Recap

Boosting reduces training error:

$$\hat{R}_S(h_T) \leq \prod_t 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \quad (A)$$

$$= \prod_t \sqrt{1-4\gamma_t^2} \quad (B) \quad \left(\varepsilon_t = \frac{1}{2} - \gamma_t\right)$$

$$\gamma_t > 0$$

$$\leq e^{-2\sum_t \gamma_t^2} \quad (C)$$

(B) $\Rightarrow$ (C)

Proof:

$$\varepsilon_t = \frac{1}{2} - \gamma_t$$

Want to prove using induction:

$$D_{t+1}(i) = \frac{e^{-y_i h_t(x_i)}}{Z_{t+1}} \quad (P)$$

$$Z_{t+1} = \sum_{i=1}^{m} e^{-y_i h_t(x_i)}$$

$$D_{t+1}(i) = D_t(i) \times \frac{e^{-w_t f_t(x_i)}}{Z_{t+1}}$$

$$\left(w_t = \frac{1}{2}\log\left(\frac{1}{\varepsilon_t} - 1\right)\right) \qquad h_t = \sum_{s \leq t} w_t f_t$$

1) Prove for $t = 1$

2) Use relationship D_{t+1} & D_t to show that (P) is true if (P) is true for t.

Insight: At each t, WL rule focuses on hard examples from iteration $t-1$

Boosting generalization error: (informal)
with high probability,

$$R(h_T) \leq \hat{R}_S(h_T) + C\sqrt{\frac{Td}{m}}$$

d: VC dimension of weak hypothesis space
(another measure of function complexity)

h_T: o/p of Boosting after T iterations

$x \in \mathbb{R}^d$ (d: i/p dimension)

$m = |S|$

→ Tradeoff between Bias-Complexity (approximation + estimation error)

→ $h_t \in \mathcal{H}_t$ → more complex with t

→ Approx error: depends on $\mathcal{H}_t$ estimation: depends on success of ERM

Larger Question

Exact interpolation / Memorization / Overfitting : Zero/"low" training error $\hat{R}_S(h)$

h is learned by solving ERM over some $\mathcal{H}$:

(i) Boosting: (composition of nonlinearity with $\mathcal{H}_{WL}$)

(ii) SVM (linear): Halfspaces

(iii) Regression: linear prediction class affine

(iv) Deep Neural Network

In practice, Care about: how well does h do on unseen data?

→ Generalization within distribution ✓
$\underset{S \sim \mathcal{D}^m}{\mathbb{E}} \hat{R}_S(h)$

→ OOD generalization *
(e.g. adversarial learning)
distributionally robust optimization

$\underset{z \sim \mathcal{D}_e}{\mathbb{E}} \ell(z, h)$

Overparametrization:
$\dim(w) >> \dim(\mathcal{X}) = d$
or increasing function complexity
→ ↑ DNN: width (# of neurons in a layer), depth (# of layers)
→ ↑ width of 2 layer network

In principle,
generalization gap $< f\left(\frac{\text{complexity}}{m}\right)$
increasing function

$= |R(h) - \hat{R}_S(h)|$

→ if $\hat{R}_S(h) \to 0$ (overfitting)
→ If you overparameterize, you should not gain!

Mystery: But, why do we gain in practice?

- inductive biases (prior knowledge in S that reduces the # of fns being searched over in $\mathcal{H}$)
- regularization
  e.g. sparsity-inducing ℓ^1 regularization leading to tighter generalization bounds in linear regression
  implicit regularization from early stopping
- what is the role of optimization algorithm?
  e.g. SGD prefers "flat minima" ⇒ better generalization
- Margin theory
  e.g. Boosting increases margin even after training error is zero

Boosting the margins

Shapire Freund

$$\rho(h;x) = y\,h(x) \quad \text{(confidence margin)}$$

With high probability,

$$R(h) \leq \underbrace{\hat{R}_{S,\rho}(h)}_{\text{pushed to } 0} + \frac{C}{\rho}\sqrt{\frac{d}{m}}$$

Boosting increases margin ρ.

offsets $\uparrow$ in complexity

Kernels

Belkin et al 2018

"Generalization despite overparameterization" ??

May be we need to understand answer for Kernels to answer the same question for DNNs.

$\mathcal{X} \subseteq \mathbb{R}^d$ (domain of inputs)

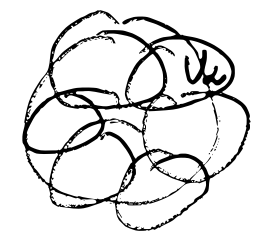
$\mathcal{X}$: Compact subset of $\mathbb{R}^d$.

