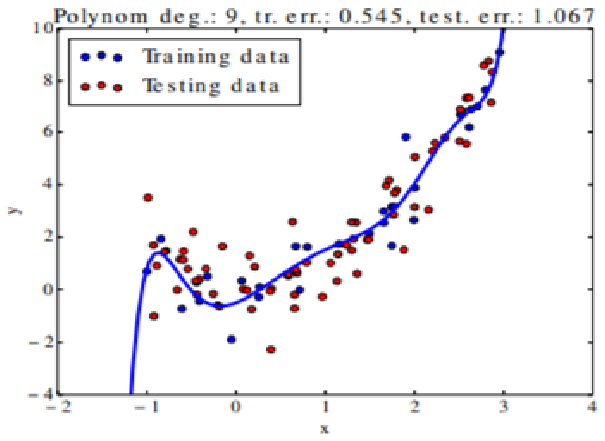
Bias vs Variance



High bias:
model not flexible enough
(Underfit)



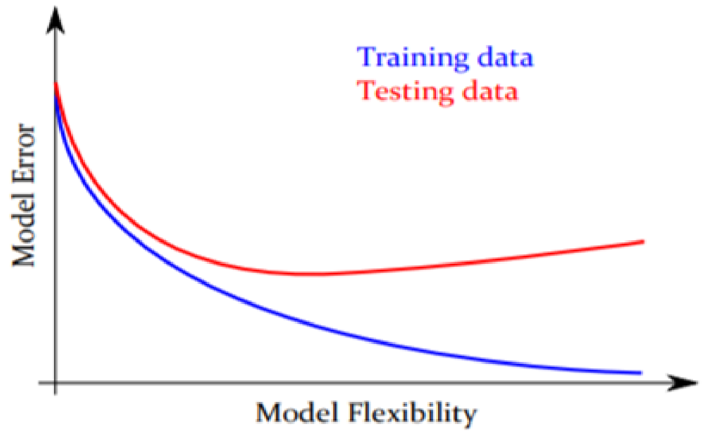
"Just right"
(Good fit)



High variance:
model flexibility too high
(Overfit)

High bias problem:

- Err_{Tr} is high
- $Err_{Tst} \approx Err_{Tr}$



High variance problem:

- Err_{Tr} is low
- $Err_{Tst} >> Err_{Tr}$

Bias – Variance Tradeoff

- Simple example

$$\begin{aligned} MSE(\hat\beta) &= \mathbb{E}((\hat\beta-\beta)^T(\hat\beta-\beta)) \\ &= \mathbb{E}(\hat\beta-\beta)\mathbb{E}((\hat\beta-\beta)^T) + \mathrm{Var}(\hat\beta) \\ &= ||\mathrm{Bias}(\hat\beta)||^2 + \mathrm{Var}(\hat\beta) \end{aligned}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

Suppose that

$$Y \sim N(\mu, \sigma^2)$$

and the estimator $\hat\mu = \alpha Y$ where $0 \le \alpha \le 1$.

The bias is $\mathbb{E}(\hat\mu) - \mu = (\alpha - 1)\mu$. The variance is $\sigma^2\alpha^2$.

