

# Theoretical Implications

CS 535: Deep Learning

# Machine Learning Theory: Basic setup

- Generic supervised learning setup:
- For  $(x_i, y_i)_{1 \dots n}$  **i.i.d.** drawn from the joint distribution  $P(x, y)$ , find a best function  $f \in F$  that minimizes the error  $E_{x,y}[L(f(x), y)]$

- $L$  is a loss function, e.g.

- Classification:

$$L(f(x), y) = \begin{cases} 1, & f(x) \neq y \\ 0, & f(x) = y \end{cases}$$

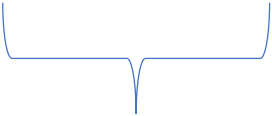
- Regression:  $L(f(x), y) = (f(x) - y)^2$

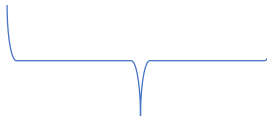
- $F$  is a function class (consists many functions, e.g. all linear functions, all quadratic functions, all smooth functions, etc.)


# Machine Learning Theory: Generalization

- Machine learning theory is about **generalizing** to unseen examples
  - **Not** the training set error!
  - And those theory **doesn't always** hold (holds with probability less than 1)
- A generic machine learning generalization bound:
  - For  $(x_i, y_i)_{1\dots n}$  drawn from the joint distribution  $P(x, y)$ , with probability  $1 - \delta$

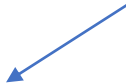
$$E_{x,y}(f(x) \neq y) \leq \frac{1}{n} \sum_{i=1}^n L(f(x_i), y_i) + \Omega(F, \delta)$$

  
Error on the  
whole distribution

  
Error on the  
training set

  
Flexibility of the  
function class

How to represent  
“flexibility”? That’s a  
course on ML theory

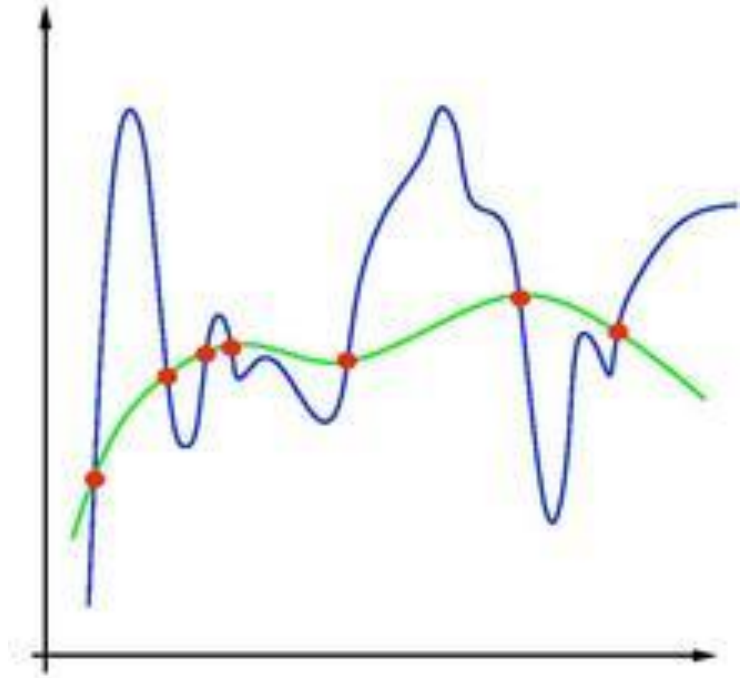


# What is “flexibility”?

- Roughly, the more functions in  $F$ , the more flexible it is
- Function class: all linear functions  $F: \{f(x) | f(x) = w^\top x + b\}$ 
  - Not very flexible, cannot even solve XOR
  - Small “flexibility” term, testing error not much more than training error
- Function class: all 9-th degree polynomials
$$F: \{f(x) | f(x) = w_1^\top x^9 + \dots\}$$
  - Super flexible
  - Big “flexibility” term, testing error can be **much more** than training

# Flexibility and overfitting

- For a very flexible function class
  - Training error is **NOT** a good measure of testing error
  - Therefore, out-of-sample error estimates are needed
    - Separate validation set to measure the error
    - Cross-validation
      - K-fold
      - Leave-one-out
    - Many times this will show to be worse than the training error with a flexible function class



# Another twist of the generalization inequality

- Nevertheless, you still want training error to be **small**
- So you don't always want to use linear classifiers/regressors

$$E_{x,y}(f(x) \neq y) \leq \frac{1}{n} \sum_{i=1}^n L(f(x_i), y_i) + \Omega(F, \delta)$$

If this is 60% error...

