CS5339 Machine Learning

Estimation II

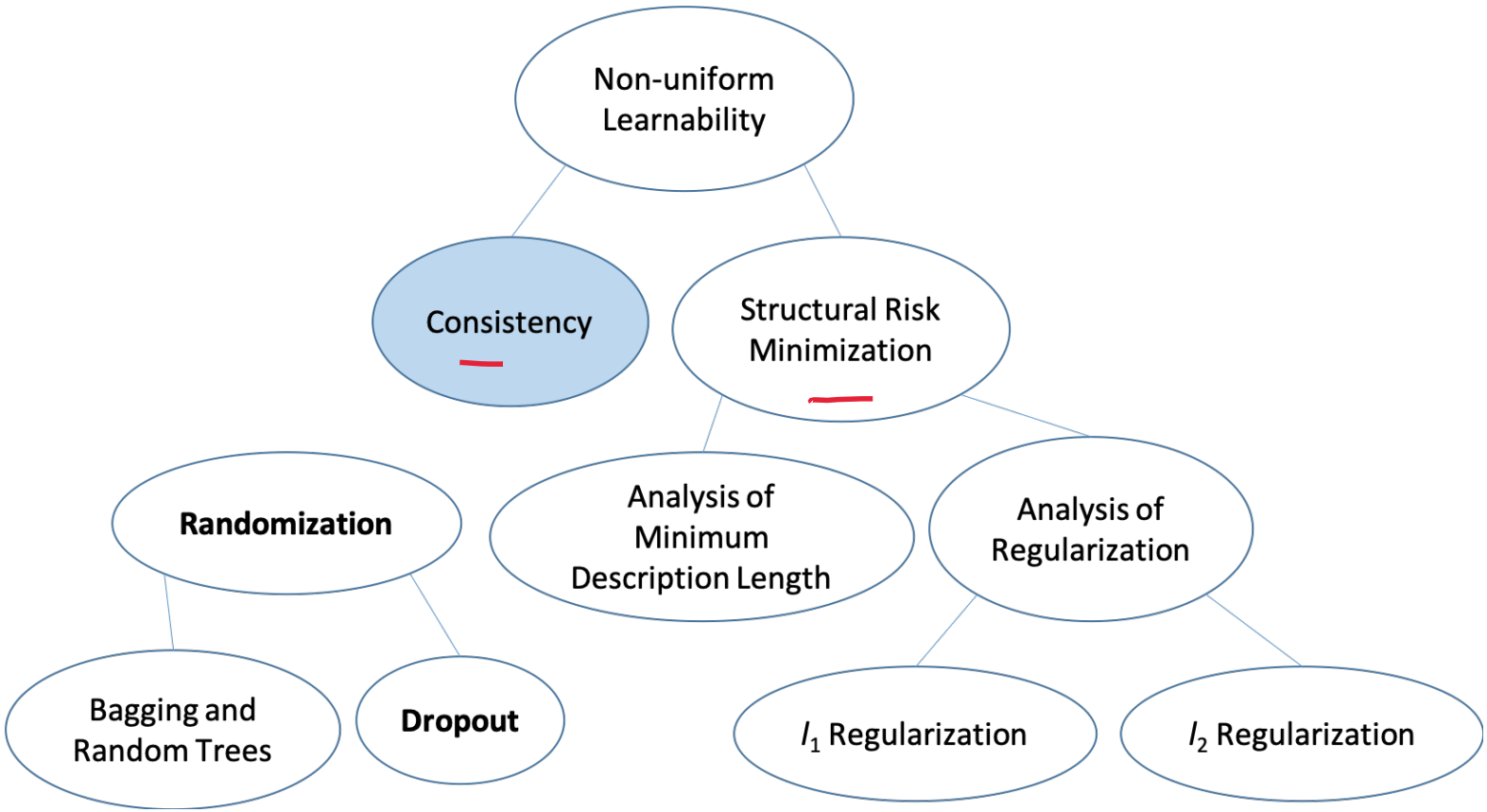
Lee Wee Sun
School of Computing
National University of Singapore
leews@comp.nus.edu.sg

Semester 2, 2019/20

Estimation

We look at more refined ways to understand generalization performance and how it leads to different algorithm designs.

Outline


Non-uniform Learnability
Consistency
Structural Risk Minimization
Randomization
Analysis of Minimum Description Length
Analysis of Regularization
Bagging and Random Trees
Dropout
l_1 Regularization
l_2 Regularization

Non-Uniform Learnability and Consistency

- In *PAC learning*, we insist on having a single sample size $m_{\mathcal{H}}(\epsilon, \delta)$ that works for all $h \in \mathcal{H}$. and all distr D
  - This leads to algorithms that restricts $\mathcal{H}$ to a reasonable class of functions then minimize the empirical risk (ERM) using $\mathcal{H}$.
- For *non-uniform learning*, we allow each hypothesis h to be estimated to accuracy ϵ using a different number of examples, or equivalently, for a fixed sample size m, *different hypotheses* are estimated with different accuracy.
  - This leads to algorithms that minimize different criterions.
- For *consistency*, we allow the sample size required for learning to accuracy ϵ with confidence $1 - \delta$ to depend on both the *hypothesis* as well as the *distribution*.

- In PAC learning, we are able to tell in advance the sample size required, given ϵ and δ.
  - Requires strong inductive bias: specify hypothesis class.

Example: lin fn with few features

- In non-uniform learning, the bound is output sensitive.
  - Depends on the hypothesis that is selected.
  - One way is to break up hypothesis class into union of smaller classes and do structural risk minimization (next section).
  - Learning with various regularization schemes can be viewed as non-uniform learning.

- For consistency, we only know that the estimation error will eventually converge to zero.
  - No requirement on rate of convergence. Weaker requirement.
  - Algorithms like k-nearest neighbour are universally Bayes consistent – converges to the best possible error for any distribution.

Exercise 1:

Universally Bayes consistent algorithms will eventually learn any function. But the no-free-lunch theorem says that no algorithm can learn the class of all classifiers in an infinite domain. Is there a contradiction? Why?

No free lunch says there is no alg that will guarantee desired err for all possible distr using the same sample size!

Consistency relaxes req to diff sample size for diff target & diff distr.

Discussion 2:

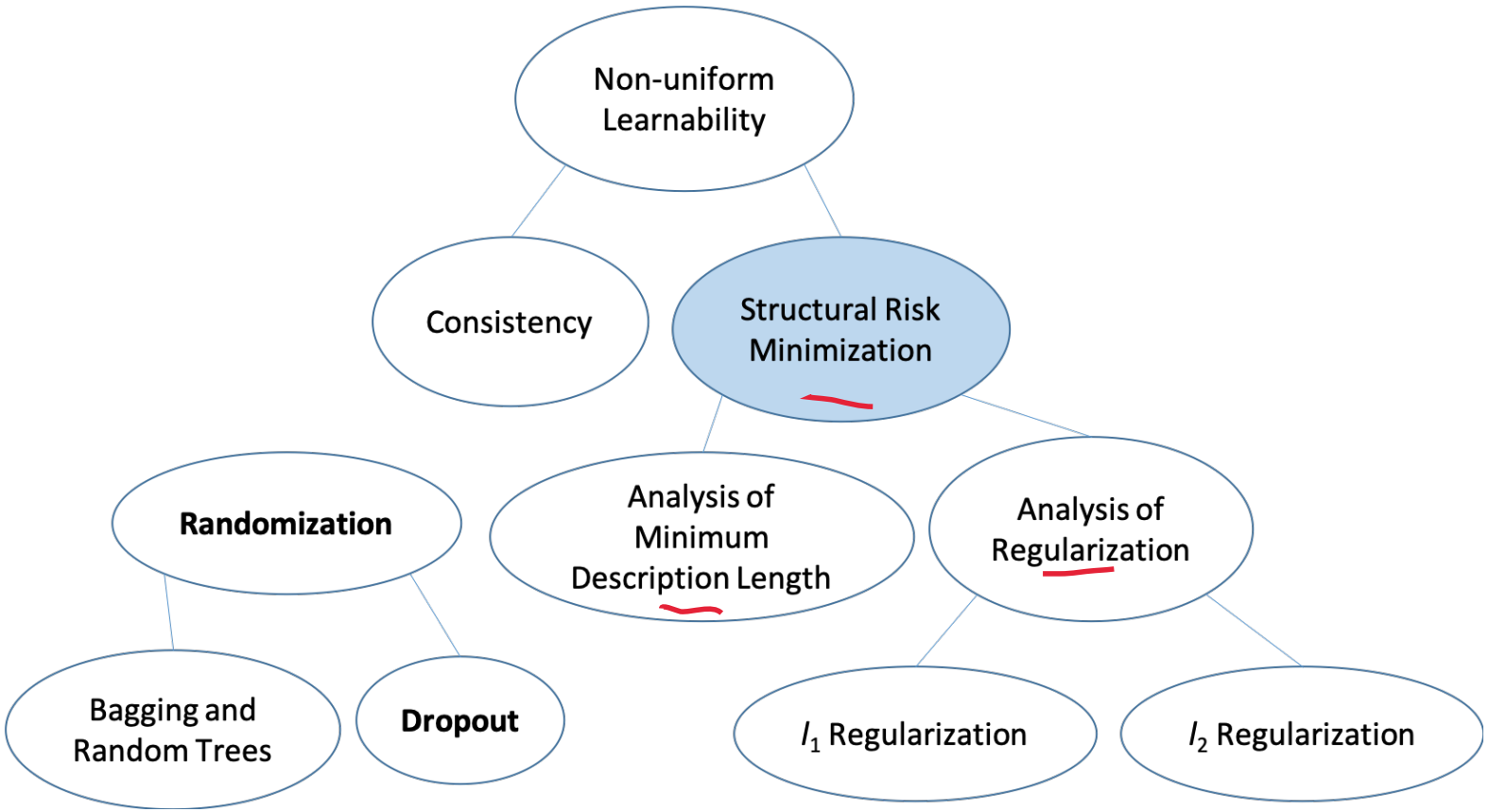
What is the role of impossiblity theorems such as the no-free-lunch theorem? What other impossibility theorems do you know and how are they useful?

Don't waste time dump impossible things.

Perpetual motion machines.

Lower bounds

Outline


Non-uniform Learnability
Consistency
Structural Risk Minimization
Randomization
Analysis of Minimum Description Length
Analysis of Regularization
Bagging and Random Trees
Dropout
l_1 Regularization
l_2 Regularization

Structural Risk Minimization

- One way to do non-uniform learning.
- Assume that $\mathcal{H}$ can be decomposed into $\cup_{n\in\mathbb{N}}\mathcal{H}_n$.
- Specify a weight function $w : \mathbb{N} \to [0, 1]$ which assigns weight to each class $\mathcal{H}_n$, such that a larger weight reflects a stronger preference for that class with $\sum_{n=1}^{\infty} w(n) \leq 1$, e.g. $w(n) = 2^{-n}$ or $w(n) = \frac{6}{\pi^2 n^2}$.

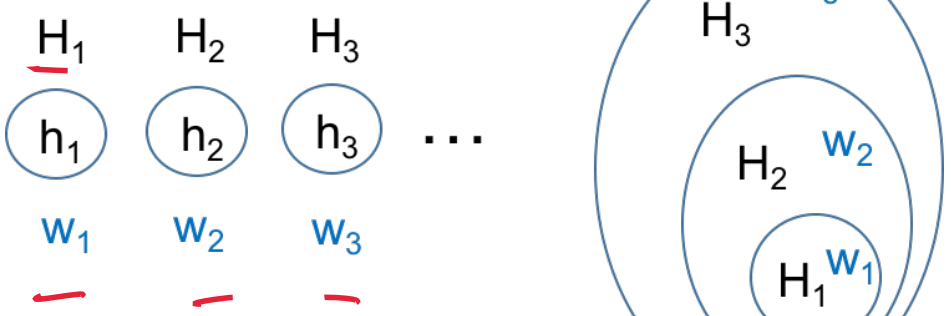


Figure: Examples of hypothesis class decompositions $\cup_{n\in\mathbb{N}}\mathcal{H}_n$. On the left, countable hypothesis class decomposed into singleton classes (single hypothesis). On the right, a nested decomposition.