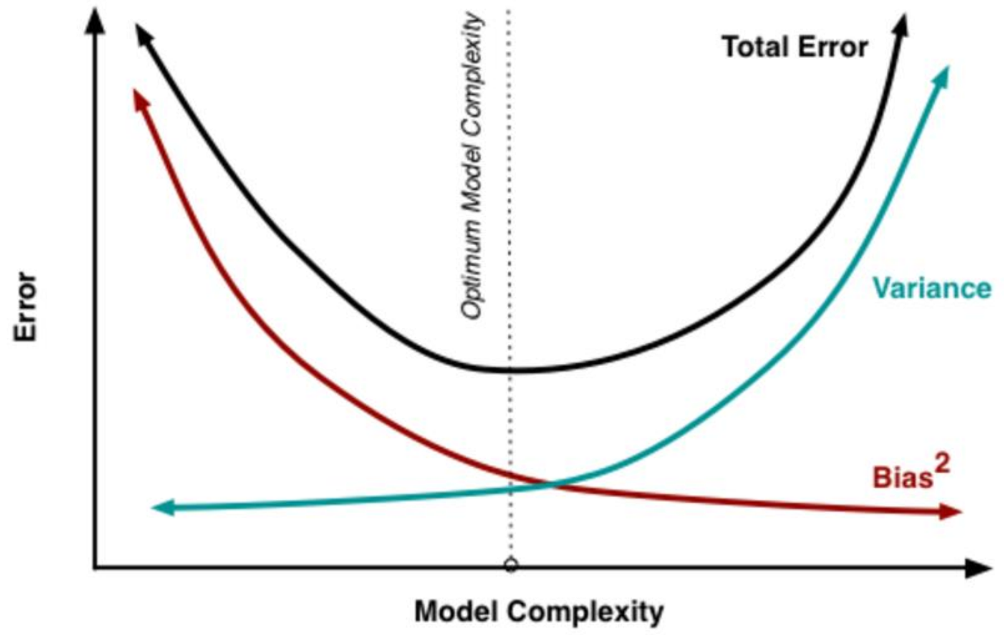
bias^2 + *variance* $=(\alpha-1)^2\mu^2 + \sigma^2\alpha^2$

For $\alpha \to 0$, the bias increases and the variance decreases.

The optimal is $\alpha^* = \mu^2/(\sigma^2+\mu^2)$

Regularization

- Who to deal with overfitting ?

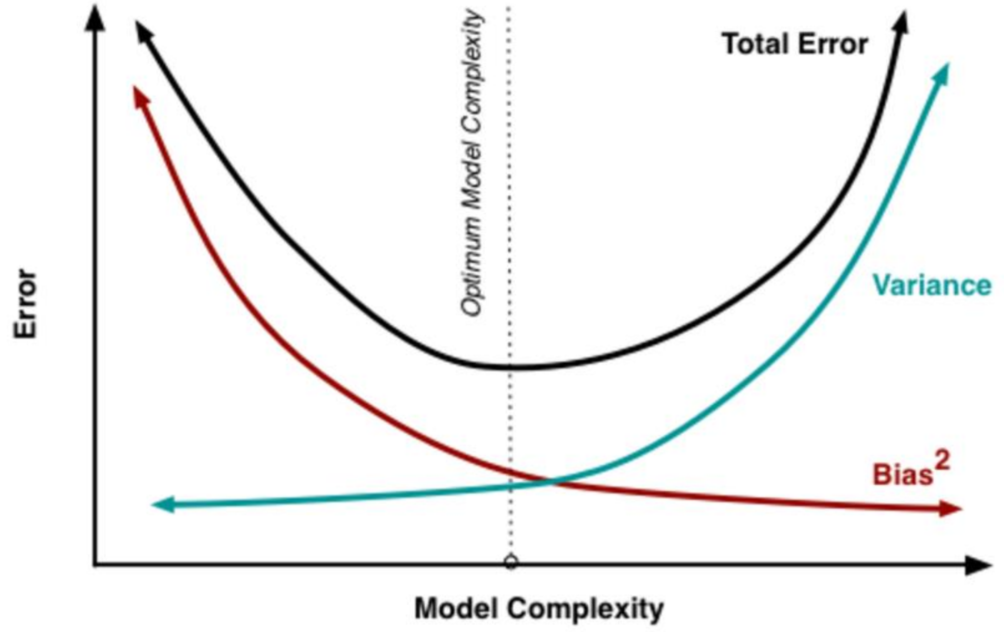


Regularization

- Who to deal with overfitting ?