Add-on term

Error on the whole distribution

Error on the training set

Flexibility of the function class

# How to deal with it when you do use a flexible function class

- Regularization

- To make the chance of choosing a highly flexible function to be low

- Example:

- Ridge Regression:

$$\min_w (w^\top X - Y)^2 + \lambda \|w\|^2$$

In order to choose a  $w$  with big  $\|w\|^2$  you need to overcome this term

- Kernel SVM

$$\min_f \sum_i L(f(x_i), y_i) + \lambda \|f\|^2$$

In order to choose a very unsmooth function  $f$  you need to overcome this term

# Bayesian Interpretation of Regularization

- Assume that a certain prior of the parameters exist, and optimize for the MAP estimate

- Example:

- Ridge Regression: Gaussian prior on  $w$ :  $P(w) = C \exp(-\lambda ||w||^2)$

$$\min_w (w^T X - Y)^2 + \lambda ||w||^2$$

- Kernel SVM: Gaussian process prior on  $f$  (too complicated to explain simply..)

$$\min_f \sum_i L(f(x_i), y_i) + \lambda ||f||^2$$



# Universal Approximators

- Universal Approximators
  - (Barron 1994, Bartlett et al. 1999) Meaning that they can approximate (learn) **any** smooth function efficiently (meaning using a polynomial number of hidden units)
  - Kernel SVM
  - Neural Networks
  - Boosted Decision Trees
- Machine learning cannot do much better
  - No free lunch theorem

# No Free Lunch

- (Wolpert 1996, Wolpert 2001) For any 2 learning algorithms, averaged over any training set  $d$  and over all possible distributions  $P$ , their average error is **the same**
- Practical machine learning only works because of certain **correct assumptions** about the data
  - SVM succeeds by successfully representing the general **smoothness** assumption as a convex optimization problem (with global optimum)
  - However, if one goes for more complex assumptions, convexity is very hard to achieve!