- Assume $\mathcal{H}_n$ enjoys uniform convergence property with sample complexity function $m^{UC}_{\mathcal{H}_n}(\epsilon, \delta)$ and define

$$\epsilon_n(m, \delta) = \min\{\epsilon \in (0,1) : m^{UC}_{\mathcal{H}_n}(\epsilon, \delta) \leq m\}.$$

Theorem: (SSBD Theorem 7.4) Let $w : \mathbb{N} \to [0,1]$ be a function such that $\sum_{n=1}^{\infty} w(n) \leq 1$. Let $\mathcal{H}$ be a hypothesis class that can be written as $\mathcal{H} = \cup_{n \in \mathbb{N}} \mathcal{H}_n$, where for each n, $\mathcal{H}_n$ satisfies the uniform convergence property with a sample complexity function $m^{UC}_{\mathcal{H}_n}$. Let $\epsilon_n(m, \delta) = \min\{\epsilon \in (0,1) : m^{UC}_{\mathcal{H}_n}(\epsilon, \delta) \leq m\}$. Then, for every $\delta \in (0,1)$ and distribution $\mathcal{D}$, with probability at least $1 - \delta$ over the choice of $S^m \sim \mathcal{D}^m$ the following bound holds (simultaneously) for every $n \in \mathbb{N}$ and $h \in \mathcal{H}_n$.

$$|L_{\mathcal{D}}(h) - L_S(h)| \leq \epsilon_n(m, w_n \delta).$$

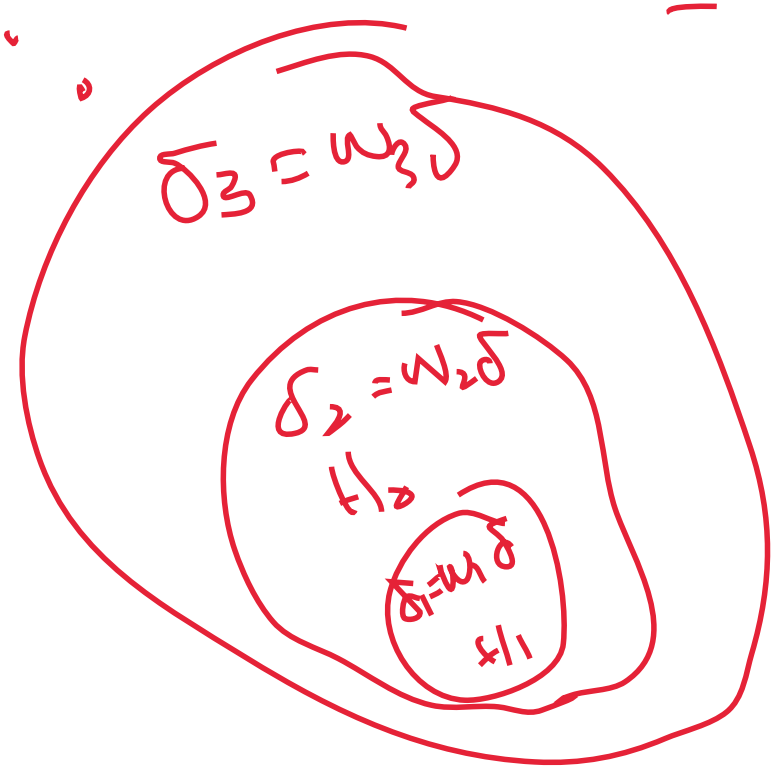
Therefore, for every $\delta \in (0,1)$, and distribution $\mathcal{D}$, with probability at least $1 - \delta$ it holds that

$$\forall h \in \mathcal{H}, \qquad L_{\mathcal{D}}(h) \leq L_S(h) + \min_{n : h \in \mathcal{H}_n} \epsilon_n(m, w_n \delta).$$

Proof: For each n, let $\delta_n = w(n)\delta$. Then we have

$$\forall h \in \mathcal{H}, \qquad |L_{\mathcal{D}}(h) - L_S(h)| \leq \epsilon_n(m, \delta_n).$$

Applying the union bound for all n, we get that it holds for all n with probability at least $1 - \sum_n \delta_n = 1 - \delta \sum_n w(n) \geq 1 - \delta$. $\square$



The structure risk minimization paradigm suggests minimizing $L_S(h) + \min_{n:h\in\mathcal{H}_n} \epsilon_n(m, w_n\delta)$, which is an upper bound to the risk of h.

If the upper bound is small, we are guaranteed that the risk will be small.

Example: Nested class
e.g. using number of nodes in a neural net.

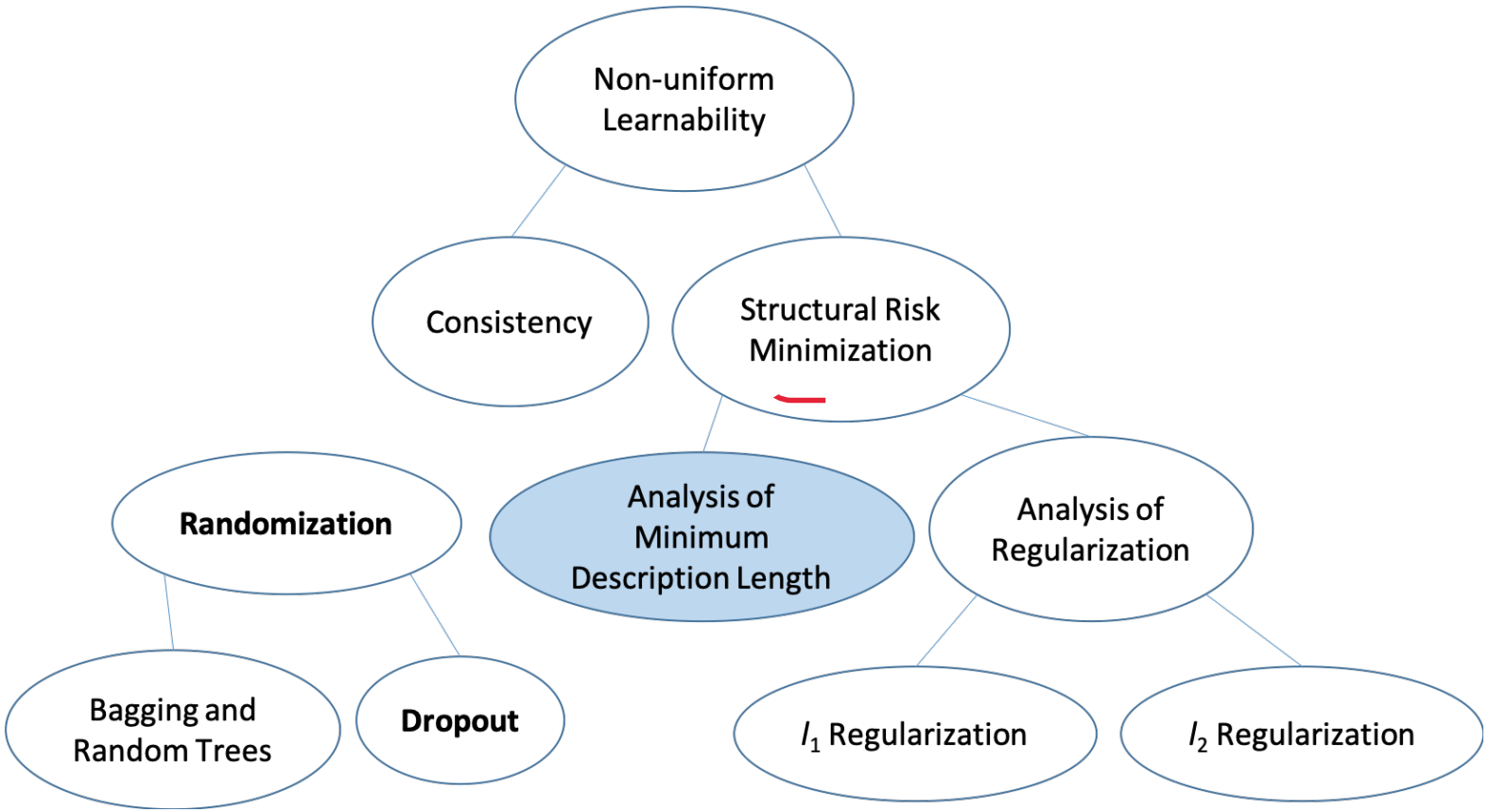
Exercise 3:

Let $\mathcal{H}$ be the class of linear functions in $\mathbb{R}^d$. Suggest a way to decompose $\mathcal{H}$ into $\cup_{n\in\mathbb{N}}\mathcal{H}_n$, i.e. what might each $\mathcal{H}_n$ be?

Use l_2 norm of weights
- correspond to l_2 regularization

Use l_1 norm
- correspond to l_1 reg.

Outline


Non-uniform Learnability
Consistency
Structural Risk Minimization
Randomization
Analysis of Minimum Description Length
Analysis of Regularization
Bagging and Random Trees
Dropout
l_1 Regularization
l_2 Regularization

Occam's Razor

Exercise 4:

Assume that you are given two hypotheses that has zero empirical risk. Which one should you choose and why?

1. Use Occam's Razor and select that one that has a shorter description. In the philosophy of science, Occam's razor is a problem solving principle that suggests that when there are two explanations that explain the data equally well, we should select the simpler one. One interpretation of *simpler* is having a shorter description.
2. Select the hypothesis that has higher probability of being the correct hypothesis.

- If you know the prob, option 2 is optimal. But usually prob is subjective. People may disagree.
- Occam's razor is a reasonable approx under assumptions
  - If we assume language is designed to minimize average description length of true sentences, the highest prob sentence has shortest desc.
  - Since we share language, we are likely to agree on desc length.

To get to MDL, we first generalize the notion of most probable hypothesis with zero error to case of non-zero empirical risk by using structural risk minimization.

- We write $\mathcal{H}$ as a countable union of **singleton** classes, $\mathcal{H} = \cup_{n \in \mathbb{N}} \{\mathcal{H}_n\}$.
- Recall Hoeffding's inequality

$$\Pr\left[\left|\frac{1}{m}\sum_{i=1}^{m} Z_i - \mu\right| > \epsilon\right] \leq 2\exp(-2m\epsilon^2).$$

- By Hoeffding's inequality, each singleton class has uniform convergence property with rate $m^{UC}_{\mathcal{H}_n}(\epsilon, \delta) = \frac{\log(2/\delta)}{2\epsilon^2}$, giving $\epsilon_n(m, \delta) = \sqrt{\frac{\log(2/\delta)}{2m}}$.

- Recall that structural risk minimization gives us

$$L_{\mathcal{D}}(h) \leq L_S(h) + \min_{n:h\in\mathcal{H}_n} \epsilon_n(m, w_n\delta).$$

- Structural risk minimization becomes

$$\arg\min_{h\in\mathcal{H}} \left[L_S(h) + \sqrt{\frac{-\log(w(h)) + \log(2/\delta)}{2m}} \right].$$

$\sqrt{\log 2/w_n\delta / 2m}$

Hoeffding

Minimum Description Length

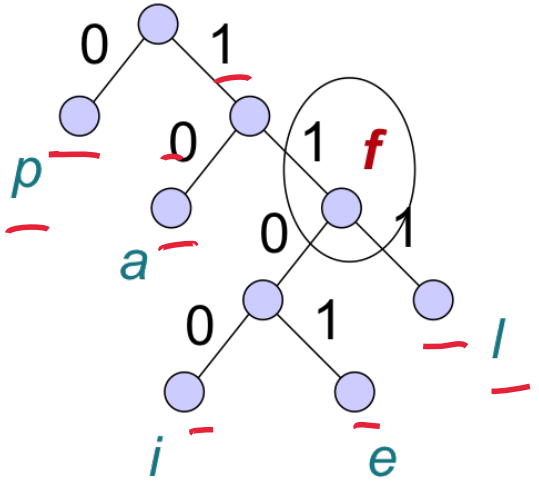
Sometimes more convenient to use description length, compared to assigning probabilities to hypotheses.
We describe a way to assign weight to hypotheses derived from the length of the description of each hypothesis.

- We assume that a language is used to describe each hypothesis. For concreteness, we assume the alphabet of the language is $\Sigma = \{0, 1\}$.
- A string is a finite sequence of symbols from Σ.
- For a string σ, denote the length as $|\sigma|$.
- The set of all strings is denoted Σ^*.
- A description is a string of symbols from Σ^*.
- Each member $h \in \mathcal{H}$ is mapped to its description by a function $d(h)$ and its length is $|h|$.

We assume that the description language is prefix-free.

In prefix free $d(h)$ satisfies

- For any two functions h and h', $d(h)$ is not a prefix of $d(h')$.
- Once we match a description, we cannot match another description by reading more symbols.
- A prefix-free set of strings can be described as a tree where each leaf forms a description.



Example: Prefix code with $a = 10$, $p = 0$, $e = 1101$, $l = 111$, $i = 1100$. No longer prefix code if $f = 11$ added.