Simple ways to avoid overfitting :

- Make your model simpler
- Regularization and/or Use Dropouts (NN)
- Train your model on more data
- Augment your dataset
- Stop the learning process
- Standardize input data

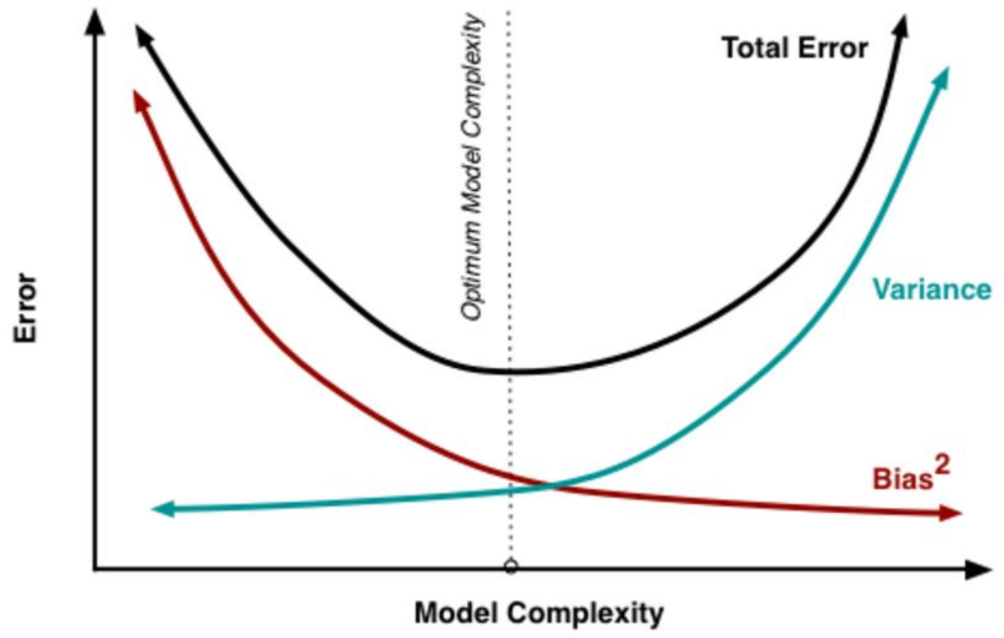


Regularization

- Reduce the model complexity
    - Reduce the width of the tree, size, …
    - Reduce the number of parameters (number of neurons, layers, …)
    - Reduce the values of parameters (small values, zeros, …)

- In parameterized model, a regularization technique is a penalty mechanism which applies shrinkage (driving them closer to zero) of coefficient to build a more robust and parsimonious model.

- Reduce model variance at the cost of introducing some bias.

