Mathematical Foundation of Machine Learning

I: Introduction and error analysis

Outline

Part I: Deep Learning

ref. book: Ian GoodFellow, Yoshua Benjio, Aaron Conrville – Deep Learning (https://www.deeplearningbook.org/)

- Machine Learning Basics, Error Analysis

  - basic models, model performance evaluation

- Optimization in Deep Learning

  - DL models, structure from dynamical system point of view, non-convex optimization

- Deep Generative Modeling and Inference

  - VAE, Normalizing Flow, GAN, Structured latent variables, self-supervised

• Part II: Reinforcement Learning

ref. Book:

- *Richard S. Sutton, Andrew G. Barto: Refincement Learning:An introduction*
- *Dimitri P. Bertsekas, reinforcement learning and optimal control*

• Introduction and comparison with optimal control

- Chapter 1 - 3: Introduction and Markov Decision Process

• Value Based RL and Policy based RL

- Chapter 4 - 6: Dynamical Programing, Monte Carlo and TD Learning
- Chapter 9 - 11: On-policy, Off-policy, Actor-Critic

• Frontiers of RL and Applications

- Connection between optimal control and RL:
- constrained hidden states
- Multi-Agent Deep Reinforcement Learning

• Part III: Research directions

- Learning stochastic dynamical systems from data
- Missing data reconstruction and prediction with applications in NLP, CV, math biology ect.
- Learning dynamics：invariant manifolds, bifurcation, chaos
- Understand dynamics of neural networks
- Nonlocal, Anomaly diffusion, numerial algorithms

You are more than welcome to present !

I. Machine Learning basics, Error Analysis

Outline

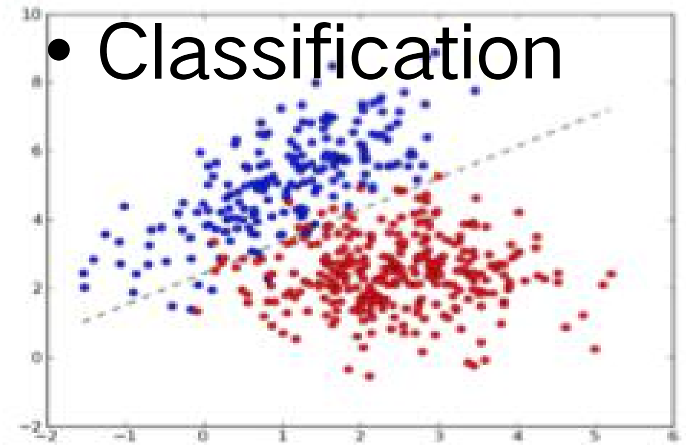
- Machine Learning Basics
  - tasks/problems
  - models
  - algorithms
- Model Evaluation
- Research: quantifying generalization error in deep learning
  - training data size
  - model compacity
  - smoothness of Neural Network

Outline

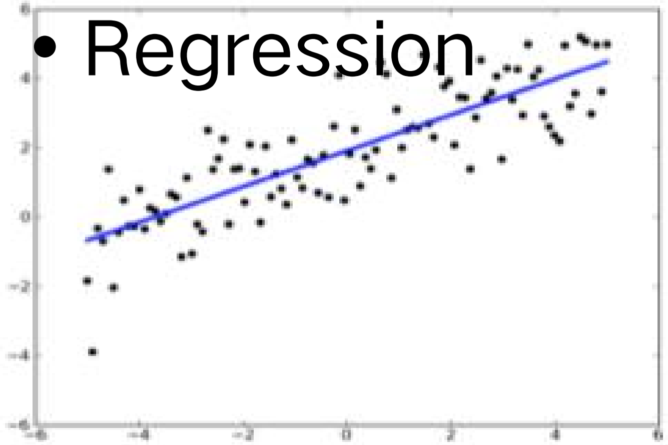
- Machine Learning Basics
  - tasks/problems
  - models
  - algorithms
- Model Evaluation
- Research:
  - trade off of large scale learning
  - quantifying generalization error in deep learning

- Machine Learning Basics: Tasks

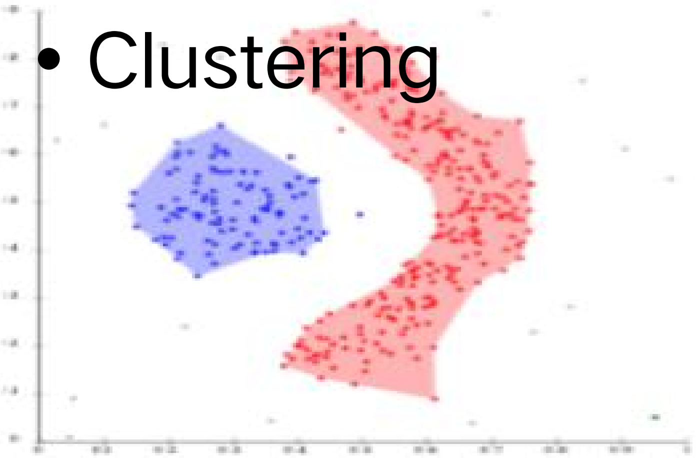
- Classification



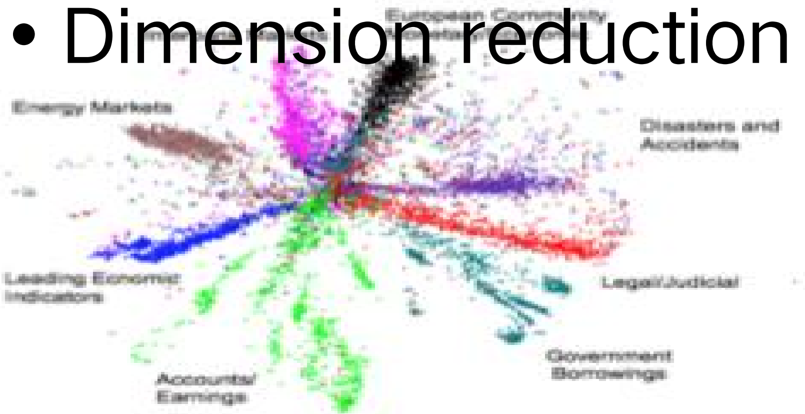
- Regression



- Clustering



- Dimension reduction



- # Machine Learning Basics: Models

Tasks	Models	
Classification	Logistic Regression, SVM, KNN,	Decision Tree, Random Forest, Adaboost, Gradient Boosting, Neural Network
Regression	Linear, Polynomial,	
Clustering	K-means, Hierachy, Density based, Neural Network	
Dimension reduction	SVD, PCA, LDA, Neural Network	

ML challenges in real applications (to my understanding)

- ➢ Big Data: high dimension, sparsity
- ➢ Data Distribution shift over time, Or discrepency btw training vs. predicting;
- ➢ Catestrophic forgeting and model generalization

- Machine Learning Basics: Model

Take Supervised Learning for Example:

- Linear regression
- Logistic regression
- Nueral Network

Linear Regression

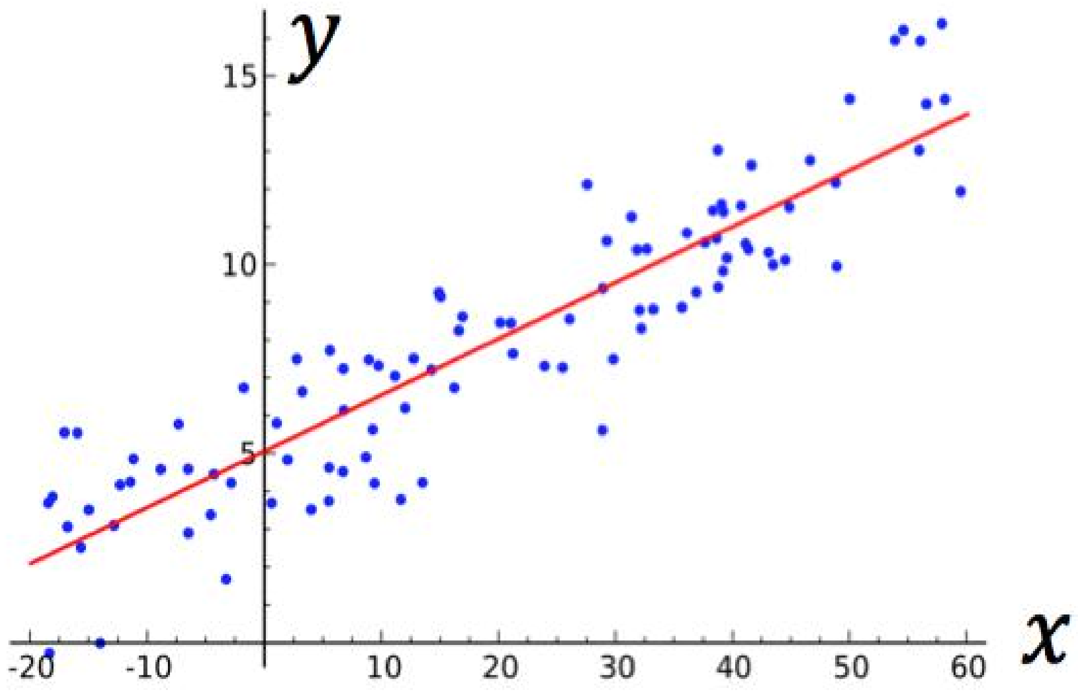
Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$

Output: a vector $\mathbf{w} \in \mathbb{R}^d$ and scalar $b \in \mathbb{R}$ such that $\mathbf{x}_i^T \mathbf{w} + b \approx y_i$.

1-dim ($d = 1$) example:

Solution:

$$y_i \approx 0.15\, x_i + 5.0$$



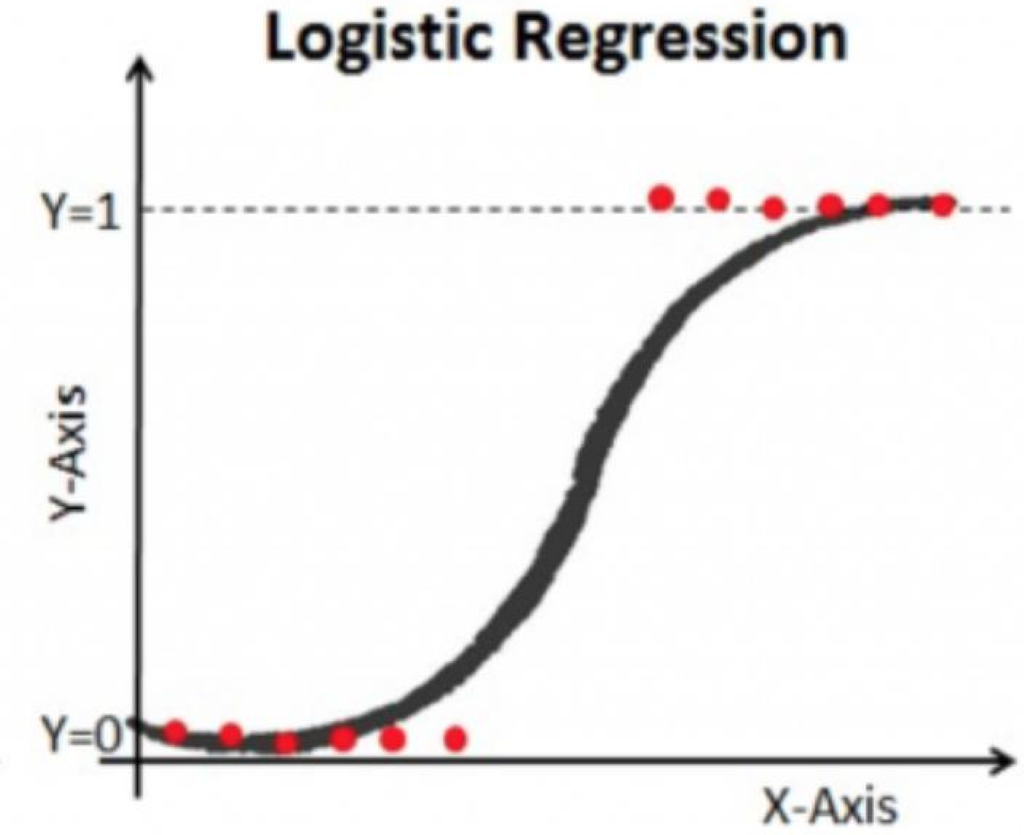
Logistic regression

Binary classification: 1-dim case

Solve: a, b = ?

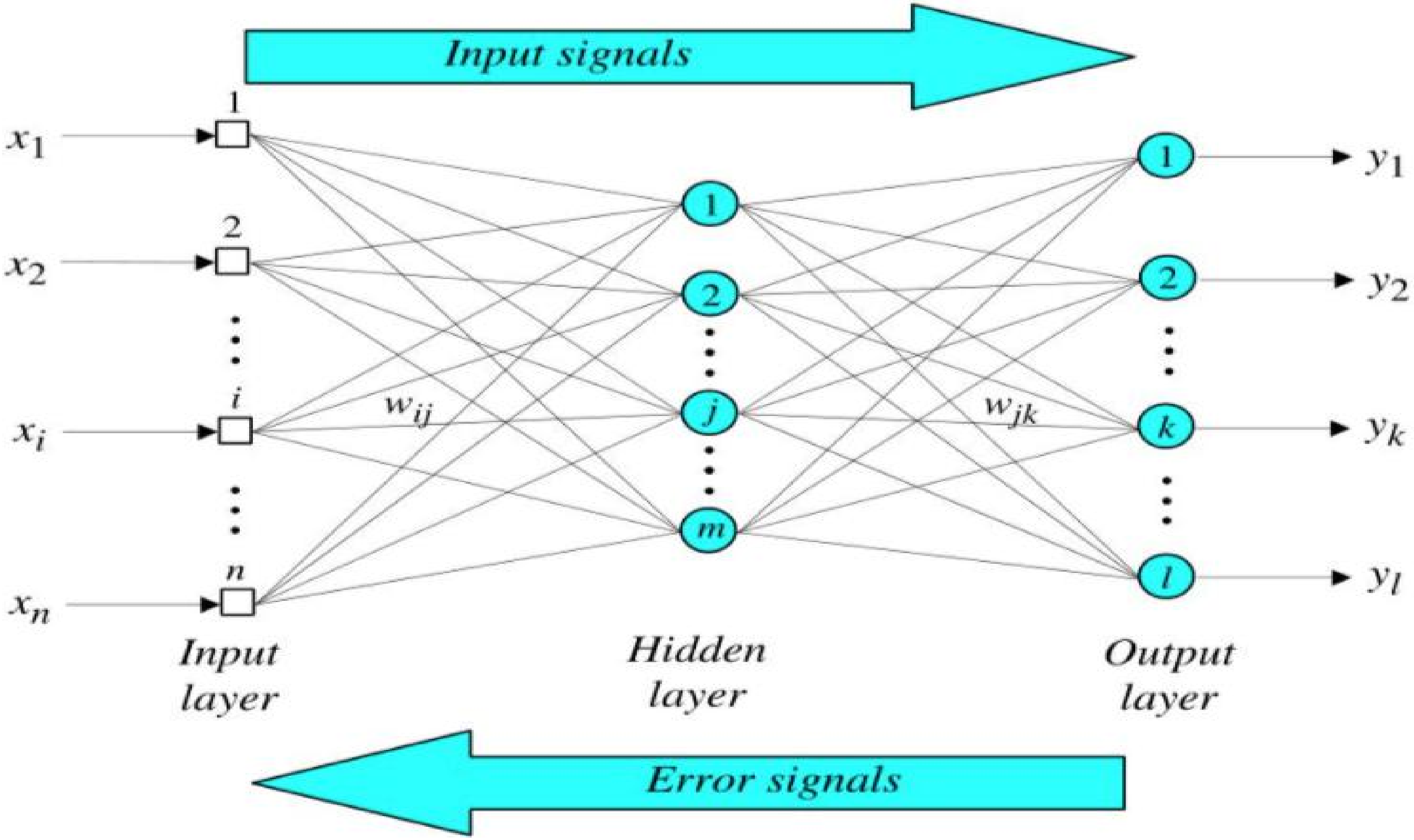
$$y = \frac{1}{1 + e^{-(a.x+b)}}$$

Sigmoid ?