Example: a p p l e
10 0 0 111 1101
to "apple" uniquely decode

If add f=11, no longer prefix code "appl"
1000 111 1101 decode to "appfff" or

Lemma (Kraft Inequality): If $S \subseteq \{0,1\}^*$ is a prefix free set of strings, then

$$\sum_{\sigma \in S} \frac{1}{2^{|\sigma|}} \leq 1.$$

Given a prefix-free language of a class $\mathcal{H}$, we set $w(h) = \frac{1}{2^{|h|}}$.

Theorem: (SSBD Theorem 7.7) Let $\mathcal{H}$ be a hypothesis class and let $d : \mathcal{H} \to \{0,1\}^*$ be a prefix-free description language for $\mathcal{H}$. Then, for every sample size, m, every confidence parameter, $\delta > 0$, and every probability distribution, $\mathcal{D}$, with probability greater than $1 - \delta$ over the choice of $S \sim \mathcal{D}^m$ we have that,

$$\forall h \in \mathcal{H}, L_{\mathcal{D}}(h) \leq \left[L_S(h) + \sqrt{\frac{|h| + \log(2/\delta)}{2m}} \right].$$

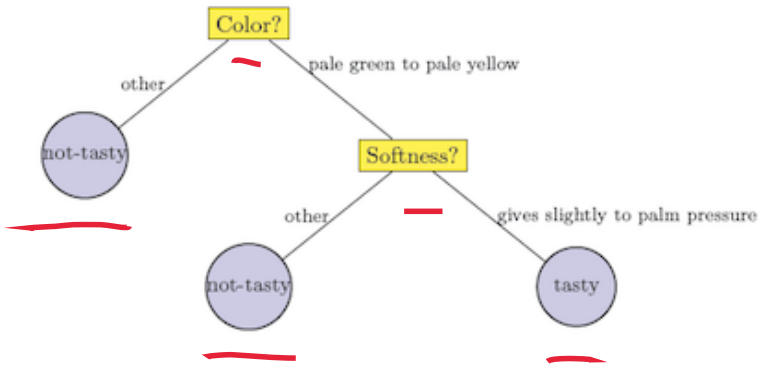
where $|h|$ is the length of $d(h)$.

Proof: Apply the previous theorem (SSBD Thm 7.4) with $\epsilon_n(m,\delta) = \sqrt{\frac{\log(2/\delta)}{2m}}$, $w(h) = 1/2^{|h|}$ and note that $\ln(2^{|h|}) = |h| \ln 2 < |h|$. □

Decision Trees

We can apply MDL to obtain a criterion to optimize for decision trees.

- For simplicity, assume $\mathcal{X} = \{0,1\}^d$, i.e. all features are binary.
- We define a prefix-free description language to describe decision trees.



Decision tree (SSBD).

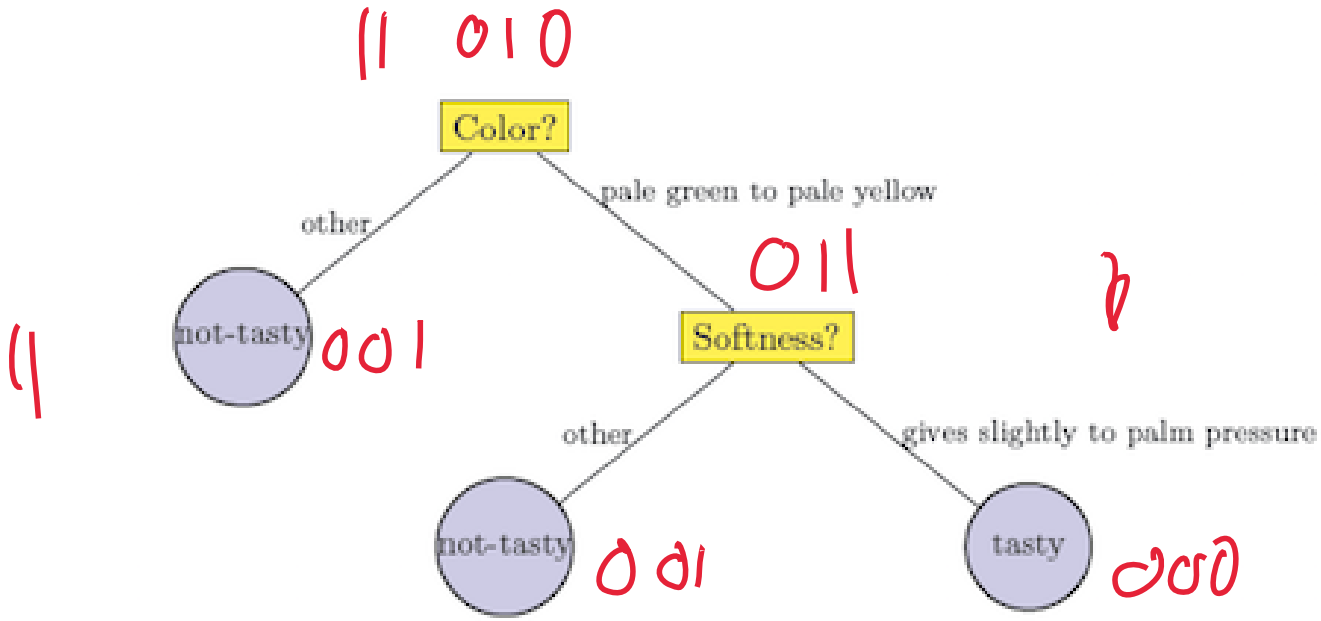
- We encode a n-node decision tree using $n+1$ blocks of length $\log(d+3)$ bits each.
- The first n blocks encode the type of node in the tree in depth-first order (preorder), while the last block encodes the end of the code.
- Each block can be one of these $d+3$ types: (1) one of d possible features as internal nodes, (2) a leaf node with value 1, (3) a leaf node with value 0, or (4) end of code.

With this code, we have with probability at least $1-\delta$, every decision tree with n nodes satisfies

$$L_D(h) \leq L_S(h) + \sqrt{\frac{(n+1)\log_2(d+3) + \log(2/\delta)}{2m}}.$$

Used as optimization criterion, it trades off the empirical risk with the description length.

Exercise 5:



Decision tree (SSBD).

Assuming that we encode *tasty* with *000*, *not-tasty* as *001*, *Color?* as *010*, *Softness?* as *011*, and *end of code* as *100*. What is the code for the tree shown?

Exercise 6:

You use a particular encoding scheme for MDL and obtain f_1 as the selected function function that minimizes the MDL bound. Your friend uses a different scheme and obtains a different function f_2 as the function that minimizes the MDL bound. Which is the function with the correct bound?

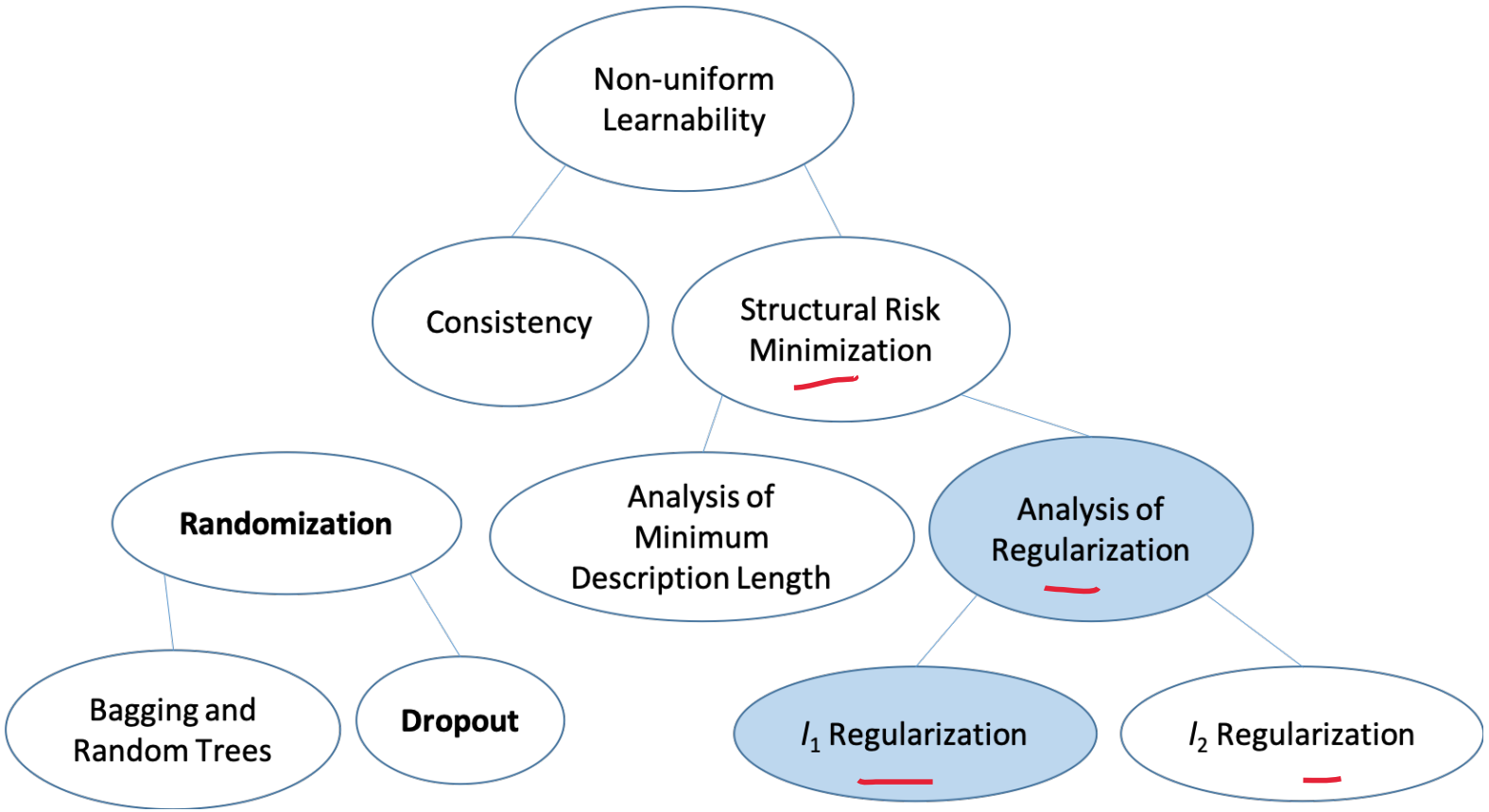
A. f_1 is the only function with the correct bound.

B. f_2 is the only function with the correct bound.

C. Both bounds would hold but with poorer confidence.

For C, use union bound for both bounds to hold.

Outline


Non-uniform Learnability
Consistency
Structural Risk Minimization
Randomization
Analysis of Minimum Description Length
Analysis of Regularization
Bagging and Random Trees
Dropout
l_1 Regularization
l_2 Regularization

In this section, we consider ℓ_1 and ℓ_2 regularization and how they relate to structural risk minimization. We also look at large margin classifiers.

ℓ_1 Regularization

Consider the function class
$\mathcal{H}_1 = \{f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} : \|\mathbf{w}\|_1 \leq 1, \mathbf{w} \in \mathbb{R}^d\}$.

$\sum_{i=1}^{d} |w_i| \leq 1$

if $\|w\|_1 \leq c$, multiply $R(H \circ S)$ by c.

First we show that the convex hull of A has the same Rademacher complexity as A.

$a_1 = [f_1(x_1), \ldots, f(m)]$
$a_2 = (f_2(x_1), \cdots, f_2(m]$

Recall the definition of the Rademacher complexity:

$$R(A) = E_{\boldsymbol{\sigma}}\left[\sup_{a \in A} \frac{1}{m}\sum_{i=1}^{m} \sigma_i a_i\right],$$

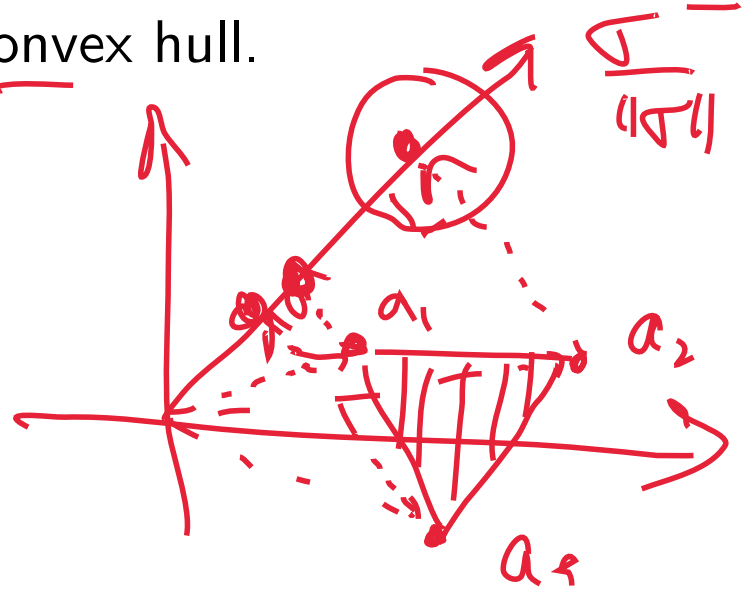
$a_d = [f_d(x_1), \ldots f_d(x_m)]$

where σ_i are i.i.d. sampled with $P(\sigma_i = 1) = P(\sigma_i = -1) = 1/2$.

