Double Descent, continued

Let's discuss some key steps in Hastie et al.

Fact 2: $R(\hat{\beta}, \beta) = B(\hat{\beta}, \beta) + V(\hat{\beta}, \beta)$

where

$$B(\hat{\beta}, \beta) = \| \mathbb{E}[\hat{\beta} | X] - \beta \|_\Sigma^2$$

$\nwarrow \|b\|_\Sigma^2 = b^T \Sigma b$

$$V(\hat{\beta}, \beta) = \mathrm{Tr}\left[\mathrm{cov}(\hat{\beta} | X) \Sigma \right]$$

Moreover the bias and variance terms have useful closed-form expressions

Proposition [folklore]

$$B(\hat{\beta}, \beta) = \beta^T \Pi \Sigma \Pi \beta$$

$$V(\hat{\beta}, \beta) = \frac{\sigma^2}{n} \mathrm{Tr}(\hat{\Sigma}^+ \Sigma)$$

where $\hat{\Sigma} = \frac{X^T X}{n}$ uncentered sample covariance

$\Pi = I - \hat{\Sigma}^+ \hat{\Sigma}$ projection onto nullspace of X

Proof: Let's compute the conditional expectation and covariance of $\hat{B}$, conditioned on the samples

$$\hat{B} \cong \text{min euclidean norm soln to } Xb = y$$

$$= (X^T X)^+ X^T (XB + \varepsilon)$$

Thus we have

$$\mathbb{E}[\hat{B} \mid X] = (X^T X)^+ X^T X B$$
$$= \hat{\Sigma}^+ \hat{\Sigma} B$$

And furthermore let's compute

$$\hat{B}\hat{B}^T = (X^T X)^+ X^T (XB+\varepsilon)(XB+\varepsilon)^T X (X^T X)^+$$

And thus we have

$$\begin{aligned}\mathrm{cov}(\hat{\beta}|X) &= (X^TX)^+ X^T \mathrm{cov}(\varepsilon\varepsilon^T) X (X^TX)^+ \\ &= \sigma^2 (X^TX)^+ = \frac{\sigma^2 \hat{\Sigma}^+}{n}\end{aligned}$$

All that remains is to plug these into the expressions in Fact 2

$$\begin{aligned}B(\hat{\beta}, \beta) &= \| \mathbb{E}[\hat{\beta}|X] - \beta \|_\Sigma^2 \\ &= \| (\hat{\Sigma}^+ \hat{\Sigma} - I)\beta \|_\Sigma^2 \\ &= \| \Pi \beta \|_\Sigma^2 = \beta^T \Pi \Sigma \Pi \beta\end{aligned}$$

And for the variance

$$\begin{aligned}V(\hat{\beta}, \beta) &= \mathrm{Tr}[\mathrm{cov}(\hat{\beta}|X)\Sigma] \\ &= \frac{\sigma^2}{n} \mathrm{Tr}[\hat{\Sigma}^+ \Sigma]\end{aligned}$$

Let's specialize to Gaussians with $\Sigma = I$ and sketch the calculations

Fact 3: X and XU have the same distribution, for any orthogonal U

Now we can compute

$$B(\hat{\beta}, \beta) = \beta^T \Pi \beta$$

$$= \beta^T (I - (X^T X)^+ (X^T X)) \beta$$

$$= \beta^T (I - U^T (X^T X)^+ \underbrace{U U^T}_{I} (X^T X) U) \beta$$

$$= r^2 - \underbrace{(U\beta)^T}_{re_i^T} (X^T X)^+ (X^T X) \underbrace{(U\beta)}_{re_i}$$

Now choose U s.t. $U\beta = re_i$

(e_i: i^{th} standard basis vector)

and averaging over $i = 1$ to p we get

$$B(\hat{\beta}, \beta) = r^2 \, \mathbb{E}\left[1 - \frac{\mathrm{Tr}\left((X^T X)^+ X^T X\right)}{p}\right]$$