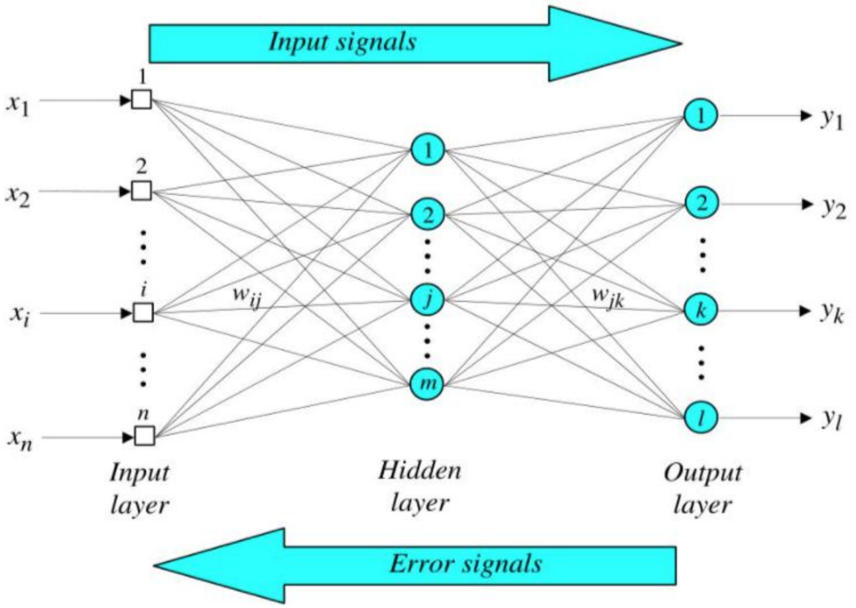


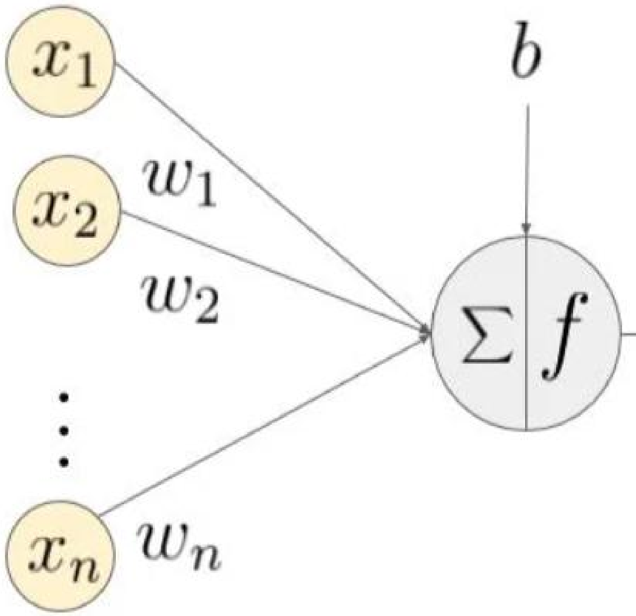
Neural Network

solve: weights **w**?