Lemma: (SSBD Lemma 26.7) Let A be a subset of $\mathbb{R}^m$ and let $A' = \{\sum_{j=1}^{N} \alpha_j \mathbf{a}^{(j)} : N \in \mathbb{N}, \forall j, \mathbf{a}^{(j)} \in A,\ \alpha_j > 0\ \|\alpha\|_1 = 1\}$. Then, $R(A) = R(A')$.

Proof idea: The main idea is that the sup computation in Rademacher complexity is achieved at one of the vectors $\mathbf{a}^{(j)}$ instead of the interior of the convex hull.

$$\sup_{a \in A'} \frac{1}{m} \sum_{i=1}^{m} \sigma_i a_i$$



Using the lemma, we can bound the Rademacher complexity of $\mathcal{H}_1$ applied to a data set S.

Lemma: (SSBD Lemma 26.11) Let $S = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ be vectors in $\mathbb{R}^d$. Then

$$R(\mathcal{H}_1 \circ S) \leq \max_i \|x_i\|_\infty \sqrt{\frac{2\log(2d)}{m}}.$$

Proof in the Appendix.

$h(x) = \sum \alpha_i x_i$

$m \left\{ \begin{bmatrix} x_{11}, x_{12} \ldots x_{1d}, -x_{11}, \ldots -x_{1d} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ x_{m1}, \quad -x_{md} \end{bmatrix} \right.$

2d vectors in $\mathbb{R}^m$

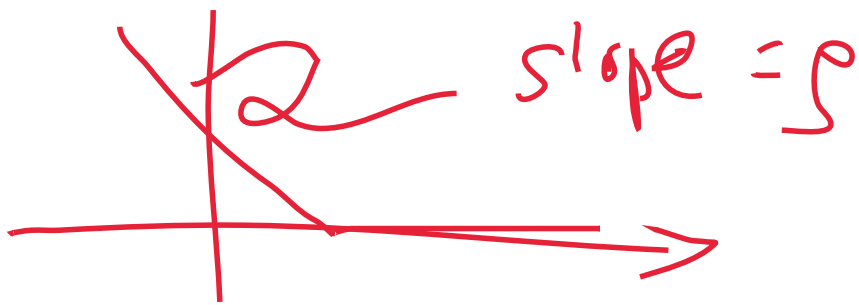
R.C. of convex hull of 2d vectors = RC of set of 2d vectors

Massart's lemma gives bound

We need to compose our function with a loss function. Assume that the loss function $\ell : \mathcal{H} \times Z \to \mathbb{R}$ is of the form

$$\ell(\mathbf{w}, (\mathbf{x}, y)) = \phi(\mathbf{w}^T \mathbf{x}, y),$$

where $\phi : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$ such that for all $y \in \mathcal{Y}$ the scalar function $a \mapsto \phi(a, y)$ is ρ-Lipschitz.



Definition (Lipschitzness): Let $C \subset \mathbb{R}^d$. A function $f : \mathbb{R}^d \to \mathbb{R}^k$ is ρ-Lipschitz over C if for every $\mathbf{w}_1, \mathbf{w}_2 \in C$ we have that $\|f(\mathbf{w}_1) - f(\mathbf{w}_2)\| \leq \rho \|\mathbf{w}_1 - \mathbf{w}_2\|$.

We define Lipschitzness with respect to the Euclidean norm, but it can also be defined with respect to other norms.

- With the hinge-loss (used e.g. in soft-margin support vector machine), the loss function $\phi(a, y) = \max\{0, 1 - ya\}$ is 1-Lipschitz with respect to $y \in \{-1, 1\}$ (purple in fig).
- The logistic loss $\phi(a, y) = \log_2(1 + e^{-ya})$ is $\frac{1}{\ln 2}$-Lipshitz for $y \in \{-1, 1\}$ as $\sup |\phi'(a)| = \sup_a \frac{1}{\ln 2} \left| \frac{-ye^{-ya}}{1+e^{-ya}} \right| = \frac{1}{\ln 2}$. (yellow in fig)
- The absolute loss $\phi(a, y) = |a - y|$ is 1-Lipshitz for all $y \in \mathbb{R}$.
- The square loss $\phi(a, y) = (a - y)^2$ is $4B$-Lipschitz when $|a|, |y| \leq B$. (green in fig)

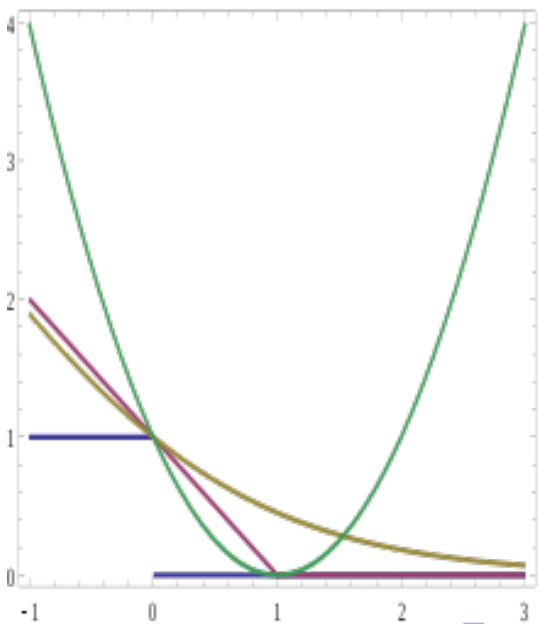


Image from Wikipedia.

The following bounds the Rademacher complexity of the composed class.

Lemma (Contraction Lemma): (SSBD Lemma 26.9) For each $i \in [m]$, let $\phi_i : \mathbb{R} \to \mathbb{R}$ be a ρ-Lipschitz function. For $\mathbf{a} \in \mathbb{R}^m$, let $\phi(\mathbf{a})$ denote the vector $(\phi_i(a_1), \ldots, \phi_m(a_m))$. Let $\phi \circ A = \{\phi(\mathbf{a}) | \mathbf{a} \in A\}$. Then

$$R(\phi \circ A) \leq \rho R(A).$$

Proof skipped.

Having obtained the Rademacher complexity, we can now bound the generalization error.

Theorem: (SSBD Theorem 26.15) Suppose that $\mathcal{D}$ is a distribution over $\mathcal{X} \times \mathcal{Y}$ such that with probability 1, we have that $\|x\|_\infty \leq R$. Let $\mathcal{H} = \{\mathbf{w} \in \mathbb{R}^d : \|\mathbf{w}\|_1 \leq B\}$ and let the loss function $\ell : \mathcal{H} \times Z \to \mathbb{R}$ be of the form $\ell(\mathbf{w}, (\mathbf{x}, y)) = \phi(\mathbf{w}^T\mathbf{x}, y)$ such that for all $y \in \mathcal{Y}$, $a \mapsto \phi(a, y)$ is a ρ-Lipschitz function and such that $\max_{a \in [-BR, BR]} |\phi(a, y)| < c$. Then, for any $\delta \in (0, 1)$, with probability of at least $1 - \delta$ over the choice of an i.i.d. sample of size m,

$$\forall h \in \mathcal{H}, \qquad L_{\mathcal{D}}(h) \leq L_S(h) + 2\rho BR\sqrt{\frac{2\log(2d)}{m}} + c\sqrt{\frac{2\ln(2/\delta)}{m}}.$$

What insights does the bound provide?

$$L_{\mathcal{D}}(h) \leq L_S(h) + 2\rho BR\sqrt{\frac{2\log(2d)}{m}} + c\sqrt{\frac{2\ln(2/\delta)}{m}}.$$

- $\sqrt{2\log(2d)}$ factors, where d is input dimension: bound grows slowly with input dimension – may work well even when very large number of features.
- Effect of B: bound small when ℓ_1 norm of weight vector small, e.g sparse functions.
  - Expect method to be useful when small number of important features, large number of irrelevant features.

- Normalization of input features can be useful.
  Example:

Case 1 : True fn is $w_1 x_1 + w_2 x_2$

$w_1 = w_2 = 1$, input $x_1, x_2 \in [0,1]$

case 2 : Same fn, but rescale x_2, now measured in km rather than meters

Now $w_1 = 1$, $w_2 = 1000$

$x_1 \in [0,1]$, $x_2 \in [0, 0.001]$

Case 2

$B = \|w\|_1 = 1001$

$R = \|x\|_\infty = 1$

$BR = 1001$

Case 1

$B = \|w\| = 2$

$R = \|x\|_\infty = 1$

$BR = 2$

Examples: The theorem applies to commonly used loss functions.

- The hinge loss $\phi(a, y) = \max\{0, 1 - ya\}$ is 1-Lipschitz $y \in \{-1, 1\}$, so $\rho = 1$ with $c \leq 1 + BR$.
- The logistic loss $\phi(a, y) = \log_2(1 + e^{-ya})$ is $\frac{1}{\ln 2}$-Lipshitz for $y \in \{-1, 1\}$, so $\rho = 1/\ln 2$ with $c \leq 1 + BR/\ln 2$.
- The absolute loss $\phi(a, y) = |a - y|$ is 1-Lipshitz for all $y \in \mathbb{R}$, so $\rho = 1$ with $c = BR$.
- With $|y| \leq BR$, the square loss $\phi(a, y) = (a - y)^2$ is $4BR$-Lipschitz, so $\rho = 4BR$ with $c \leq (2BR)^2$.

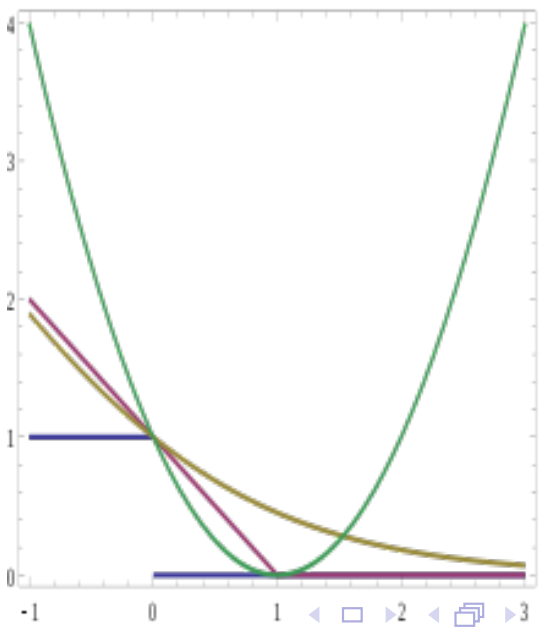
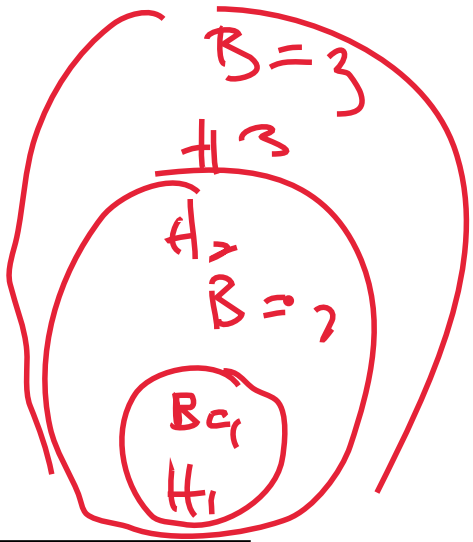


Image from Wikipedia.