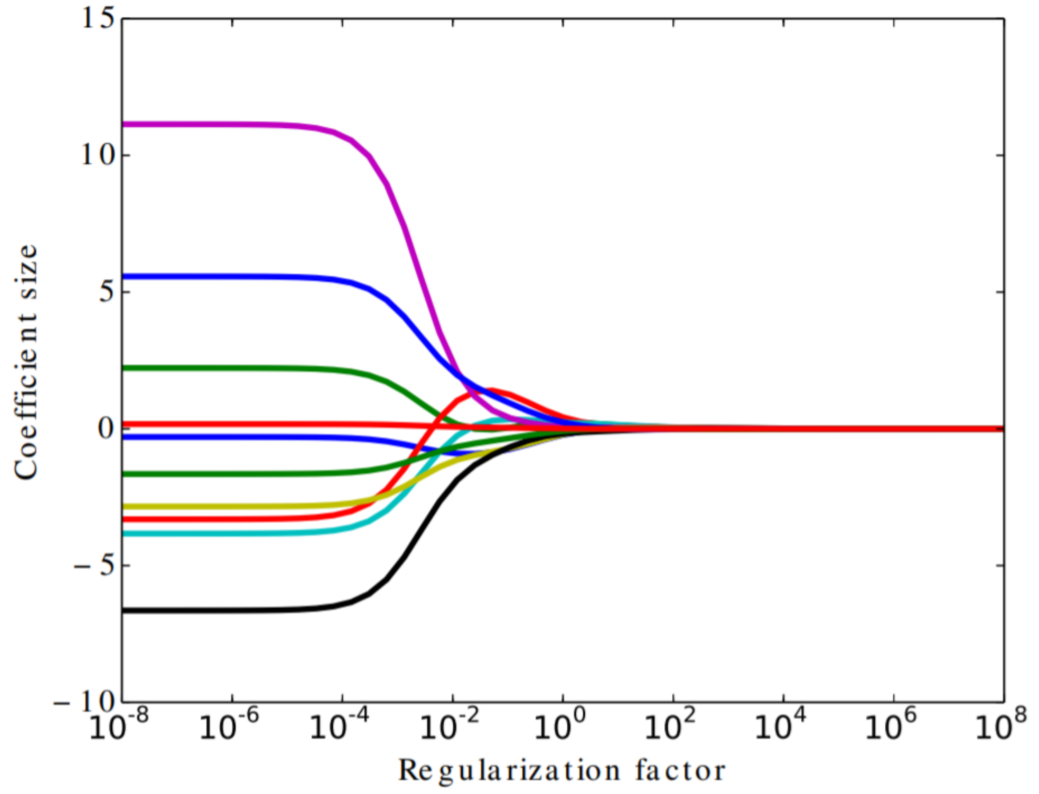


Ridge regularization (L2 regularization)

- Ridge regularization imposes a penalty on the size of the model coefficients.
- The penalty is equal to the sum of the squared value of the coefficients w_i

$$Error_{Ridge} = Error + \alpha \sum_{i=1}^{d} {w_i}^2$$

- α is the regularization parameter
- Ridge forces **the parameters to be relatively small**
- The bigger the penalization, the smaller (and the more robust) the coefficients are.



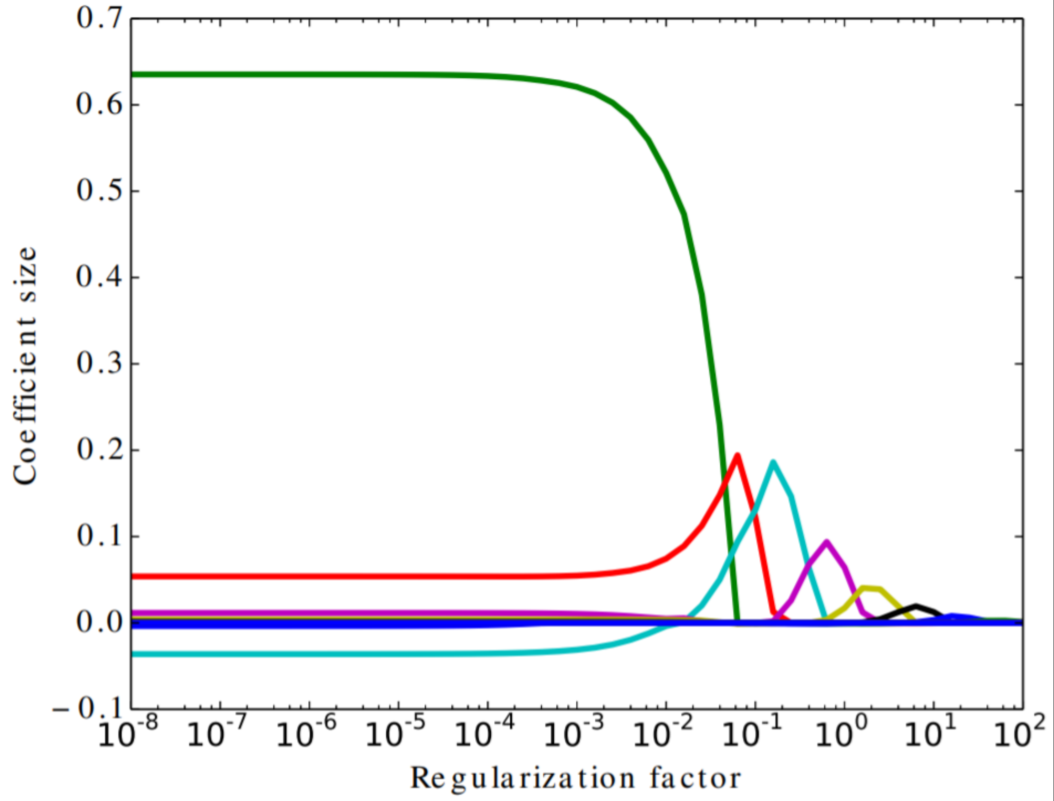
- $\alpha \rightarrow 0, \quad w^{ridge} \rightarrow w^{Error}$
- $\alpha \rightarrow \infty, \quad w^{ridge} \rightarrow 0$