Neural Network

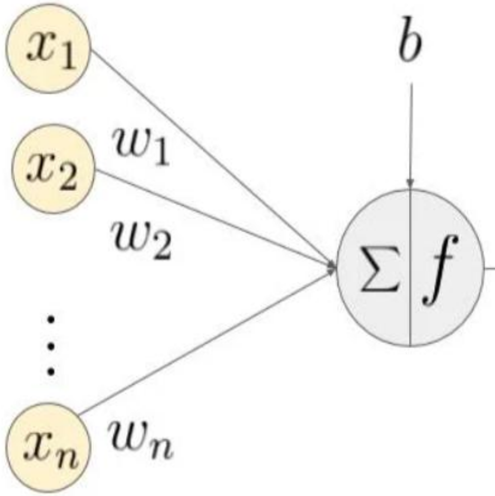




$$f\left(b + \sum_{i=1}^{n} x_i w_i\right)$$

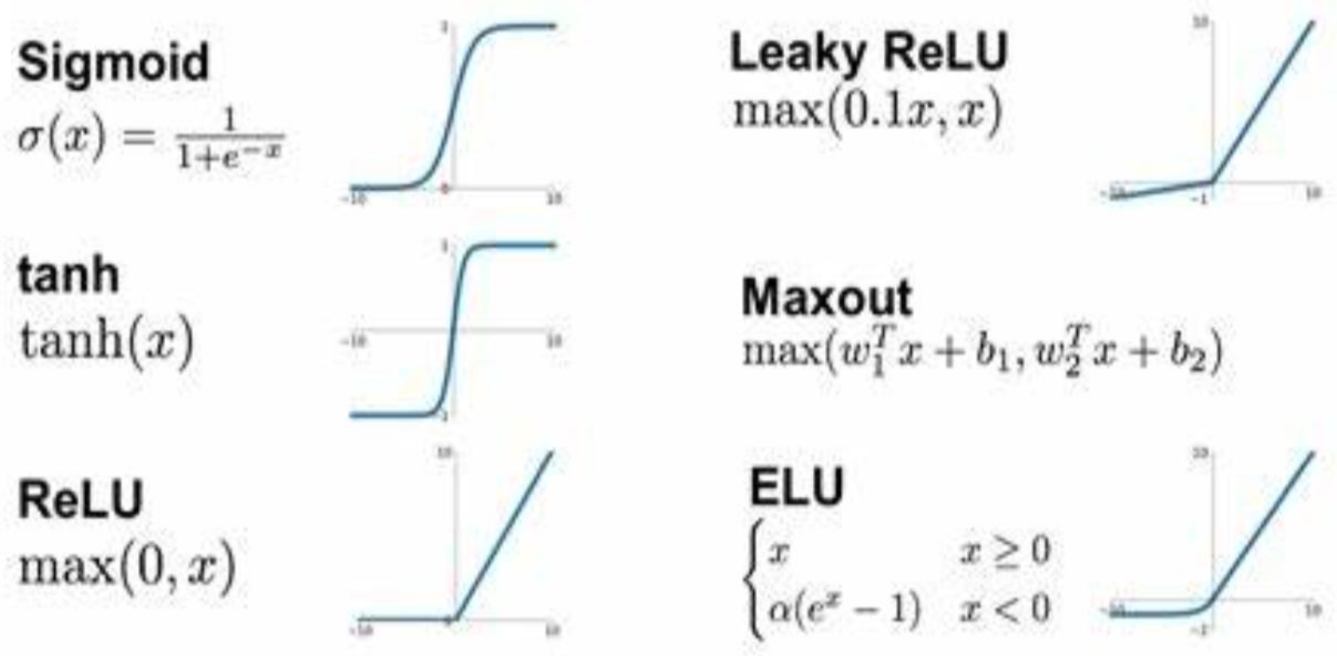
f: activation function

Neural Network



$$f\left(b + \sum_{i=1}^{n} x_i w_i\right)$$

f: activation function

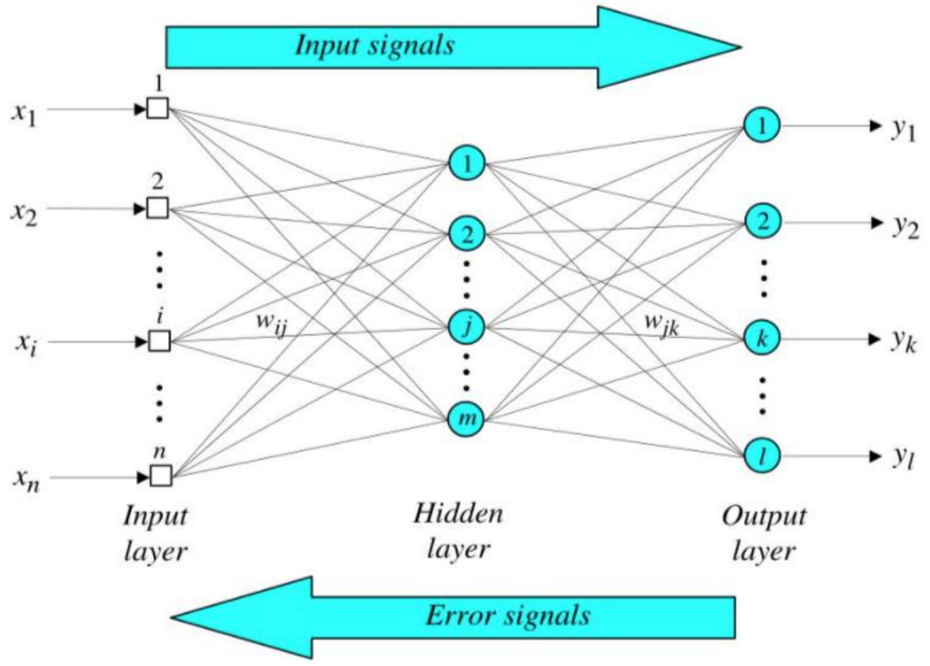


Neural Network

$$\sum_j a_j \sigma(w_j^T x + b_j) : w_j \in \mathbb{R}^d, a_j, b_j \in \mathbb{R}$$

σ is the activation function,

solve: weights **w**?



Summary:

Task

Data: (X, y)

Goal: find y = **f***(x)

Model

define objective function: *f(x, w)* *where w are unknown parameters.*

Algorithm

Define loss function: $L(f(x, w), y)$

Optimization: $f^*(x) = \arg\min_w [L(f(x, w), y)]$

- Machine Learning Basics: Algorithms

Assume $w \mapsto \ell(f_w(x), y)$ is convex and has a single minimum; the Hessian matrix H and the gradient covariance matrix G, both measured at the empirical optimum.

$$
\begin{aligned}
H &= \frac{\partial^2 C}{\partial w^2}(w_n) = \mathbb{E}_n\left[\frac{\partial^2 \ell(f_{w_n}(x), y)}{\partial w^2}\right], \\
G &= \mathbb{E}_n\left[\left(\frac{\partial \ell(f_{w_n}(x), y)}{\partial w}\right)\left(\frac{\partial \ell(f_{w_n}(x), y)}{\partial w}\right)'\right].
\end{aligned}
$$

- Machine Learning Basics: Algorithms

  - **Gradient Descent (GD)** iterates

$$w(t+1) = w(t) - \eta \frac{\partial C}{\partial w}(w(t)) = w(t) - \eta \frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial w} \ell(f_{w(t)}(x_i), y_i)$$

  - **Second Order Gradient Descent (2GD)** iterates

$$w(t+1) = w(t) - H^{-1} \frac{\partial C}{\partial w}(w(t)) = w(t) - \frac{1}{n} H^{-1} \sum_{i=1}^{n} \frac{\partial}{\partial w} \ell(f_{w(t)}(x_i), y_i)$$

✓ Stochastic Gradient Descent

✓ Batch training

Recall:

Task

Data: (X, y)

Goal: find y = **f***(x)

Model

define objective function: *f(x, w)* *where w are unknown parameters.*

Algorithm (next class ...)

Define loss function: $L(\, f(x, w), y \,)$

Optimization: $f^*(x) = \underset{w}{\mathrm{argmin}}[\, L(\, f(x, w), y \,) \,]$

Outline

- Machine Learning Basics
  - tasks/problems
  - models
  - algorithms
- Model Evaluation
- Research:
  - trade off of large scale learning
  - quantifying generalization error in deep learning

- Model Evaluation
  - regression: mean squared error
  - classification:

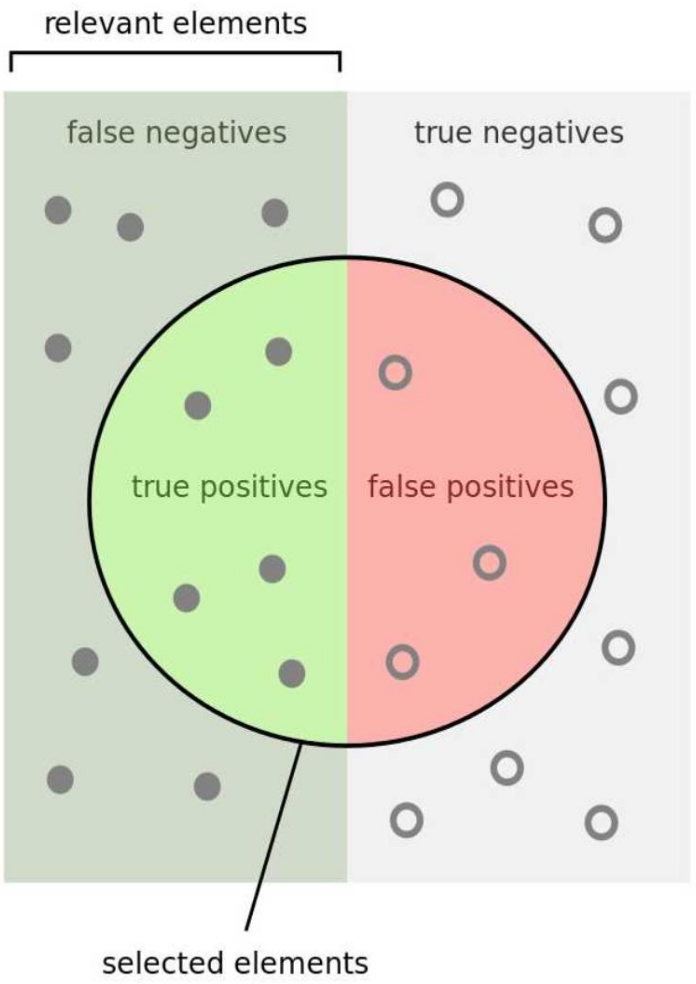
    accuracy

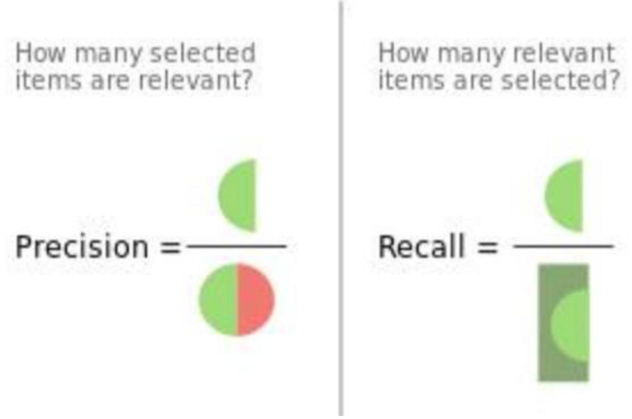
    error rate

    precision

    Recall

    F-score





- classification:

accuracy = TP+TN)/(TP+TN+FP+FN)

error rate = 1- accuracy

precision = TP/(TP+FP)

recall = TP/(TP+FN)

$$F - Score = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$$

混淆矩阵		真实值	
		Positive	Negative
预测值	Positive	TP	FP (Type II)
	Negative	FN (Type I)	TN