Regularization and Structural Risk Minimization

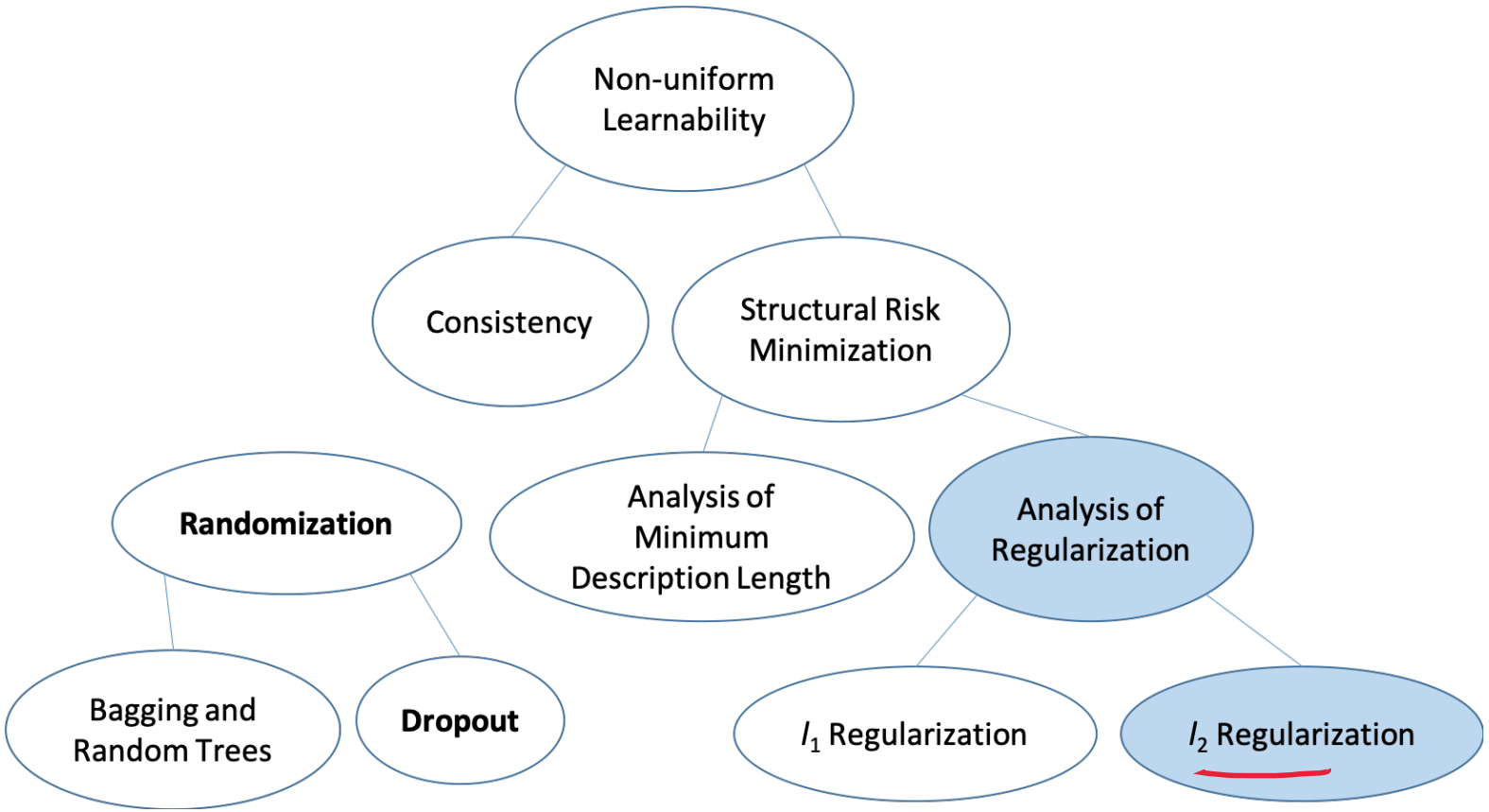
The theorem requires knowing B in advance. Can avoid that by using structural risk minimization.



- Define a sequence of classes $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \ldots$ where $\mathcal{H}_i$ contains functions with ℓ_1 norm no more than B_i for $B_1 < B_2, < \ldots$.
- Define a weighting function w_i such that $\sum_{i=1}^{\infty} w(i) \leq 1$, e.g. $w(i) = \frac{6}{\pi^2 i^2}$.
- Then we obtain

$$L_{\mathcal{D}}(h) \leq L_S(h) + 2\rho B_i R \sqrt{\frac{2\log(2d)}{m}} + c_i \sqrt{\frac{2\ln(2/w_i\delta)}{m}}$$

  for the smallest i such that $h \in \mathcal{H}_i$.
- Holds without prior knowledge of which B_i that minimizes the bound.
- In practice, we often minimize an approximation $L_S(\mathbf{w}) + \lambda\|\mathbf{w}\|_1$.


Non-uniform Learnability
Consistency
Structural Risk Minimization
Randomization
Analysis of Minimum Description Length
Analysis of Regularization
Bagging and Random Trees
Dropout
l_1 Regularization
l_2 Regularization

ℓ_2 Regularization

Consider the function class $\mathcal{H}_2 = \{f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle : \|\mathbf{w}\|_2 \leq 1\}$.

Lemma: (SSBD Lemma 26.10) Let $S = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ be vectors in a Hilbert space. Define
$\mathcal{H}_2 \circ S = \{(\langle \mathbf{w}, \mathbf{x}_1 \rangle, \ldots, \langle \mathbf{w}, \mathbf{x}_m \rangle) : \|\mathbf{w}\|_2 \leq 1\}$. Then

$$R(\mathcal{H}_2 \circ S) \leq \frac{\max_i \|\mathbf{x}_i\|_2}{\sqrt{m}}.$$

Proof in the Appendix.

If $\|w\|_2 \leq B$, multiply Rademacher complexity by B

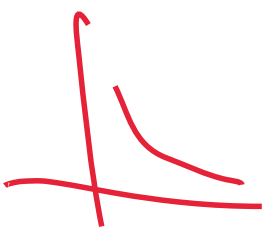
Given the bound on the Rademacher complexity, we have the following.

Theorem: (SSBD Theorem 26.12) Suppose that D is a distribution over $\mathcal{X} \times \mathcal{Y}$ such that with probability 1 we have that $\|\mathbf{x}\|_2 \leq R$. Let $\mathcal{H} = \{\mathbf{w} : \|\mathbf{w}\|_2 \leq B\}$ and let $\ell : \mathcal{H} \times Z \to \mathbb{R}$ be a loss function such that for all $y \in \mathcal{Y}$, $a \mapsto \phi(a, y)$ is a ρ-Lipschitz function and such that $\max_{a \in [-BR;BR]} |\phi(a, y)| \leq c$. Then, for any $\delta \in (0, 1)$, with probability of at least $1 - \delta$ over the choice of an i.i.d. sample of size m,

$$\forall \mathbf{w} \in \mathcal{H},\ L_{\mathcal{D}}(\mathbf{w}) \leq L_S(\mathbf{w}) + \frac{2\rho BR}{\sqrt{m}} + c\sqrt{\frac{2\ln(2/\delta)}{m}}.$$

Implications:

- The bound suggests minimizing the regularized loss function $L_S(\mathbf{w}) + \lambda \|\mathbf{w}\|_2$.
- In practice, usually optimize $L_S(\mathbf{w}) + \lambda' \|\mathbf{w}\|_2^2$. Often a nicer optimization problem and gives same solution as $L_S(\mathbf{w}) + \lambda \|\mathbf{w}\|_2$ for some value of λ'.
- Compared to ℓ_1 norm, having many small weights does not increase increase norm much.



Example: If $w_i = \frac{1}{i}$

$$\|w\|_1 = \sum_{i=1}^{\infty} \frac{1}{i} = \infty$$

$$\|w\|_2 = \sqrt{\sum_{i=1}^{\infty} \frac{1}{i^2}} = \sqrt{\frac{\pi^2}{6}}$$

- As in case of ℓ_1 norm, normalization of inputs can be helpful.

- Note that the bound depends on $\|\mathbf{w}\|$ and not on the dimension of the input space.
  - This means that the bounds also applies when we use kernels with support vector machines to map the input to a high or infinite dimensional feature space.
  - In this case the squared norm is

$$\|\mathbf{w}\|^2 = \sum_{i,j=1}^{m} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j).$$

Exercise 7

In this exercise, we will compare ℓ_1 and ℓ_2 regularization for image denoising using wavelet transform.

One way to view an image is as a function $f(i,j)$ on the pixel position (i,j). Using wavelet transform, we can represent the function (image) as $f(i,j) = \langle \mathbf{w}, \phi(i,j) \rangle$ where $\mathbf{w}$ are the linear function weights (wavelet coefficients) and $\phi(i,j)$ is the feature vector formed by the inverse wavelet transform bases. If we add noise to each pixel and try to learn a predictor (weight vector $\mathbf{w}$), we have a regression problem.

For this problem, it is simpler to think of an image as a vector $\mathbf{v}$ in $\mathbb{R}^n$, where n is the number of pixels. The i-th coefficient of the wavelet transform is then the projection of $\mathbf{v}$ on the i-th basis. Because wavelet is an orthonormal transform, it preserves distance (see background material on linear algebra), i.e. if $\bar{\mathbf{w}}$ is an estimate of $\mathbf{w}$ that gives an estimate $\bar{\mathbf{v}}$ of $\mathbf{v}$, then $\|\mathbf{v} - \bar{\mathbf{v}}\| = \|\mathbf{w} - \bar{\mathbf{w}}\|$ or $\sum_i (v_i - \bar{v}_i)^2 = \sum_i (w_i - \bar{w}_i)^2$. This makes the implementing ℓ_1 and ℓ_2 regularization easy.

For ℓ_2 regularization,
$\sum_i (v_i - \bar{v}_i)^2 + \lambda \sum_i \bar{w}_i^2 = \sum_i (w_i - \bar{w}_i)^2 + \lambda \sum_i \bar{w}_i^2$. Each coefficient $\bar{w}_i$ can be optimized independently giving the solution $\bar{w}_i = \frac{1}{1+\lambda} w_i$, or each coefficient is multiplied by a constant factor smaller than 1.

For ℓ_1 regularization,
$\sum_i (v_i - \bar{v}_i)^2 + \lambda \sum_i |\bar{w}_i| = \sum_i (w_i - \bar{w}_i)^2 + \lambda \sum_i |\bar{w}_i|$. Each coefficient can again be optimized independently, giving the soft thresholding function as the solution: if $|w_i|$ is smaller than λ, set $\bar{w}_i$ to 0, otherwise set $\bar{w}_i$ to $sign(w_i)(|w_i| - \lambda)$, i.e. reduce the size of the coefficient by λ while keeping the same sign.

Run the experiment comparing ℓ_1 and ℓ_2 regularization. Performance in signal processing is commonly measured using signal to noise ratio: $SNR(\mathbf{p}, \bar{\mathbf{p}}) = 10 \log_{10} \frac{\|\mathbf{p}\|^2}{\|\mathbf{p}-\bar{\mathbf{p}}\|^2}$ (the larger the better). Try other images such as boat.png, flowers.png and mandrill.png. Try varying the soft threshold and ridge parameters.

Which regularization method performs better for wavelet denoising? Why?

- Wavelet decomposition on images tend to be sparse
- When noise is added, small coeffs tend to be mostly noise
- When l_1-reg small coeffs are removed mostly removing noise, without increasing error much
- When l_2 reg, all coeffs are made smaller, signal values are also substantially ~~retained~~ reduced.

Large Margin Classifiers

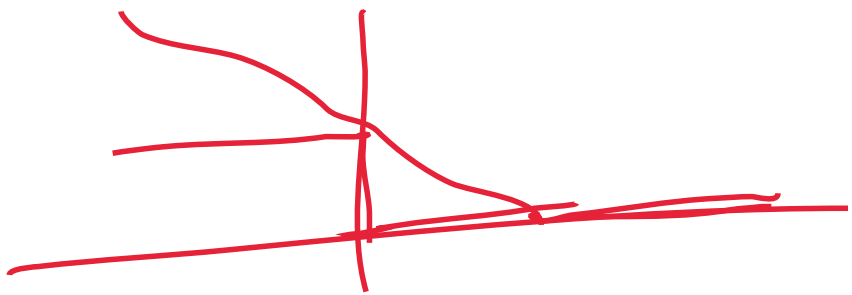
- The results do not apply when $0-1$ loss is used as it is not Lipschitz.
- Other losses can upper bound the $0-1$ loss, including hinge loss and logistic loss. In addition, the optimization problem becomes convex.

surrogate loss often convex

- The linearly separable case is insightful as we can relate the bounds to the margin.

- Assume that we have a classifier with $|f(\mathbf{x})| \geq 1$ for all $\mathbf{x} \in S$ and f classifies all the examples in S correctly.
- The hinge loss upper bounds the $0-1$ loss, hence

$$\Pr_{(\mathbf{x},y)\sim\mathcal{D}}[y \neq \text{sign}(\mathbf{w}_S^T, \mathbf{x})] \leq L_{\mathcal{D}}(w_S).$$



- The empirical risk is zero since all examples are correctly classified with magnitude at least 1, i.e. $\phi(a, y) = \max\{0, 1 - ya\} = 0$, so $L_S(h) = 0$.