Lasso regularization (L1 regularization)

- Like ridge, Lasso regularization penalizes the size of the model coefficients.
- The penalty is equal to the sum of the absolute value of the coefficients w_i

$$Error_{Lasso} = Error + \alpha \sum_{i=1}^{d} |w_i|$$

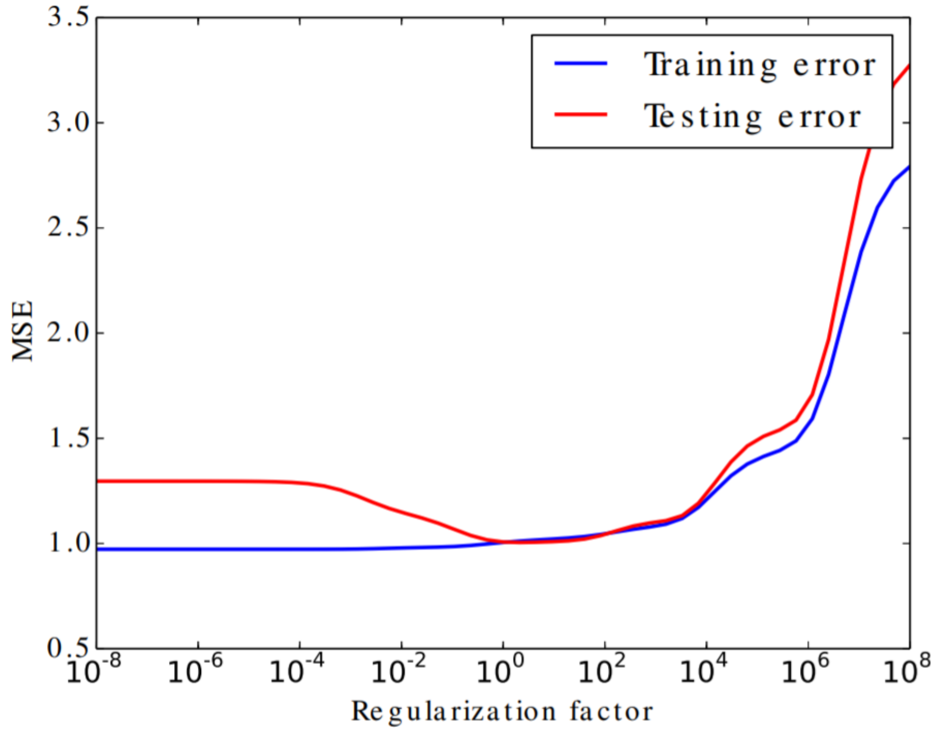
- Lasso regularization will **shrink some parameters to zero**.
- It can be seen as a way to select features in a model.