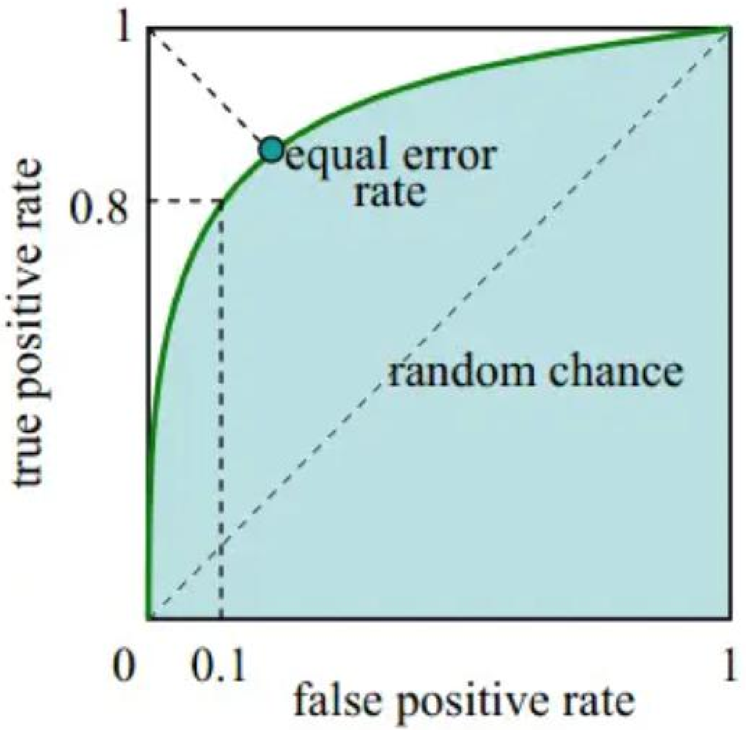
For unbalanced data, different use cases:

e.g. abnormal detection in finance, caner detection - improve recall

search engine - improve precision

- classification:

ROC, AUC:



(a)

Bias-variance decomposition:

Error: model compacity(algorithm), data size, task difficulty

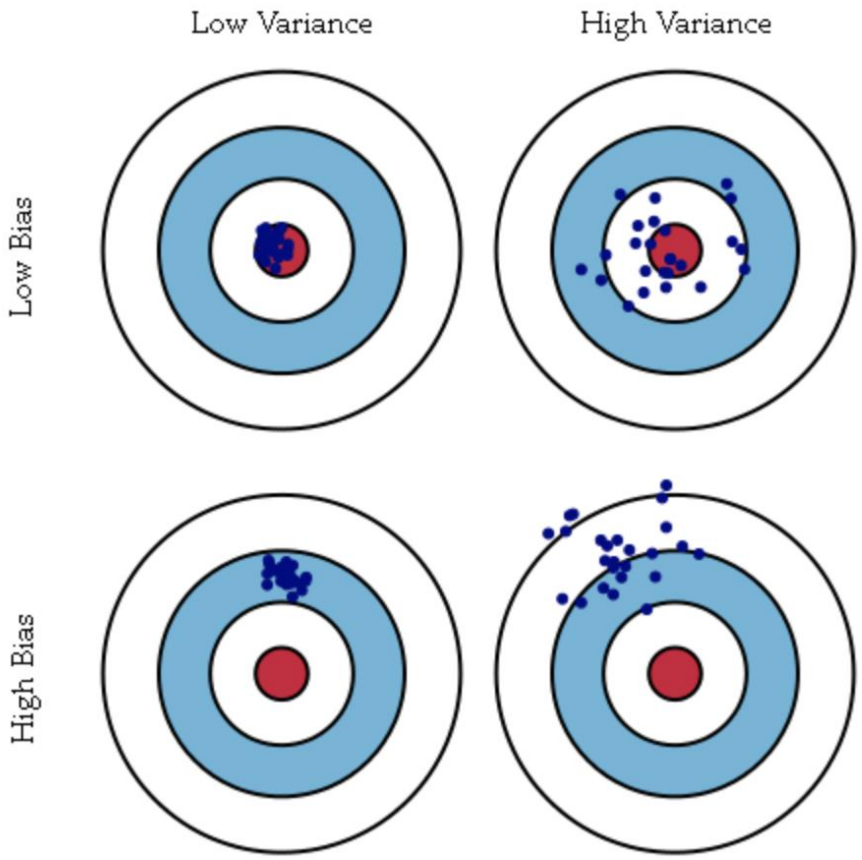
Machine Learning by Zhihua Zhou, ref. Friedman 2001

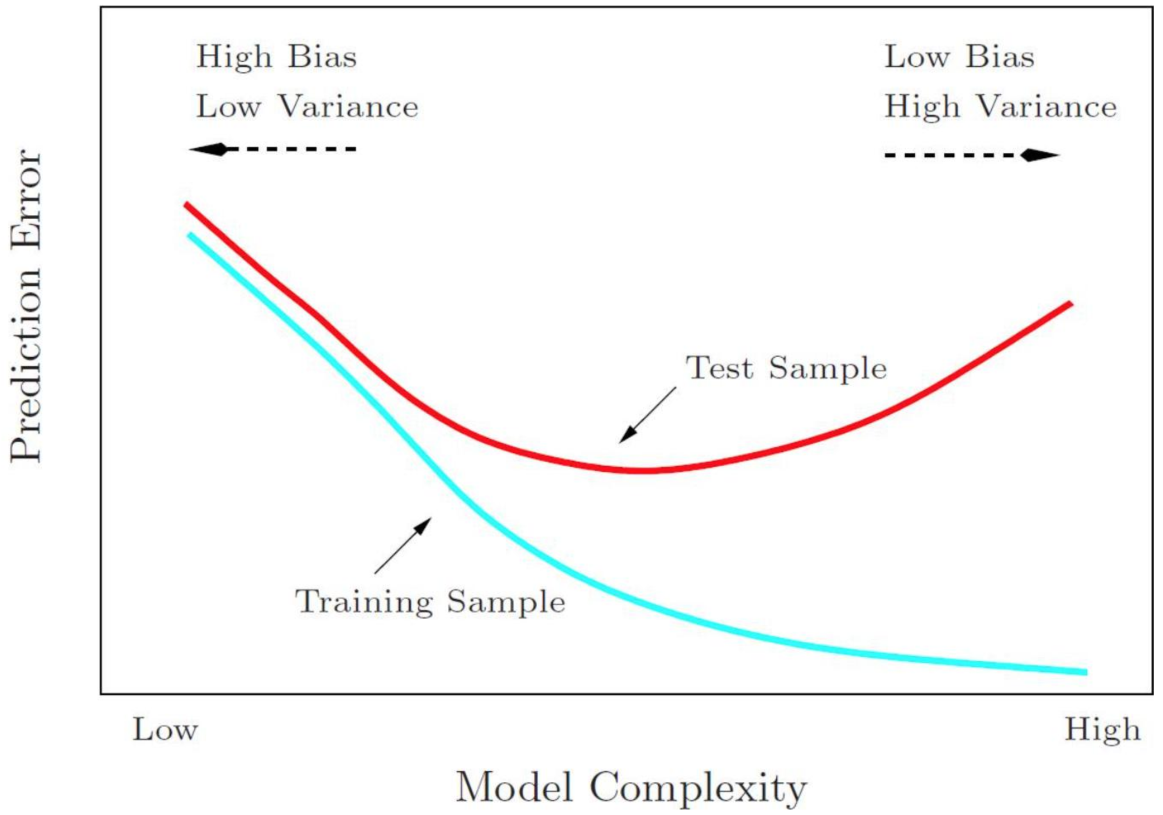
Suppose f(x; D) is the prediction result of x on training datas $\bar{f}(x) = \mathbb{E}_D\left[f(x;D)\right]$

y is the true label for x and y_d is the label for x in data D, we have training error:

$$
\begin{aligned}
&= \mathbb{E}_D\left[\{f(x;D) - y_d\}^2\right] \\
&= \mathbb{E}_D\left[\{f(x;D) - \bar{f}(x) + \bar{f}(x) - y_d\}^2\right] \\
&= \mathbb{E}_D\left[\{f(x;D) - \bar{f}(x)\}^2\right] + \mathbb{E}_D\left[\{\bar{f}(x) - y_d\}^2\right] + 2\mathbb{E}_D\left[\{(f(x;D) - \bar{f}(x)) \cdot (\bar{f}(x) - y_d)\}\right] \\
&= \mathbb{E}_D\left[\{f(x;D) - \bar{f}(x)\}^2\right] + \mathbb{E}_D\left[\{\bar{f}(x) - y_d\}^2\right] \\
&= \mathbb{E}_D\left[\{f(x;D) - \bar{f}(x)\}^2\right] + \mathbb{E}_D\left[\{\bar{f}(x) - y + y - y_d\}^2\right] \\
&= \mathbb{E}_D\left[\{f(x;D) - \bar{f}(x)\}^2\right] + \mathbb{E}_D\left[\{\bar{f}(x) - y\}^2\right] + \mathbb{E}_D\left[\{y - y_d\}^2\right] + 2\mathbb{E}_D\left[\{\bar{f}(x) - y\}\{y - y_d\}\right] \\
&= \mathbb{E}_D\left[\{f(x;D) - \bar{f}(x)\}^2\right] + \{\bar{f}(x) - y\}^2 + \mathbb{E}_D\left[\{y - y_d\}^2\right] \\
&= Variance + Bias + Noise
\end{aligned}
$$