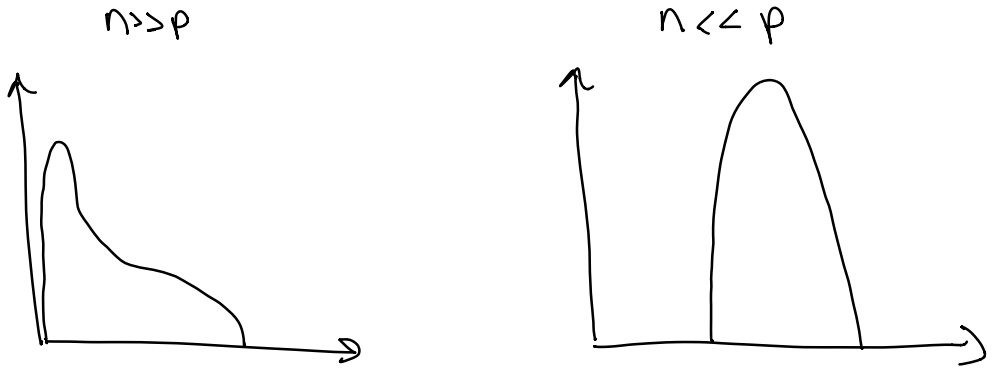
Since $(X^T X)^+ (X^T X)$ is a projection onto an n-dimensional subspace, we have

$$B(\hat{\beta}, \beta) = r^2 \left(1 - \frac{n}{p}\right) = r^2 \left(1 - \frac{1}{\gamma}\right)$$

The rest of the computations come from

(1) The eigenvalues of $\underset{p \times p}{\frac{X^T X}{n}}$ follow Marcenko-Pasteur distribution

For example



(2) The moments have nice combinatorial expressions, can even get other kinds of functionals

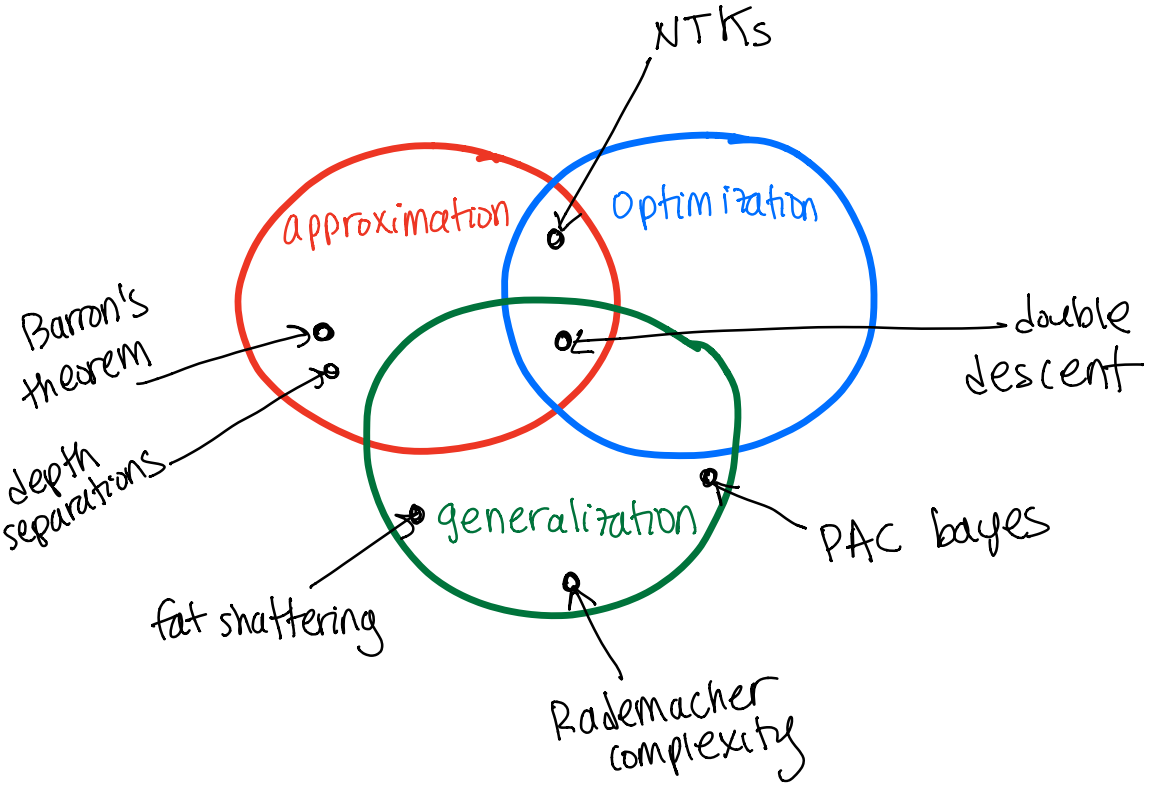
$$\underbrace{\mathrm{Tr}\left(\left(\frac{X^TX}{n} - zI\right)^{-1}\right)}_{\text{stieltjes transform}} = \sum_i \frac{1}{\lambda_i - z}$$