- In the ℓ_1 case, we can apply SSBD Theorem 26.15:

$$L_{\mathcal{D}}(h) \leq L_S(h) + 2\rho BR\sqrt{\frac{2\log(2d)}{m}} + c\sqrt{\frac{2\ln(2/\delta)}{m}}.$$

- With the hinge loss, $\rho = 1$, and $c = BR + 1$. This gives us

$$\Pr_{(\mathbf{x},y)\sim\mathcal{D}}[y \neq \operatorname{sign}(\mathbf{w}_S^T\mathbf{x})] \leq 2BR\sqrt{\frac{2\log(2d)}{m}} + (BR+1)\sqrt{\frac{2\ln(2/\delta)}{m}}.$$

- Since $|f(\mathbf{x})| \geq 1$ and $\|\mathbf{w}\|_1 = B$, the margin (smallest magnitude on data set when $\mathbf{w}$ rescaled to $\|\mathbf{w}\|_1 = 1$) $\gamma = 1/B$, giving $B = 1/\gamma$. Substituting, we get

$$\Pr_{(\mathbf{x},y)\sim\mathcal{D}}[y \neq \operatorname{sign}(\mathbf{w}_S^T \mathbf{x})] \leq \frac{2R}{\gamma}\sqrt{\frac{2\log(2d)}{m}} + \left(\frac{R}{\gamma} + 1\right)\sqrt{\frac{2\ln(2/\delta)}{m}}.$$

- This suggests that large margin classifiers generalizes well.

- This helps to explain some puzzling observations seen with the Adaboost algorithm that iteratively adds features to the classifier.
  - As we add features to the classifier, conventional VC theory tells us to expect poorer estimation error.
  - If we keep adding features after all training examples are correctly classified, we expect overfitting to start.
  - But, in practice, it is often observed that test error keeps improving as we add features even after all training examples are correctly classified.
- Adaboost is known to increase the margin as more features are added.
  - As we add features, the increasing margin may reduce the overall complexity instead of increasing it.
  - Increasing the margin often works better than trying to reduce the number of features used.

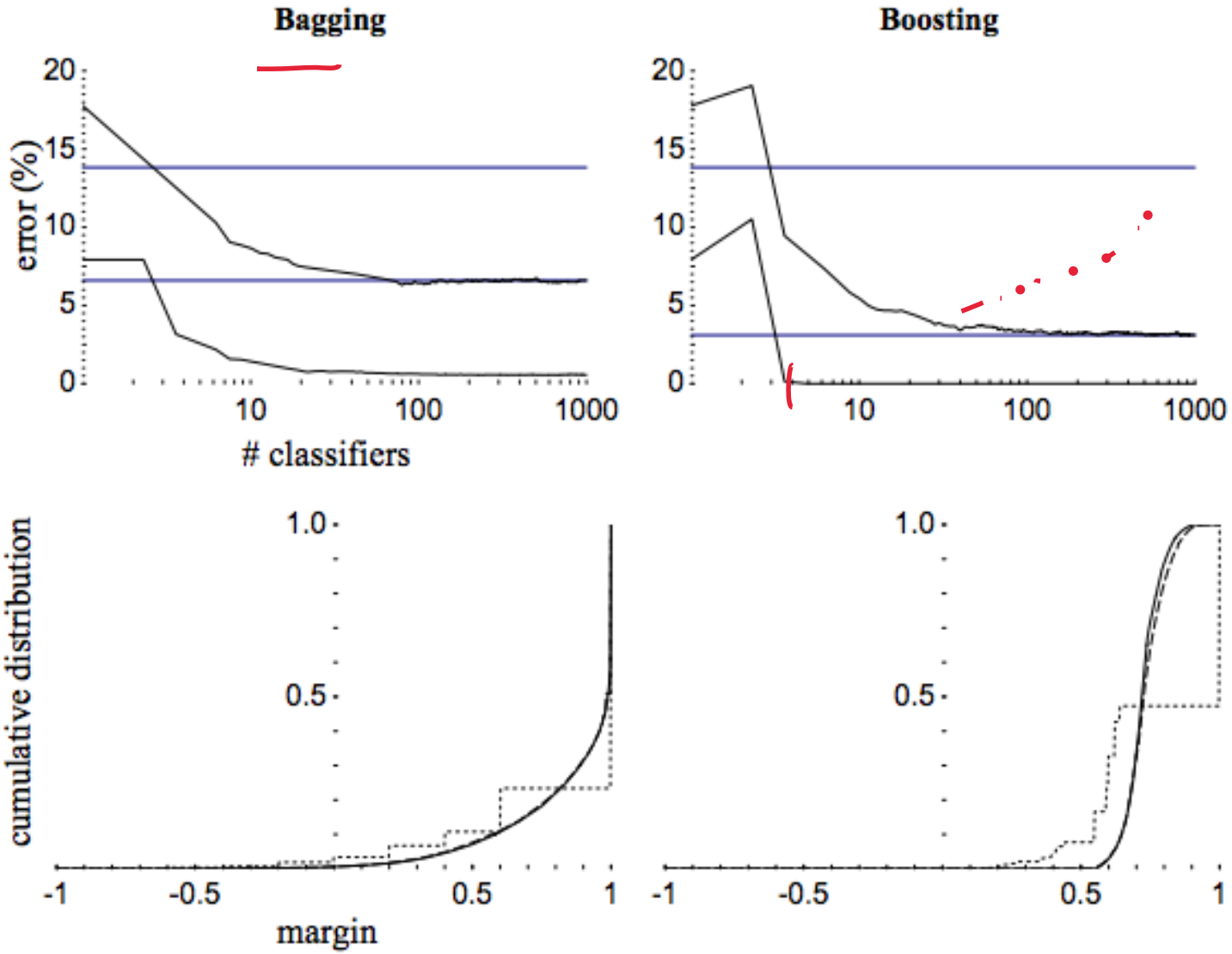


Figure from [3]. Boosting and bagging decision trees on letter dataset.

- The reasoning can be similarly applied to hard SVM.
- When the data is linearly separable, we can convert the norm-based bounds into margin based bound to show that maximizing margin while keeping $\|\mathbf{w}\| = 1$ gives good performance bound.
- (SSBD Theorem 26.13) If $\|\mathbf{x}\| \leq R$, $\|\mathbf{w}\| = 1$ and the margin is γ, then with probability at least $1 - \delta$, then the $0 - 1$ error of the hard-margin homogeneous SVM is at most

$$\frac{2R}{\gamma}\sqrt{\frac{1}{m}} + \left(\frac{R}{\gamma} + 1\right)\sqrt{\frac{2\log(2/\delta)}{m}}.$$

ℓ_1 Vs ℓ_2 Regularization

- ℓ_1 regularization usually produces sparse solution, ℓ_2 regularization usually does not. ℓ_1 may be better if
  - Solution is sparse – most variables are irrelevant.
  - Actually interested in the weights/features. In extreme cases, can recover the weights exactly (if interested, see SSBD Chapter 23.3 on compressed sensing). Demo `http://scikit-learn.org/stable/auto_examples/applications/plot_tomography_l1_reconstruction.html`.
  - In practice, prediction accuracy of ℓ_1 regularization is often better than ℓ_2 regularization on problems such as image denoising. On classification, less clear unless there is extreme sparsity, but still useful for feature selection.

- Finite ℓ_2 norm covers a larger class of functions
  - E.g. $\sum_{i=1}^{\infty} \frac{1}{i^2}$ converges, whereas $\sum_{i=1}^{\infty} \frac{1}{i}$ does not.
  - Computationally easier to solve.
  - Kernel methods provide a way to map features to very high/infinite dimensional spaces of features with finite ℓ_2 norms and operate on the functions efficiently.

Deep Neural Networks

Deep neural networks are usually trained with regularization.

We give a bound on the Rademacher complexity of deep neural networks in terms of the ℓ_1 norm of the weights of the units, although it is not clear yet what describes generalization well for deep networks in practice.

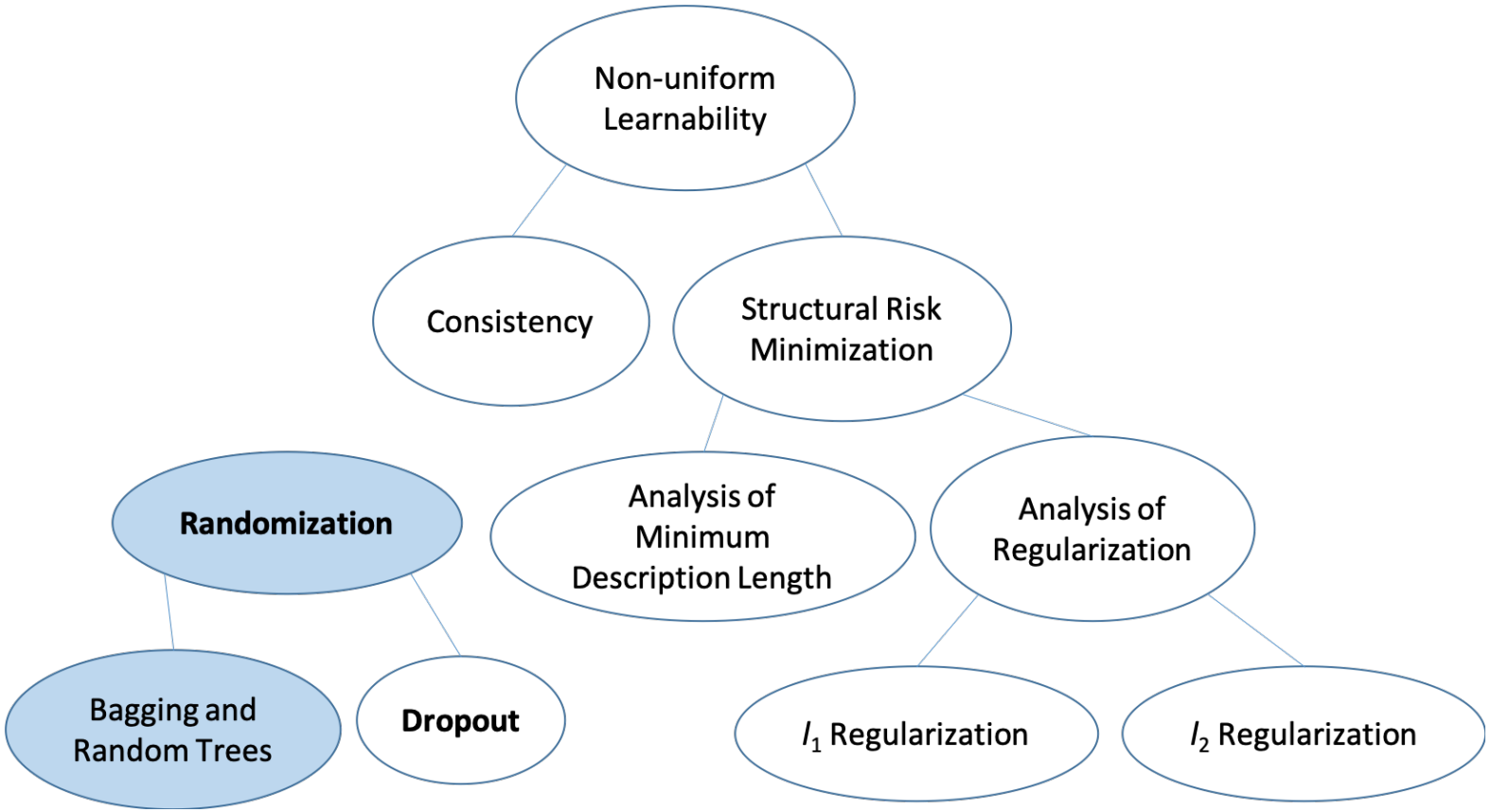
Theorem: Consider a network $\mathcal{F}$ with k hidden layers and hidden and output units $\phi(\langle \mathbf{w}, \mathbf{z} \rangle)$ where ϕ is 1-Lipschitz, $\phi(0) = 0$ and $\|\mathbf{w}\|_1 \leq B$ for all units. Suppose the input S is a subset m points from $\mathbb{R}^d$ with ℓ_∞ norm bounded by R. Then

$$R(\mathcal{F} \circ S) \leq (2B)^k BR\sqrt{\frac{2\ln(2d)}{m}}.$$

Proof in Appendix.