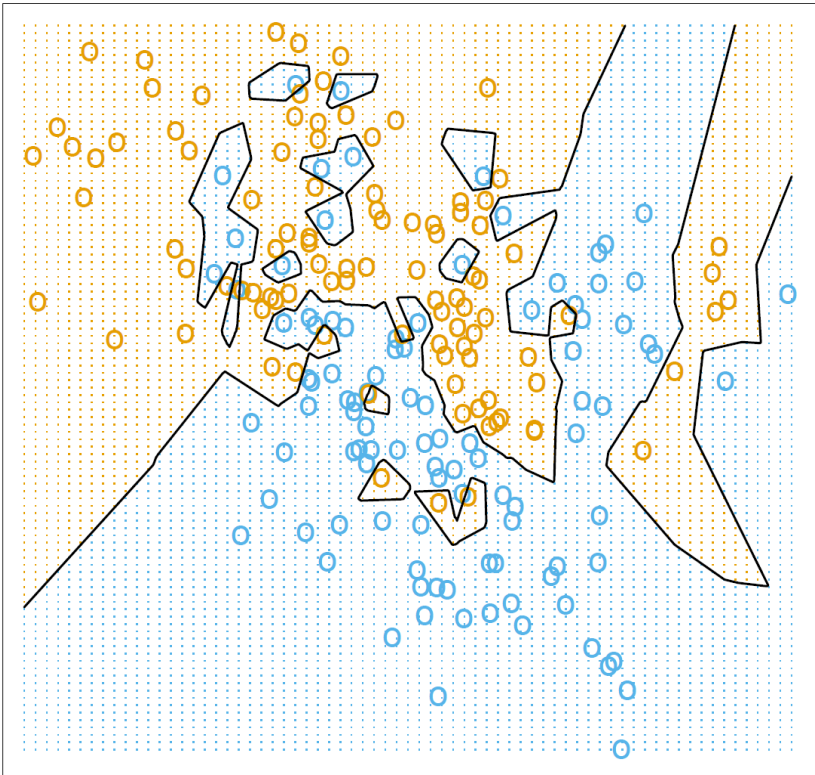
# High-dimensionality

Philosophical discussion about high-dimensional spaces

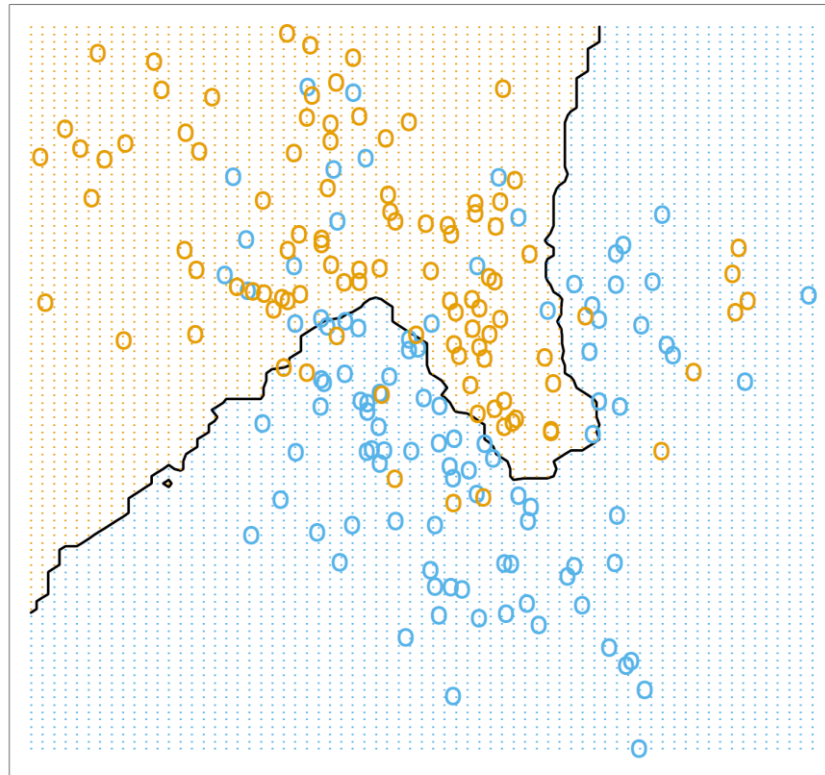
# Distance-based Algorithms

- K-Nearest Neighbors: weighted average of k-nearest neighbors

1-Nearest Neighbor Classifier

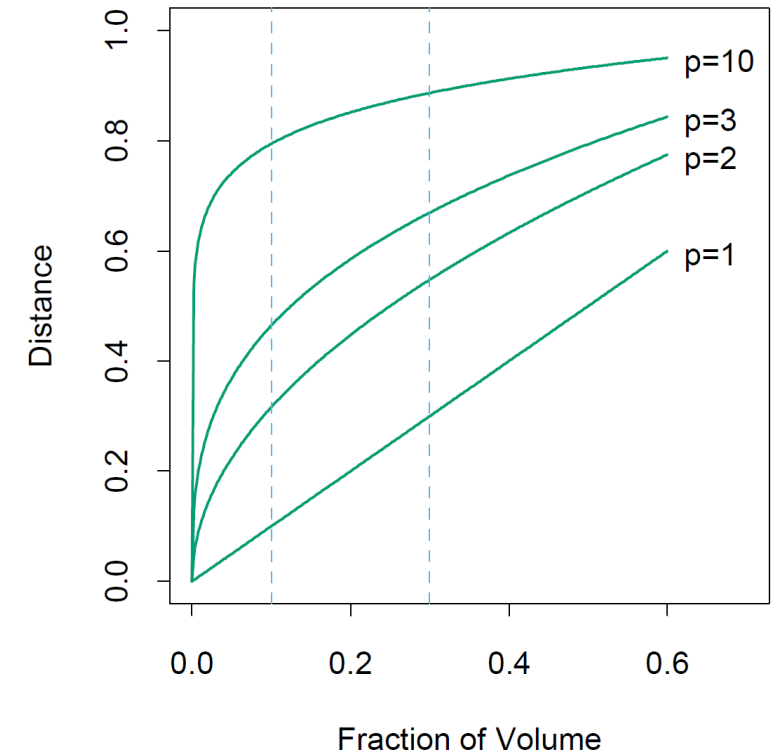
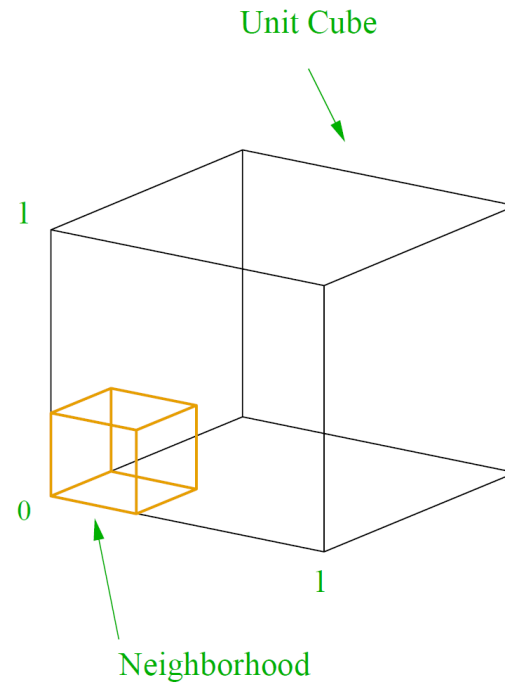


15-Nearest Neighbor Classifier



# Curse of Dimensionality

- Dimensionality brings interesting effects:
- In a 10-dim space, to cover 10% of the data in a unit cube, one needs a box to cover 80% of the range



**FIGURE 2.6.** The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction  $r$  of the volume of the data, for different dimensions  $p$ . In ten dimensions we need to cover 80% of the range of each coordinate to capture 10% of the data.