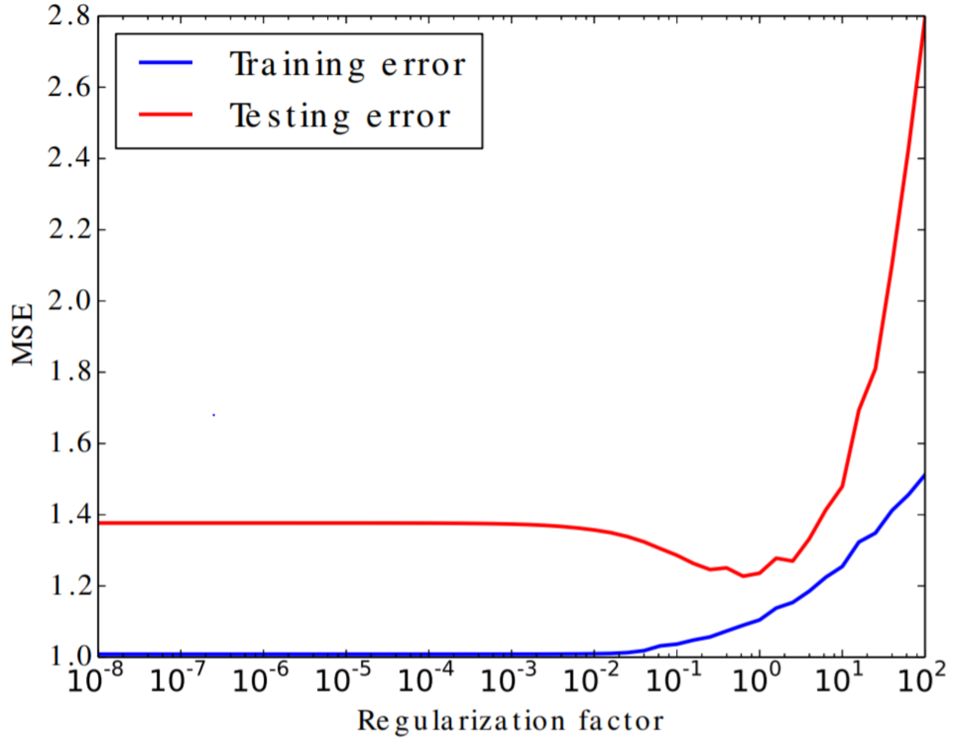
- $\alpha \to 0, \quad w^{Lasso} \to w^{Error}$
- $\alpha \to \infty, \quad w^{Lasso} = 0$

Ridge vs Lasso

- Evolution of error vs regularization factor



Ridge (L2 regularization)



Lasso (L1 regularization)

When the LASSO fails?

- In the $d > n$ case, the lasso can select at most n variables before it saturates (d: number of paramters and n: number of samples)
- If there is a group of variables with very high pairwise correlations, the lasso tends to select only one variable from the group, not caring which one
- For usual $n > d$ situations, if there are high correlations between variables, the prediction performance of lasso is poor with respect to ridge.

Elastic-net

- The lasso sometimes does not perform well with highly correlated variables, and often performs worse than ridge in prediction
- Elastic-net is a mix of **both Ridge and Lasso regularizations**.

$$Error_{Elastic} = Error + \alpha \left(\rho \sum_{i=1}^{d} |w_i| + \left(\frac{1-\rho}{2}\right) \sum_{i=1}^{d} {w_i}^2\right)$$