Bias-variance decomposition





Outline

- Machine Learning Basics
  - tasks/problems
  - models
  - algorithms
- Model Evaluation
- Research:
  - trade off of large scale learning
  - quantifying generalization error in deep learning

Research paper: quantifying generalization error in deep learning

Bottou L, etc. The tradeoffs of Large scale learning （2018 NIPs best)
Jin P, Lu L, etc. Quantifying the generalization error in deep learning in terms of data distribution and neural network smoothness.

The Tradeoffs of Large Scale Learning.

Data: $(x, y) \in \mathcal{X} \times \mathcal{Y}$

Goal: find $y = f^*(x)$ i.e. conditional distribution $P(y|x)$.

Define: hypothesis space $\mathcal{F}$.

e.g. $\mathcal{F} = \{ \sum_j a_j \varphi_j(x), \; a_j \in R \}$. for regression or classification.

$\mathcal{F} = \{ \sum_j a_j \sigma(w_j^T x + b_j), \; w_j \in R^d, \; a_j, b_j \in R \}$. σ is activation.

for shallow Neural Nets.

$\mathcal{F} = \{ F_T \circ F_{T-1} \circ \cdots \circ F_0(x)$, each F_t is a shallow NN $\}$.

for deep Neural Nets.

Find best f^* that minimize the expected risk:

$$E(f) = \int l(f(x), y)\, dP(x,y)$$
$$= \mathbb{E}[l(f(x), y)].$$

Denote: $f^*_{\mathcal{F}}(x) = \underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \mathbb{E}[l(f(x), y)]$ $\quad$ f^* not need to be in $\mathcal{F}$.

Denote: $f_n = \underset{f \in \mathcal{F}}{\operatorname{argmin}}\, E_n(f)$ $\quad$ where $E_n(f) = \frac{1}{n}\sum_{i=1}^{n} l(f(x_i), y_i)$
$$= \mathbb{E}_n[l(f(x), y)]$$

Denote: $\tilde{f}_n$ s.t.

$E_n(\tilde{f}_n) - E_n(f_n) < \rho.$ $\quad$ assuming our minimization algorithm returns approximate solution $\tilde{f}_n$, s.t.

$$E_n(\tilde{f}_n) - E_n(f_n) < \rho \quad (\rho \geq 0)$$

Decomposition:

$$\mathcal{E} = \mathbb{E}\left[E(f^*_{\mathcal{F}}) - E(f^*)\right] + \mathbb{E}\left[E(f_n) - E(f^*_{\mathcal{F}})\right] + \mathbb{E}\left[E(\tilde{f}_n) - E(f_n)\right].$$

$\mathbb{E}$: random choice of the training set.

- $\mathcal{E}_{app.}$: how close $\mathcal{F}$ to f^*.
  - Reduce by using a larger model
- $\mathcal{E}_{est.}$: the effect of training examples and model compacity.
  - Reduce by ① increasing example size ② choosing smaller model.
- $\mathcal{E}_{opt.}$: impact of approximate optimization on the generalization performance.
  - Reduce by
    - ① Running opt. alg. longer
    - ② choosing more efficient alg. with faster conv. rate.

Opt. Problem:

$$\min_{\mathcal{F}, \rho, n} \mathcal{E} = \mathcal{E}_{app} + \mathcal{E}_{est.} + \mathcal{E}_{opt.}$$

$\mathcal{F}$ → task difficulty; ρ → opt. alg.; n → training data

subject to
$$\begin{cases} n \le n_{max} \\ T(\mathcal{F}, \rho, n) \le T_{max} \end{cases}$$

- Small-scale ML tasks:
  - Mainly contrained by training data size n.
  - Computing time is not an issue. can choose small ρ.
  - balance $\mathcal{E}_{app.}$ and $\mathcal{E}_{est.}$
- Large-scale ML tasks:
  - Mainly constrained by time. so SGD preferred for $\mathcal{E}_{opt}$
  - n is large, $\mathcal{E}_{est.}$ can be reduced.
  - Large model is preferred to reduce $\mathcal{E}_{app.}$

Algorithm	Cost of one iteration	Iterations to reach ρ	Time to reach accuracy ρ	Time to reach $\mathcal{E} \leq c\,(\mathcal{E}_{\text{app}} + \varepsilon)$
GD	$\mathcal{O}(nd)$	$\mathcal{O}\left(\kappa \log \frac{1}{\rho}\right)$	$\mathcal{O}\left(nd\kappa \log \frac{1}{\rho}\right)$	$\mathcal{O}\left(\frac{d^2\,\kappa}{\varepsilon^{1/\alpha}} \log^2 \frac{1}{\varepsilon}\right)$
2GD	$\mathcal{O}(d^2 + nd)$	$\mathcal{O}\left(\log\log \frac{1}{\rho}\right)$	$\mathcal{O}\left((d^2 + nd) \log\log \frac{1}{\rho}\right)$	$\mathcal{O}\left(\frac{d^2}{\varepsilon^{1/\alpha}} \log \frac{1}{\varepsilon} \log\log \frac{1}{\varepsilon}\right)$
SGD	$\mathcal{O}(d)$	$\frac{\nu\kappa^2}{\rho} + \mathrm{o}\left(\frac{1}{\rho}\right)$	$\mathcal{O}\left(\frac{d\nu\kappa^2}{\rho}\right)$	$\mathcal{O}\left(\frac{d\,\nu\,\kappa^2}{\varepsilon}\right)$
2SGD	$\mathcal{O}(d^2)$	$\frac{\nu}{\rho} + \mathrm{o}\left(\frac{1}{\rho}\right)$	$\mathcal{O}\left(\frac{d^2\nu}{\rho}\right)$	$\mathcal{O}\left(\frac{d^2\,\nu}{\varepsilon}\right)$