λ_i: eigenvalues of $\frac{X^TX}{n}$

Again, closed-form expressions are known and can evaluate as $z \to 0$

Hardness

So far we have studied:



Main Question: Are there interesting, non-linear concept classes (related to deep nets) that provably learn?

i.e. polynomial sample complexity, efficient algorithms, provable generalization

We'll start with lower bounds

Theorem [Blum, Rivest] There is an n-input neural net with depth two, three nodes and threshold activations that is NP-hard to learn

Problem Setup

(1) given a collection of training data

(2) the architecture is fixed

It is NP-hard to decide if there is a setting of weights that lets the network fit perfectly

Is this a convincing lower bound? Or could there be ways around it?

def: In proper learning, the algorithm is required to output a hypothesis in the class $h \in \mathcal{H}$

In contrast, in improper learning we're allowed to output any hypothesis

$$h : \mathbb{R}^d \to \{\pm 1\}$$

provided it generalizes:

$$\text{err}_D(h) \triangleq \mathbb{P}_{(x,y)\sim D}[h(x) \neq y] \leq \underbrace{\text{OPT}}_{\text{best agreement in } \mathcal{H}} + \varepsilon$$

It turns out that hardness of proper learning does not imply hardness of improper learning

In fact, there are some simple, old counterexamples

def: A 3-CNF (conjunctive normal form) is Boolean formula

$$\bigwedge_{i=1}^{m} (L_{i_1} \vee L_{i_2} \vee L_{i_3})$$

each L_i is literal (either x_j or $\overline{x_j}$)

Similarly

def: A k-term DNF (disjunctive normal form) is a Boolean formula

$$\bigvee_{i=1}^{k} \left(\bigwedge_{j=1}^{n_k} L_{ij} \right)$$

Theorem [Pitt, Valiant] Unless RP = NP, there is no polynomial time algorithm for proper learning a k-term DNF for $k \geq 3$

In contrast:

Theorem [Valiant] There is a poly. time algorithm for improperly learning

k-term DNFs

The intuition behind these results comes from the following:

Fact 4: k-CNFs $\supsetneq$ k-term DNFs

Proof: we will prove this for $k = 3$

Let T_1, T_2 and T_3 be the terms in the DNF. Then

$$T_1 \vee T_2 \vee T_3 = \bigwedge_{\substack{L_1 \in T_1 \\ L_2 \in T_2,\, L_3 \in T_3}} (L_1 \vee L_2 \vee L_3)$$

To convince yourself of this identity, consider

($\Longrightarrow$) Suppose wlog T_1 is true

Then for any $L_1 \in T_1$, it must be true and so every clause in the CNF that includes it is true too

($\Leftarrow$) Suppose T_1, T_2 and T_3 are false.

Then there is a choice of $L_i \in T_i$ s.t. each L_i is false. This term in the CNF is false

Notice that the CNF we get in the transformation has a special structure:

"Every clause has one literal from each T_i"

Pitt and Valiant exploit this structure to encode a 3-coloring problem

But why are 3-term DNFs improperly learnable?

The idea is to learn them as arbitrary 3-CNFs

In particular if you take the complement of a 3-CNF

$$\bigwedge_{i=1}^{m} (L_{i_1} \vee L_{i_2} \vee L_{i_3})$$

you get a 3-DNF

$$\bigvee_{i=1}^{m} (\bar{L}_{i_1} \wedge \bar{L}_{i_2} \wedge \bar{L}_{i_3}) = \overline{\Phi}$$

Now the algorithm is

(1) start with all size three conjunctions

(2) whenever you see an example that does not fit

i.e. $h(x) = 1 \neq y = 0$

take every clause that is true,
and remove it

This algorithm is called winnow

The intuition why it works boils down to:

Observation: Winnow never removes a clause that is Φ

This implies

Lemma 1: It always outputs a hypothesis that perfectly fits the training data

And finally, why does it generalize?

Lemma 2: It outputs a hypothesis from H', where $|H'| \leq 2^{8n^3}$

Thus we can apply the generalization bounds we proved earlier

Main Takeaway: Don't believe hardness results for proper learning

Caveat: Usually need to substantially enrich the class you are learning from, to the point where you get much worse sample complexity

These are called computational vs. statistical tradeoffs

Thus our new question is:

Can we improperly learn deep nets?