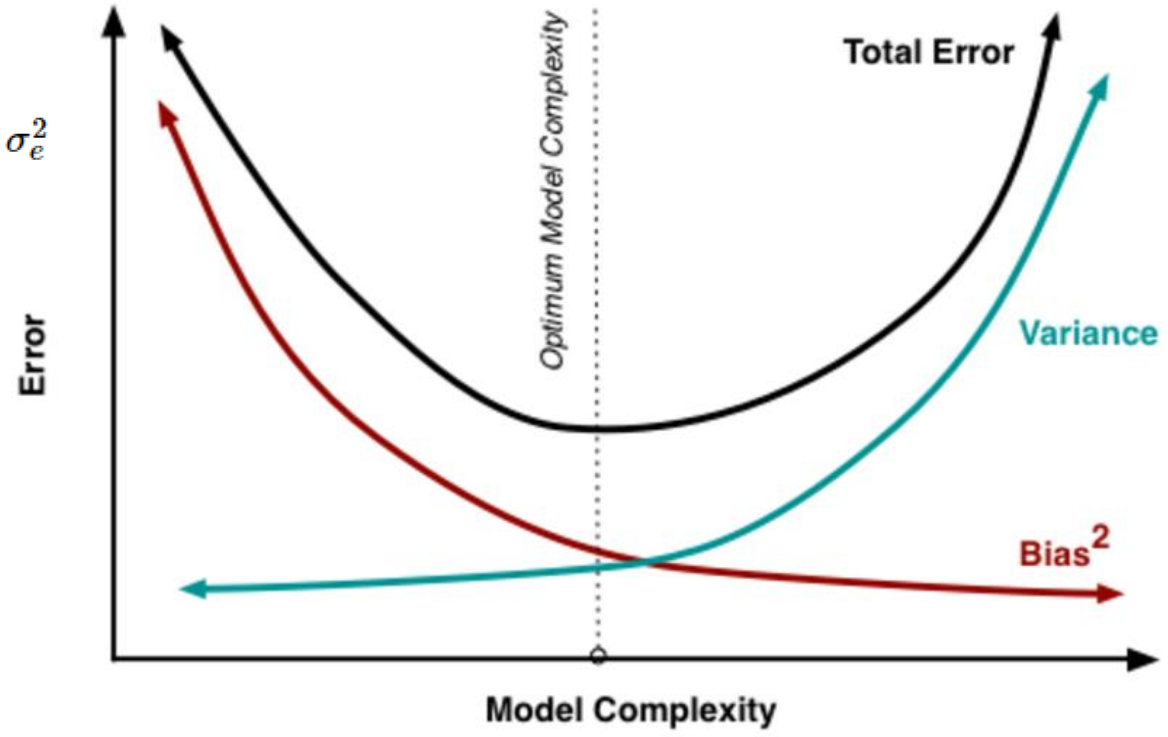
- α is a shared penalization parameter
- ρ is a mixing parameter, it sets the ration between ridge (L2) and lasso (L1) regularization
  - $\rho = 0$: Ridge regularization
  - $\rho = 1$: Lasso regularization

Bias-Variance Tradeoff

- To get good predictions, we need to find a balance of Bias and Variance that minimizes total error

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{Irreducible\ Error}$$



Backup slides

Proof of No Free Lunch Theorem

- Let's show the "no free lunch" theorem in a simple setting:
  - The x^i and y^i are binary, and y^i being a deterministic function of x^i.
- With 'd' features, each "learning problem" is a map from each of the 2^d feature combinations to 0 or 1: $\{0,1\}^d \rightarrow \{0,1\}$

Feature 1	Feature 2	Feature 3
0	0	0
0	0	1
0	1	0
…	…	…

Map 1	Map 2	Map 3	…
0	1	0	…
0	0	1	…
0	0	0	…
…	…	…	…

- Let's pick one of these maps ("learning problems") and:
  - Generate a set training set of 'n' IID samples.
  - Fit model A (convolutional neural network) and model B (naïve Bayes).

Proof of No Free Lunch Theorem

- Define the "unseen" examples as the $(2^d - n)$ not seen in training.
  - Assuming no repetitions of x^i values, and $n < 2^d$.
  - Generalization error is the average error on these "unseen" examples.
- Suppose that model A got 1% error and model B got 60% error.
  - We want to show model B beats model A on another "learning problem".

- Among our set of "learning problems" find the one where:
  - The labels y^i agree on all training examples.
  - The labels y_i disagree on all "unseen" examples.
- On this other "learning problem":
  - Model A gets 99% error and model B gets 40% error.

Proof of No Free Lunch Theorem

- Further, across all "learning problems" with these 'n' examples:
  - Average generalization error of **every** model is 50% on unseen examples.
    - It's right on each unseen example in exactly half the learning problems.
- With 'k' classes, the average error is (k-1)/k (random guessing).

- This is kind of depressing:
  - For general problems, no "machine learning" is better than "predict 0".

- But the proof also reveals the problem with the NFL theorem:
  - Assumes every "learning problem" is equally likely.
  - World encourages patterns like "similar features implies similar labels".