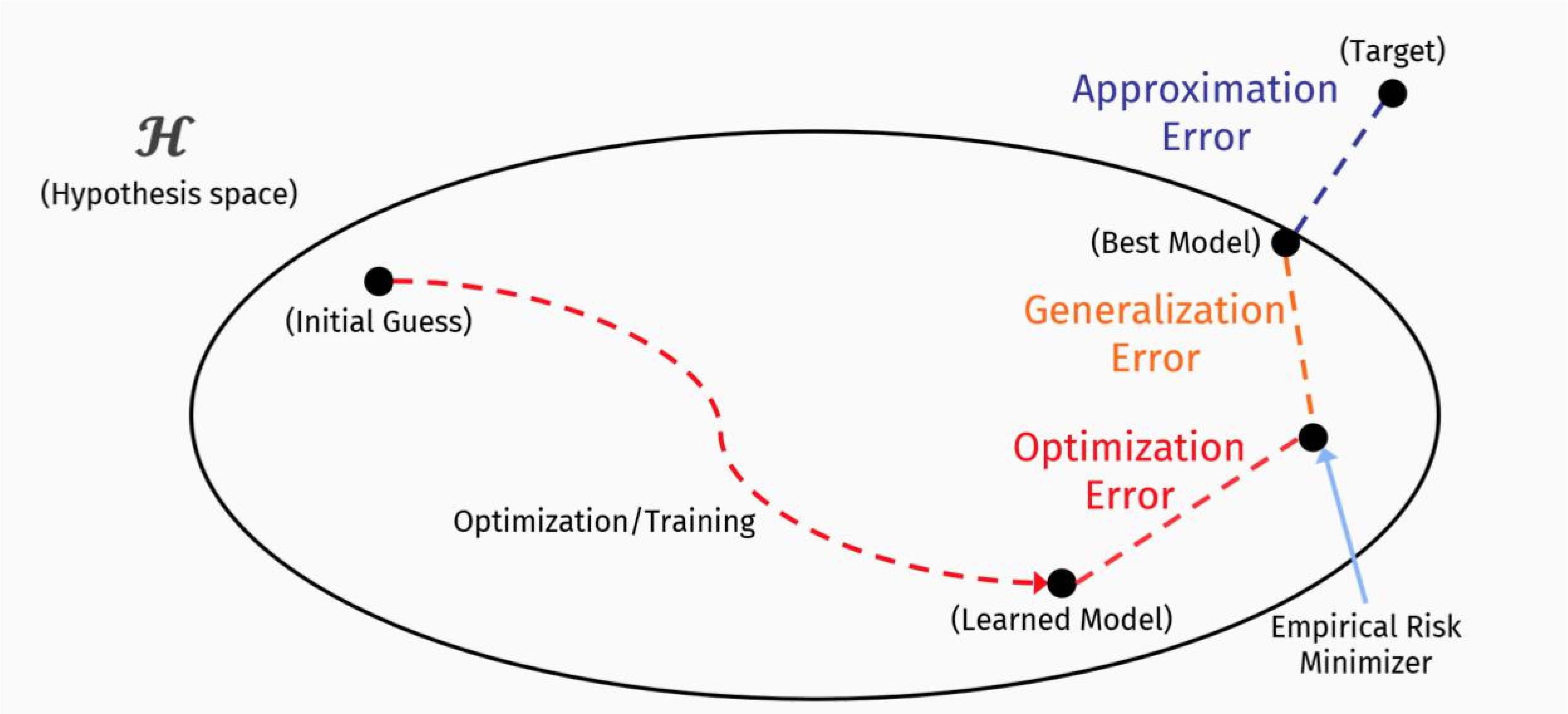
$$\begin{aligned}\mathcal{E} &= \mathbb{E}\left[E(f^*_{\mathcal{F}}) - E(f^*)\right] + \mathbb{E}\left[E(f_n) - E(f^*_{\mathcal{F}})\right] + \mathbb{E}\left[E(\tilde{f}_n) - E(f_n)\right] \\ &= \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}}.\end{aligned}$$

large-scale learning systems:

- ✓ depends on objective function + computational properties of the chosen optimization algorithm.
- ✓ SGD and 2SGD results do not depend on the estimation rate α. When the estimation rate is poor, there is less need to optimize accurately, leave time to process more examples.
- ✓ Stochastic algorithms (SGD, 2SGD) yield the best generalization performance despite showing the worst optimization performance on the empirical cost.

small-scale learning systems:

- ✓ generalization performance is solely determined by the statistical properties of the objective function



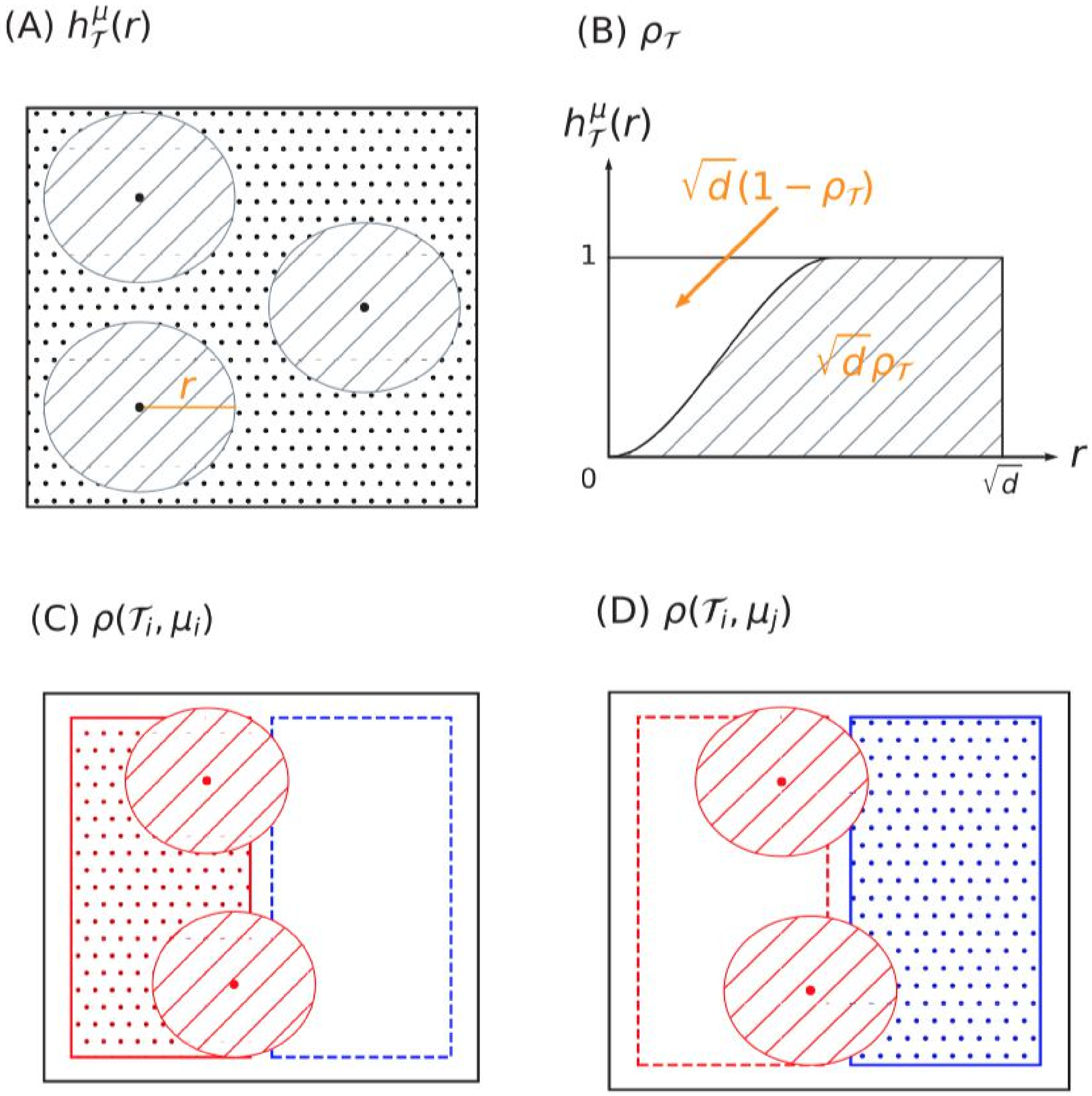
Quantifying the generalization error in deep learning in terms of data distribution and neural network smoothness.

$$\mathcal{E} = \mathbb{E}\left[E(f^*_{\mathcal{F}}) - E(f^*)\right] + \mathbb{E}\left[E(f_n) - E(f^*_{\mathcal{F}})\right] + \mathbb{E}\left[E(\tilde{f}_n) - E(f_n)\right]$$

(approximation error, generalization error, optimization error)

Question:

- training data
- model compacity
- smoothness of Neural Network



$$CD(\mathcal{T}) := \frac{1}{K}\sum_i \rho(\mathcal{T}_i, \mu_i) - \frac{1}{K(K-1)}\sum_{i\neq j} \rho(\mathcal{T}_i, \mu_j),$$

$$\mathcal{T} = \{x_1, x_2, \ldots, x_n\} \subseteq D.$$

(A) $h^{\mu}_{\mathcal{T}}(r) := \mu\left(D \cap \bigcup_{x_i \in \mathcal{T}} B(x_i, r)\right)$ *data density*

$$\rho(\mathcal{T}, \mu) := \frac{1}{\sqrt{d}}\int_0^{\sqrt{d}} h^{\mu}_{\mathcal{T}}(r)dr.$$