- This bound does not grow with the number of hidden units but with B instead. However, grows exponentially with depth k.
- The bound may be too loose to describe real behaviour well.
- Finding bounds that describe deep neural networks on real data well is an active area. For example, [1] suggests that the product of the spectral norms of the weight matrices of each layer (which approximately bounds the Rademacher complexity) correlates well with observed practical behaviour.
- With bounds that correlates well with observed behaviour, may be possible to develop algorithms that approximately optimize generalization behaviour.

Outline


Non-uniform Learnability
Consistency
Structural Risk Minimization
Randomization
Analysis of Minimum Description Length
Analysis of Regularization
Bagging and Random Trees
Dropout
l_1 Regularization
l_2 Regularization

Bias and Variance Tradeoff

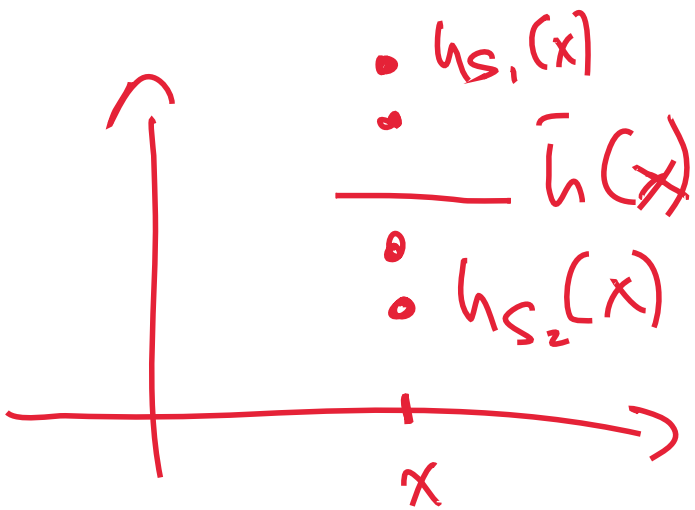
- For regression with square loss, the expected error can be decomposed into a bias and variance part
  - The bias part measures how well the average value of the learned function approximate the true function
  - The variance part measures the variance of the learned function around the average value.
  - Similar to the approximation/estimate error analysis we have been doing.

- Assume noiseless target function $f(x)$. Can be extended to $f(x) = E[y|x]$ for noisy case.
- For a function $h_S(x)$, learned from training set S, expected test error is $E_D[(f(x) - h_S(x))^2]$, where D is distribution of x.
- We are interested in the expectation over training sets S drawn from the distribution of training sets $\mathcal{D}$.

$$E_{\mathcal{D}} E_D[(f(x) - h_S(x))^2] = E_D E_{\mathcal{D}}[(f(x) - h_S(x))^2].$$

- Consider a fixed value of x and analyse $E_{\mathcal{D}}[(f(x) - h_S(x))^2]$.
  - Then average over x drawn from distribution D to get $E_D E_{\mathcal{D}}[(f(x) - h_S(x))^2]$.

- For the approximation part, we use expectation of learned function $\bar{h}(x)$
  - $\bar{h}(x) = E_{\mathcal{D}}[h_S(x)]$
  - Expectation taken over distribution of training sets $\mathcal{D}$.
  - Note that: $E_{\mathcal{D}}[h_S(x) - \bar{h}(x)] = 0$.



At x, we have

$$\begin{aligned} E_{\mathcal{D}}[(f(x) - h_S(x))^2] &= E_{\mathcal{D}}[(f(x) - \bar{h}(x) + \bar{h}(x) - h_S(x))^2] \\ &= E_{\mathcal{D}}[(f(x) - \bar{h}(x))^2] + E_{\mathcal{D}}[(\bar{h}(x) - h_S(x))^2] \\ &\quad + E_{\mathcal{D}}[(f(x) - \bar{h}(x))(\bar{h}(x) - h_S(x))] \\ &= (f(x) - \bar{h}(x))^2 + E_{\mathcal{D}}[(\bar{h}(x) - h_S(x))^2] \end{aligned}$$

Constant $\quad E[\bar{h} - h_S] = 0$

$\text{bias}(x)^2 \qquad \text{Var}(x)$

- Bias at x: $\mathrm{bias}(x)^2 = (f(x) - \bar{h}(x))^2$
- Variance at x: $\mathrm{var}(x) = E_{\mathcal{D}}[(\bar{h}(x) - h_S(x))^2]$
- Expected error over random training sets

$$\begin{aligned} E_{\mathcal{D}} E_D[(f(x) - h_S(x))^2] &= E_D E_{\mathcal{D}}[(f(x) - h_S(x))^2] \\ &= E_D[\mathrm{bias}(x)^2 + \mathrm{var}(x)] \\ &= \mathrm{bias}^2 + \mathrm{var} \end{aligned}$$

- As the hypothesis class becomes more powerful, bias tends to be smaller, but variance larger, leading to bias-variance tradeoff.

Model Averaging

Assume that we have k sets of training data. A simple method for potentially improving performance is to learn k hypotheses and then average the outputs.

- Predictor becomes $\frac{1}{k}\sum_{i=1}^{k} h_{S_i}(x)$.
- Expect $E[\frac{1}{k}\sum_{i=1}^{k} h_{S_i}(x)]$ to remain the same i.e. equals $\bar{h}(x) = E_{\mathcal{D}}[h_S(x)]$.
  - So bias $(f(x) - \bar{h}(x))$ remains the same.
- But hopefully variance $E[(\frac{1}{k}\sum_{i=1}^{k} h_{S_i}(x) - \bar{h}(x))^2]$ is reduced.

- Let $\epsilon_i = h_{S_i}(x) - \bar{h}(x)$. Then variance of the average is

$$E\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2} E\left[\left(\sum_i \epsilon_i\right)\left(\sum_j \epsilon_j\right)\right]$$
$$= \frac{1}{k^2} E\left[\sum_i \left(\epsilon_i^2 + \sum_{i \neq j} \epsilon_i \epsilon_j\right)\right]$$
$$= \frac{1}{k} v + \frac{k-1}{k} c,$$

where $v = E[\epsilon_i^2]$ is the variance and $c = E[\epsilon_i \epsilon_j]$ are the covariances (we have assumed they are the same for all i and all pairs (i, j)).

- If we use the average of the k regressor as the predictor the expected squared error is

$$\frac{1}{k}v + \frac{k-1}{k}c,$$

where $v = E[\epsilon_i^2]$ is the variance and $c = E[\epsilon_i \epsilon_j]$ are the covariances (we have assumed they are the same for all i and all pairs (i, j)).

$E[\epsilon_i \epsilon_j] = 0$ e.g. when i.i.d.

- If the errors are uncorrelated, $c = 0$, we find that the variance (expected squared error) of the combined regressor is v/k where v is the variance of a single regressor.
- If the errors are totally correlated, $c = E[\epsilon_i \epsilon_j] = E[\epsilon_i^2] = v$ and the variance of the combined regressor is the same as the variance of a single regressor.

No help!

$\frac{1}{k} v + \frac{k-1}{k} V = V$

- For classification, we can also combine the output of different classifiers. In this case, we usually select the class that gets the highest number of votes from the k classifiers.

Bagging

- Don't have k training sets. Let's create some artificially.
- In **bagging** (bootstrap aggregrating), we generate each new datasets by a process called bootstrap
  - Generate a new dataset of size m from the original dataset of size n by sampling the original dataset uniformly with replacement.
  - If $m = n$, then the sampled dataset is expected to have $(1 - 1/e) \approx 0.63$ of the examples in the original dataset, the rest being duplicates.

Bagging and Random Forest

- Compared to stable algorithms such as SVM or logistic regression, decision trees tend to classify unseen examples fairly differently when some of the training examples are changed.

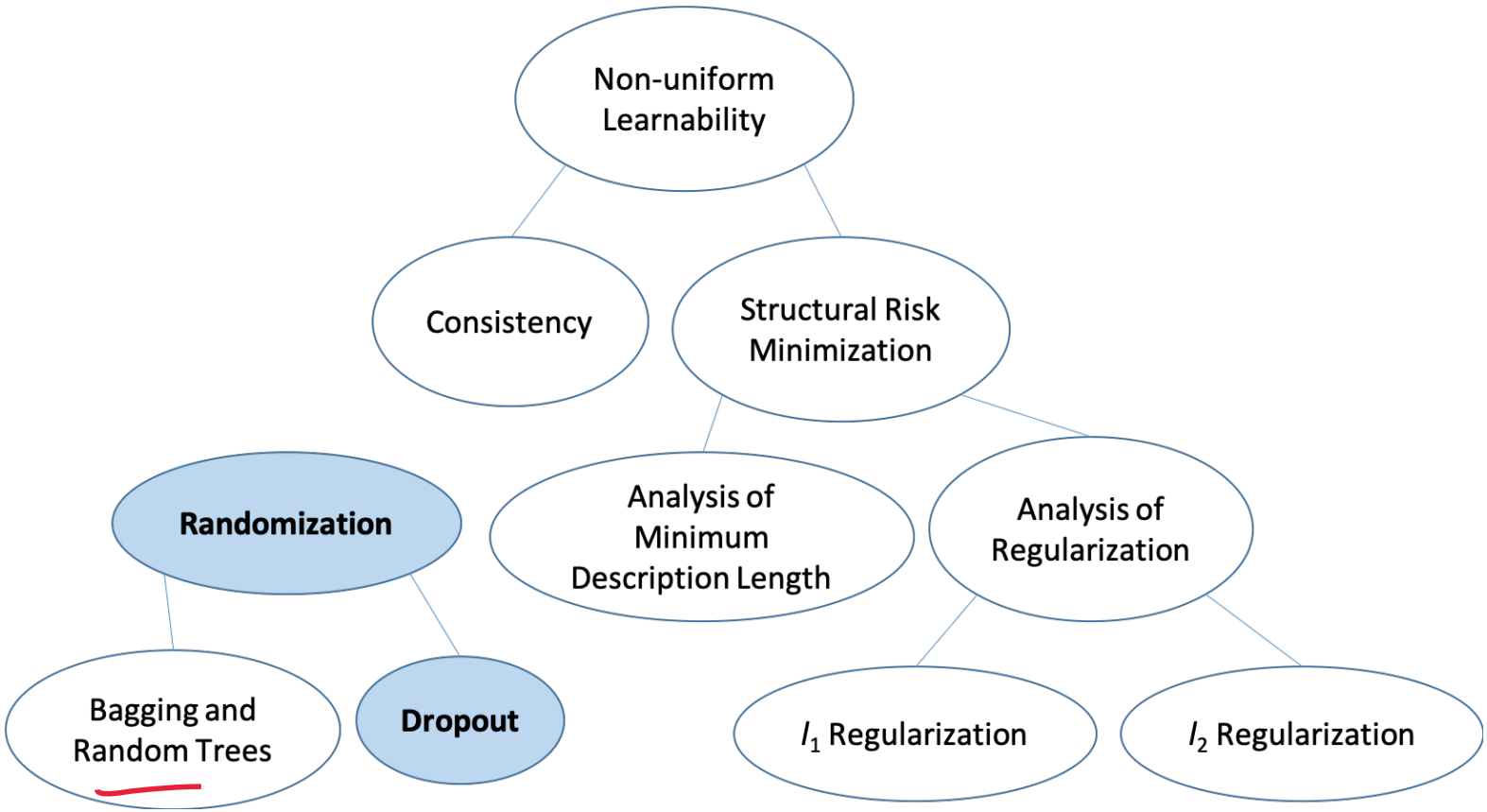
  - SVM, logistic reg tend not to find bagging helpful
  - Decision tree errors tend to be less correlated with bagging. so helpful.

- See **Exercise 8**, demo on bagging.

- Random forests uses bagging and goes further by randomly generating a subset of features to use at each node of the decision tree.
  - Decision trees are usually learned greedily by considering all possible splits of all features at the node and selecting the best one according to the greedy criterion.
  - In constructing a tree in the random forest, when each node is constructed, a subset of features of size k is randomly selected from the feature set of size d and used to do the split.

Example: features $f_1, \ldots, f_d$
At a node, sample $k=3$ features
e.g. f_3, f_7, f_{10}
Use only f_3, f_7, f_{10} to split that node.

  - Works very well in improving generalization of decision trees when used in the ensemble, and speeds up the construction of each tree as well.


Non-uniform Learnability
Consistency
Structural Risk Minimization
Randomization
Analysis of Minimum Description Length
Analysis of Regularization
Bagging and Random Trees
Dropout
l_1 Regularization
l_2 Regularization

Dropout in Neural Networks

- Neural networks are quite powerful function approximators and can be somewhat unstable.
  - Bagging tend to work quite well with neural networks.
- However, effective neural networks are often very large, which makes building multiple models expensive.
- Dropout applies randomization to obtain a similar effect within a single model. Roughly, the effect of dropout is to
  - Randomly exclude some units, including input units, to create a network then train on a subset of training example.
  - Approximately average together all these networks when training is completed.
- Unlike random forest, parameter sharing is done for all these networks (they are all part of the original larger network).