Theorem [Klivans, Sherstov] For every constant $\varepsilon > 0$, it is cryptographically hard to (improper) learn depth two nets with n^{ε} units

Without taking too much of a detour:

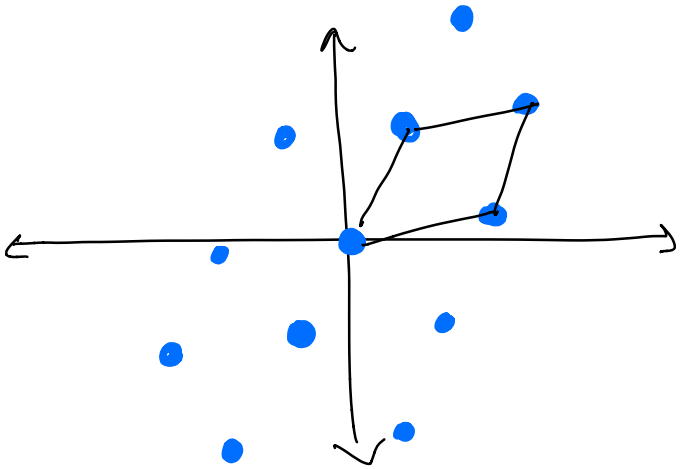
Cryptographic hardness

(1) Someone built a cryptosystem assuming the problem is hard

(2) It's not NP-hard

(think: factoring)

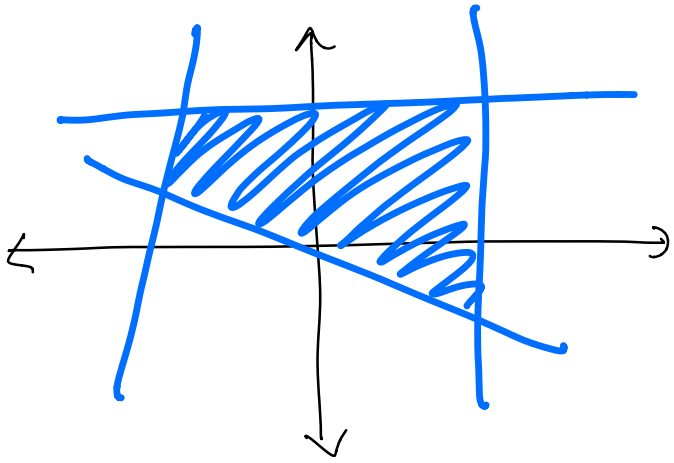
Klivans and Sherstov used the hardness of the shortest vector problem (SVP) in a lattice



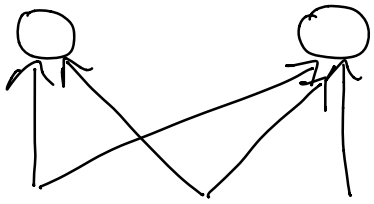
i.e. a discrete subset closed under integer linear combinations

In high-dimensions, best known algorithm can approximate SVP within c^d, and Klivans and Sherstov assume approximating SVP with $d^{3/2}$ is hard.

Actually they show hardness for learning intersections of n^ε halfspaces



How do intersections of halfspaces and depth two networks relate?



hidden node $\longleftrightarrow$ halfspace

In particular, you can set it up so

Output of hidden node	label of halfspace
0	$+1$
<0	-1

Then just add the outputs of hidden nodes and take the sgn (where 0 maps +1)

The intuition for hardness builds on a key idea of Kearns and Valiant

Theorem (informal) In a public key encryption system, a class of hypotheses that contains the decryption function is hard to improperly learn

In particular, an attacker can generate his own data as:

$(x = Enc(m), y = Dec(x) = m)$

And if you can PAC learn the decryption function (even improperly) then you can break the security of the scheme by able to predict the decryption of new strings better than random

Klivans and Sherstov use a cryptoscheme of Regev, whose security is based on SVP, whose decoding function can be implemented as an intersection of n^c halfspaces

Okay, so should we believe these lower bounds? Or is there a way around them?

Food for thought: How natural is the distribution on examples?

We can't learn deep nets from feeding in encryptions of random messages

Generative Models for Data

We'll study some fundamental learning problems on simple distributions

Next time: Learning intersections of halfspaces, and SQ lower bounds on Gaussian inputs