# High Dimensionality Facts

- Every point is on the boundary
  - With  $N$  uniformly distributed points in a  $p$ -dimensional ball, the closest point to the origin has a median distance of

$$d(p, N) = \left(1 - \frac{1}{2}^{1/N}\right)^{1/p}$$

- Every vector is almost always orthogonal to each other
  - Pick 2 unit vectors  $x_1$  and  $x_2$ , then the probability that

$$\cos(x_1, x_2) = |x_1^\top x_2| \geq \sqrt{\frac{\log p}{p}}$$

is less than  $1/p$

# Avoiding the Curse

- Regularization helps us with the curse
  - Smoothness constraints also grow stronger with the dimensionality!

$$\int |f'(x)| dx \leq C$$

$$\int \left| \frac{\partial f}{\partial x_1} \right| dx_1 + \int \left| \frac{\partial f}{\partial x_2} \right| dx_2 + \dots + \int \left| \frac{\partial f}{\partial x_p} \right| dx_p \leq C$$

- We do not suffer from the curse if we ONLY estimate sufficiently smooth functions!

# Rademacher and Gaussian Complexity

Why would CNN make sense



# Rademacher and Gaussian Complexity

*Define the random variable*

$$\hat{R}_n(F) = \mathbf{E} \left[ \sup_{f \in F} \left| \frac{2}{n} \sum_{i=1}^n \sigma_i f(X_i) \right| \middle| X_1, \dots, X_n \right],$$

*where  $\sigma_1, \dots, \sigma_n$  are independent uniform  $\{\pm 1\}$ -valued random variables. Then the Rademacher complexity of  $F$  is  $R_n(F) = \mathbf{E} \hat{R}_n(F)$ . Similarly, define the random variable*

$$\hat{G}_n(F) = \mathbf{E} \left[ \sup_{f \in F} \left| \frac{2}{n} \sum_{i=1}^n g_i f(X_i) \right| \middle| X_1, \dots, X_n \right],$$

*where  $g_1, \dots, g_n$  are independent Gaussian  $N(0, 1)$  random variables. The Gaussian complexity of  $F$  is  $G_n(F) = \mathbf{E} \hat{G}_n(F)$ .*

**Lemma 4** *There are absolute constants  $c$  and  $C$  such that for every class  $F$  and every integer  $n$ ,  $cR_n(F) \leq G_n(F) \leq C \ln n R_n(F)$ .*

# Risk Bound

**Theorem 5** *Let  $P$  be a probability distribution on  $\mathcal{X} \times \{\pm 1\}$ , let  $F$  be a set of  $\{\pm 1\}$ -valued functions defined on  $\mathcal{X}$ , and let  $(X_i, Y_i)_{i=1}^n$  be training samples drawn according to  $P^n$ .*

*(b) With probability at least  $1 - \delta$ , every function  $f$  in  $F$  satisfies*

$$P(Y \neq f(X)) \leq \hat{P}_n(Y \neq f(X)) + \frac{R_n(F)}{2} + \sqrt{\frac{\ln(1/\delta)}{2n}}.$$

# Complexity Bound for NN

**Theorem 18** *Suppose that  $\sigma : \mathbb{R} \rightarrow [-1, 1]$  has Lipschitz constant  $L$  and satisfies  $\sigma(0) = 0$ . Define the class computed by a two-layer neural network with 1-norm weight constraints as*

$$F = \left\{ x \mapsto \sum_i w_i \sigma(v_i \cdot x) : \|w\|_1 \leq 1, \|v_i\|_1 \leq B \right\}.$$

*Then for  $x_1, \dots, x_n$  in  $\mathbb{R}^k$ ,*

$$\hat{G}_n(F) \leq \frac{cLB(\ln k)^{1/2}}{n} \max_{j, j'} \sqrt{\sum_{i=1}^n (x_{ij} - x_{ij'})^2},$$

*where  $x_i = (x_{i1}, \dots, x_{ik})$ .*

# References

- (Barron 1994) A. R. Barron (1994). Approximation and estimation bounds for artificial neural networks. *Machine Learning*, Vol.14, pp.113-143.
- (Martin 1999) Martin A. and Bartlett P. *Neural Network Learning: Theoretical Foundations* 1st Edition
- (Wolpert 1996) WOLPERT, David H., 1996. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7), 1341–1390.
- (Wolpert 2001) WOLPERT, David H., 2001. The supervised learning no-free-lunch theorems. In: *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*.
- (Rahimi and Recht 2007) Rahimi A. and Recht B. Random Features for Large-Scale Kernel Machines. *NIPS 2007*.