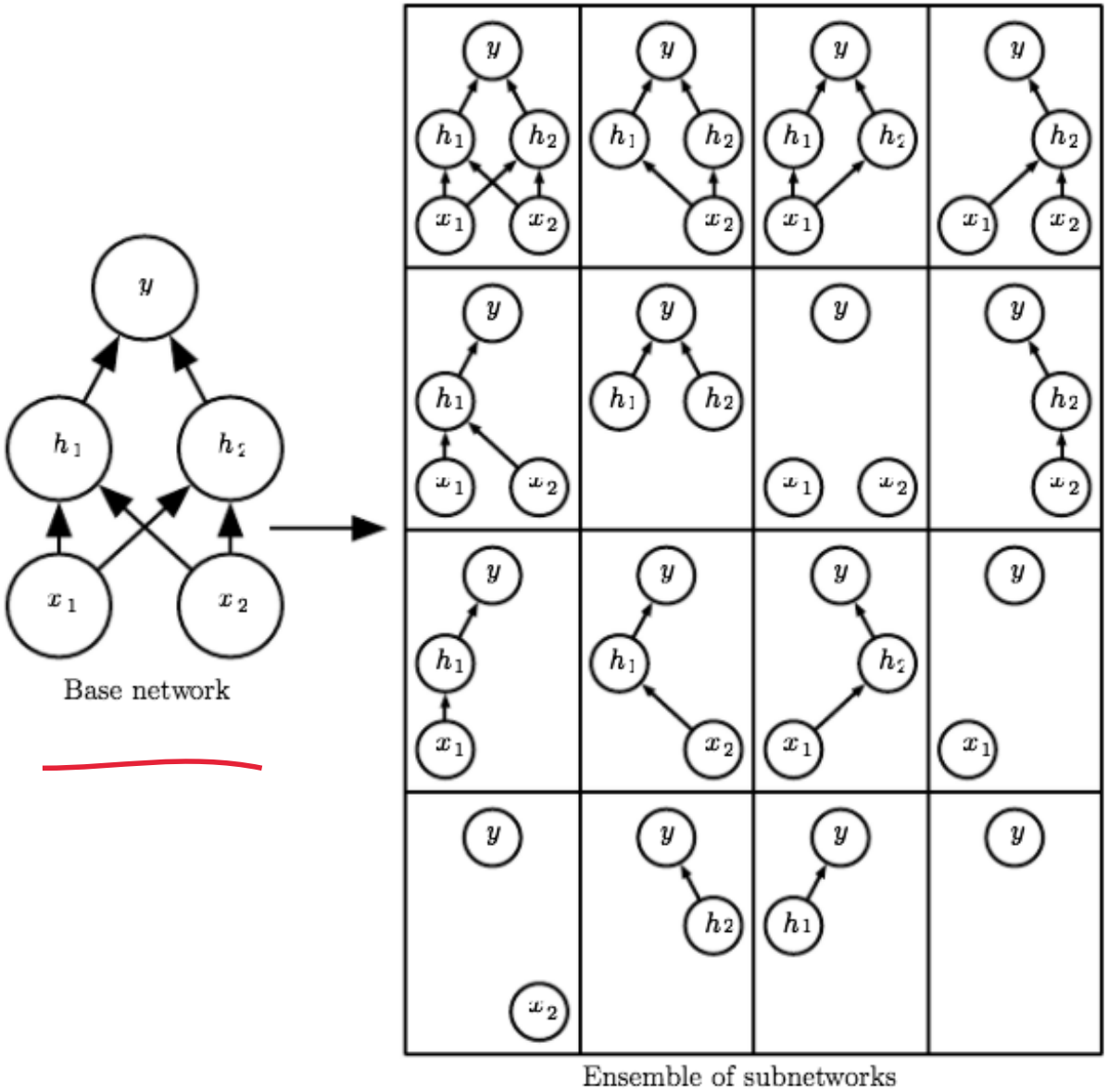


Figure from [2]. Subnetworks created by dropout.

- Neural network training is typically done with stochastic gradient descent. At each step:
  - A small subset of examples (called a mini-batch) is selected.
  - Gradient of objective function is estimated from the minibatch.
  - A step in the parameter space is taken along the (negative) direction of the gradient.
- In dropout, the following is added at each mini-batch
  - Randomly sample a binary mask.
  - Typically input units are independently included with probability 0.8 and hidden units with probability 0.5.
  - Gradient is computed with units not selected treated as if they are not there.
- For prediction, *weight scaling* is usually used: the weights going out of unit i is multiplied by the probability of including unit i.
  - This roughly approximates model averaging.

Other Methods for Improving Generalization

- For stochastic gradient descent type methods, *early stopping* is often used.
  - Training may stopped before the model that optimizes the objective function is found.
  - This has regularizing effects.
- For classification, fake data is often added, particularly those that satisfies known invariances.
  - With images, often fake data with the same label is often added by adding slightly translated, rotated and scaled images.
- Using unlabeled data to help supervised learning is called semi-supervised learning.
- Learning multiple related tasks together is often helpful as it allows some parameters to be shared among the tasks. Called multi-task learning.
- Parameter tying (sharing) is often helpful when it makes sense for the parameters to be similar.

Reading

Some material are taken directly from SSBD.

1. SSBD Chapters 7, 13, 15, 18, 26

References I

[1] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. "Spectrally-normalized margin bounds for neural networks". In: *Advances in Neural Information Processing Systems 30*. 2017.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[3] Robert E Schapire et al. "Boosting the margin: A new explanation for the effectiveness of voting methods". In: *The Annals of Statistics* 26.5 (1998), pp. 1651–1686.

Outline

Proofs for ℓ_1 Regularization

Lemma: (SSBD Lemma 26.11) Let $S = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ be vectors in $\mathbb{R}^d$. Then

$$R(\mathcal{H}_1 \circ S) \leq \max_i \|x_i\|_\infty \sqrt{\frac{2\log(2d)}{m}}.$$

Proof:

- For each $j \in [d]$, let $\mathbf{v}_j = (x_{1,j}, \ldots, x_{m,j})$.
- Let $V = \{\mathbf{v}_1, \ldots, \mathbf{v}, -\mathbf{v}_1, \ldots, -\mathbf{v}_d\}$.
- Then $\mathcal{H}_1 \circ S$ is the convex hull of V and has the same Rademacher complexity.
- From Massart's lemma,

$$R(V) \leq \max_{\mathbf{v} \in V} \|\mathbf{v} - \bar{\mathbf{v}}\|_2 \frac{\sqrt{2\log(2d)}}{m}.$$

- Noting that $\|\mathbf{v}_i - \bar{\mathbf{v}}_i\|_2 \leq \sqrt{m} \max_i \|x_i\|_\infty$ gives the result. □

Proofs for ℓ_2 Regularization

Lemma: (SSBD Lemma 26.10) Let $S = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ be vectors in a Hilbert space. Define
$\mathcal{H}_2 \circ S = \{(\langle \mathbf{w}, \mathbf{x}_1 \rangle, \ldots, \langle \mathbf{w}, \mathbf{x}_m \rangle) : \|\mathbf{w}\|_2 \leq 1\}$.

$$R(\mathcal{H}_2 \circ S) \leq \frac{\max_i \|\mathbf{x}_i\|_2}{\sqrt{m}}.$$

Proof:

- From Cauchy-Schwartz inequality, we know that $\langle \mathbf{w}, \mathbf{v} \rangle \leq \|\mathbf{w}\| \|\mathbf{v}\|$. Hence

$$
\begin{aligned}
mR(\mathcal{H}_2 \circ S) &= E_{\boldsymbol{\sigma}} \left[\sup_{\mathbf{a} \in \mathcal{H}_2 \circ S} \sum_{i=1}^{m} \sigma_i a_i \right] \\
&= E_{\boldsymbol{\sigma}} \left[\sup_{\mathbf{w}: \|\mathbf{w}\|_2 \leq 1} \sum_{i=1}^{m} \sigma_i \langle \mathbf{w}, \mathbf{x}_i \rangle \right] \\
&= E_{\boldsymbol{\sigma}} \left[\sup_{\mathbf{w}: \|\mathbf{w}\|_2 \leq 1} \langle \mathbf{w}, \sum_{i=1}^{m} \sigma_i \mathbf{x}_i \rangle \right] \\
&\leq E_{\boldsymbol{\sigma}} \left[\| \sum_{i=1}^{m} \sigma_i \mathbf{x}_i \|_2 \right]
\end{aligned}
$$

- Using Jensen's inequality, we have

$$E_{\boldsymbol{\sigma}}\left[\|\sum_{i=1}^{m}\sigma_i\mathbf{x}_i\|_2\right] = E_{\boldsymbol{\sigma}}\left[\left(\|\sum_{i=1}^{m}\sigma_i\mathbf{x}_i\|_2^2\right)^{1/2}\right] \leq \left(E_{\boldsymbol{\sigma}}\left[\|\sum_{i=1}^{m}\sigma_i\mathbf{x}_i\|_2^2\right]\right)^{1/2}$$

- Finally, since the variables $\sigma_1, \ldots, \sigma_m$ are independent, we have

$$\begin{aligned} E_{\boldsymbol{\sigma}}\left[\|\sum_{i=1}^{m}\sigma_i\mathbf{x}_i\|_2^2\right] &= E_{\boldsymbol{\sigma}}\left[\sum_{i,j}\sigma_i\sigma_j\langle\mathbf{x}_i,\mathbf{x}_j\rangle\right] \\ &= \sum_{i\neq j}\langle\mathbf{x}_i,\mathbf{x}_j\rangle E_{\boldsymbol{\sigma}}[\sigma_i]E_{\boldsymbol{\sigma}}[\sigma_j] + \sum_{i=1}^{m}\langle\mathbf{x}_i,\mathbf{x}_i\rangle E_{\boldsymbol{\sigma}}[\sigma_i^2] \\ &= \sum_{i=1}^{m}\|\mathbf{x}_i\|_2^2 \leq m\max_i\|\mathbf{x}_i\|_2^2. \end{aligned}$$

Putting them together gives the result. □

Rademacher Complexity of Deep Neural Networks

Theorem: Consider a network $\mathcal{F}$ with k hidden layers and hidden units $\phi(\langle \mathbf{w}, \mathbf{z} \rangle)$ where ϕ is 1-Lipschitz, $\phi(0) = 0$ and $\|\mathbf{w}\|_1 \leq B$ for all units. Suppose the input S is a subset m points from $\mathbb{R}^d$ with ℓ_∞ norm bounded by R. Then

$$R(\mathcal{F} \circ S) \leq (2B)^k BR\sqrt{\frac{2\ln(2d)}{m}}.$$

Proof:

- The case $k = 0$ has previously been shown when we looked at linear function with $\|\mathbf{w}\|_1 \leq B$.
- Assume that the statement hold for networks with i hidden layers.
- Let $\mathcal{F}_i$ denote the class of functions represented by networks with i hidden layers.

- We first note that $\mathcal{F}_i$ contains the zero function, as $\phi(0) = 0$ and we can set the output layer weight to zero. Let $conv(\mathcal{H})$ denote the convex hull of $\mathcal{H}$. Note that a linear function with $\|\mathbf{w}\|_1 = B$ gives B times the convex hull of $\mathcal{F}_i \cup -\mathcal{F}_i$. Then

$$\begin{aligned}
R(\mathcal{F}_{i+1} \circ S) &\leq R(Bconv(\mathcal{F}_i \cup -\mathcal{F}_i) \circ S), \qquad \phi \text{ is 1-Lipschitz} \\
&= BR(conv(\mathcal{F}_i \cup -\mathcal{F}_i) \circ S) \\
&= BR((\mathcal{F}_i \cup -\mathcal{F}_i) \circ S) \\
&= 2BR(\mathcal{F}_i \circ S) \\
&\leq 2B(2B)^i BR\sqrt{\frac{2\ln(2d)}{m}} \\
&= (2B)^{i+1} BR\sqrt{\frac{2\ln(2d)}{m}}.
\end{aligned}$$

The 4-th line is because

$$\sup_{f \in \mathcal{F}_i \cup -\mathcal{F}_i} \sum_j \sigma_j f(x_j) \leq \sup_{f \in \mathcal{F}_i} \sum_j \sigma_j f(x_j) + \sup_{f \in -\mathcal{F}_i} \sum_j \sigma_j f(x_j)$$

as $\mathcal{F}_i$ contains the zero function.