(B)

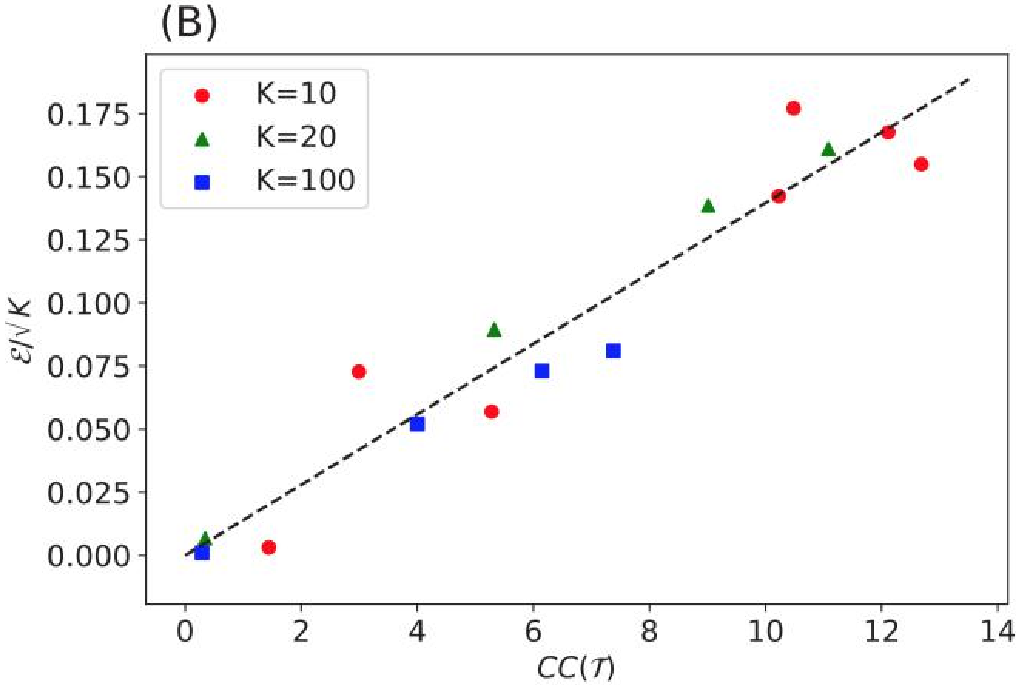
Data cover complexity: *white area: data sparsity*

$$CC(\mathcal{T}) := \frac{1-\rho_{\mathcal{T}}}{CD(\mathcal{T})}$$

(1) unchanged with scaled data points, only related to distance between the points;
(2) numerator: the smaller, the better;
(3) denominator: the bigger the better, indicating data under different labels seperated better

- training data

The best accuracy that can be achieved in practice (i.e., optimized by stochastic gradient descent) by fully-connected networks is approximately linear with respect to the cover complexity of the data set.



- model compacity

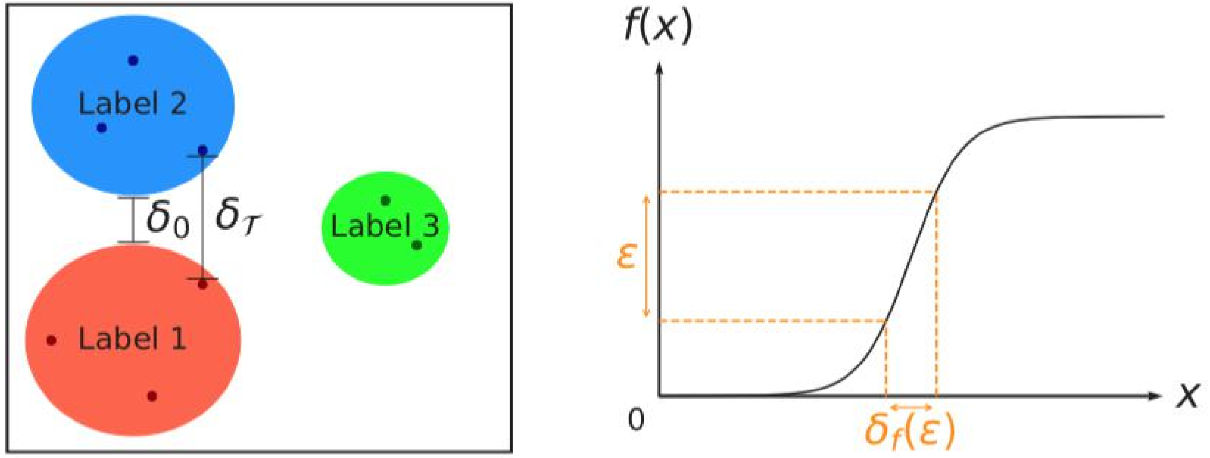
Define: c-Accuracy (smaller than the true accuracy)

$$p_c(f) := \frac{\mu(H_c^f)}{\mu(D)} = \mu(H_c^f),$$

where $H_c^f := \{x \in D | f \text{ is } c\text{-accurate at } x\}$.

$$f : D \to \mathbb{R}^K \quad . f_{i_{max}}(x) > c. \qquad (c > 0.5)$$

(E) δ_0 & δ_T (F) $\delta_f(\varepsilon)$



$$p_c(f) \geq 1 - \frac{\sqrt{d}}{\delta}(1 - \rho_T),$$

second term:

numerator: data sparsity

denominator: smoothness

- smoothness of Neural Network

The trend of the expected accuracy is consistent with the smoothness of the neural network, which provides a new ‘‘early stopping’’ strategy by monitoring the smoothness of the neural network.

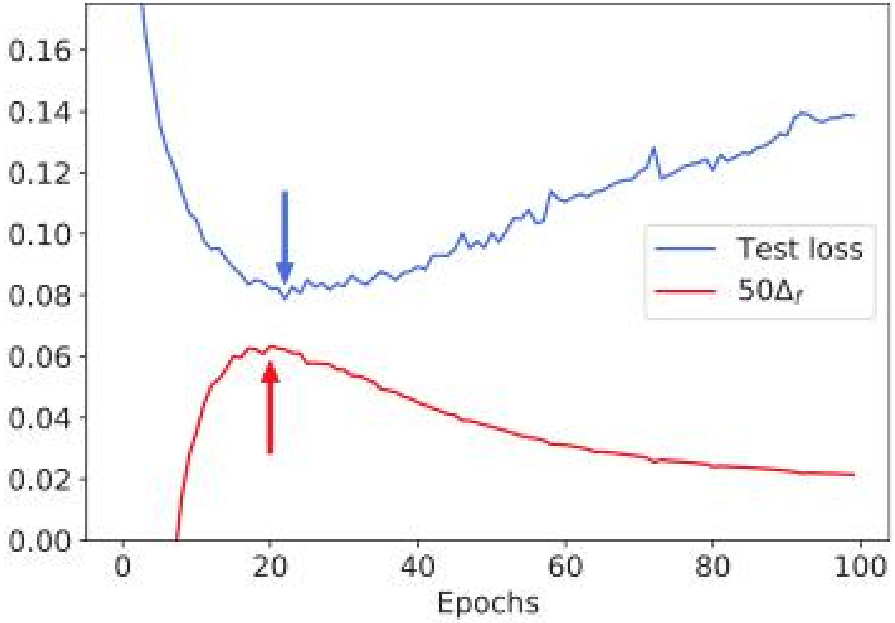


Fig. 6. Consistency between test loss and neural network smoothness during the training of the neural network for MNIST. The arrows indicate the minimum of the test loss and the maximum of Δ_f.

Some Limitations:

1. Assuming setup in multi-class classfication with max predicted component of the result > 0.5.

2. Assuming smoothness of approximation neural network.