Detour:

$\mathcal{F}$ is a vector space if
$\rightarrow f, g \in \mathcal{F}$, $f+g \in \mathcal{F}$ (closed under addition)
$\rightarrow af \in \mathcal{F}$ $\forall a \in \mathbb{R}$

$U \subseteq \mathcal{F}$ is compact if every open cover has a finite subcover

if U_α is an open cover: $U \subseteq \underbrace{\bigcup_\alpha U_\alpha}_{\text{cover}}$ U_α open sets



then $\exists \{\alpha_i\}_{i=1}^m$ s.t.

$U \subseteq \bigcup_{\alpha_i} U_{\alpha_i}$

(Topological definition)

Sequential compactness:
$U \subseteq \mathcal{F}$ is compact if every infinite sequence in U has a converging subsequence.

$\mathcal{X}$: compact set in $\mathbb{R}^d$
($\equiv$ closed and bounded set)

Kernel
k is a function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$.

Smooth kernels e.g.

1) $k(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}}$
(Gaussian kernel)

2) non-smooth kernel
$k(x, x') = e^{-\frac{\|x-x'\|}{\sigma^2}}$
(Laplace kernel)

3) $k(x, x') = (\langle x, x' \rangle + c)^n$
(n^{th} order polynomial kernel)

(Kevin Murphy — kernels)
In ML, a type of kernels called "Mercel kernels" are used

$\rightarrow$ Positive definite kernels
$\rightarrow$ Reproducing kernel $\boxed{\text{Hilbert spaces}}$ (RKHS)
$\rightarrow$ Kernel methods: ERMs on RKHS

Introduction to kernel methods

Example: (Mohri et al)

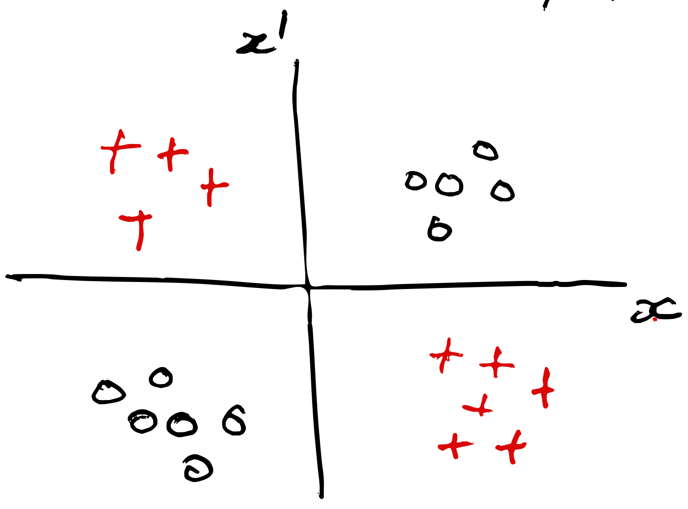
True: XOR $\quad x, x'$

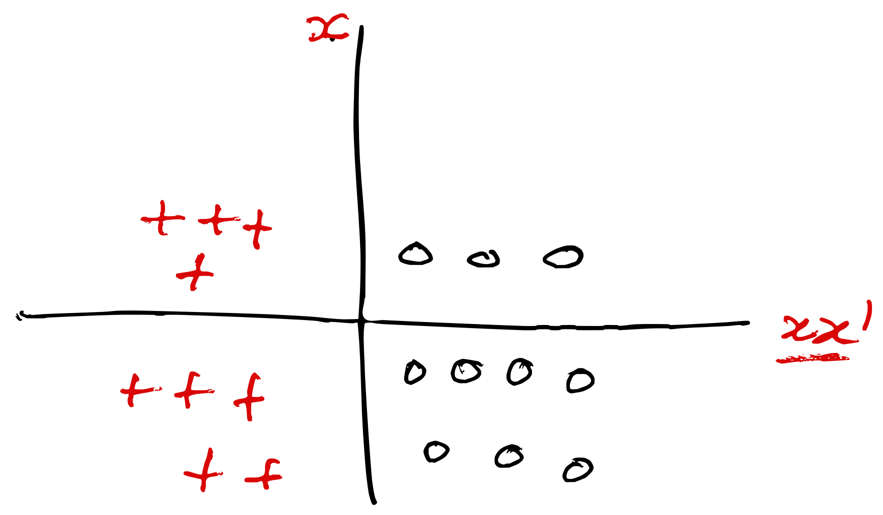
$$h_{XOR}(x, x') = \begin{cases} 0, & x, x' \geq 0 \text{ or } x, x' \leq 0 \\ 1, & (x > 0 \;\&\; x' < 0) \text{ or } (x < 0 \;\&\; x' > 0) \end{cases}$$

Linear class:

$Ax + Bx' + C$

ERM: Linear regression to learn A, B, C





Linear regression on

$xx' \in$ polynomials of order 2.

For a PDS kernel

$k(x, x') = \langle f_x(\cdot), f_{x'}(\cdot) \rangle_{\mathcal{H}}$ RKHS

e.g. $e^{-\frac{\|x - x'\|^2}{2\sigma^2}}$

Recall soft-SVM solution

$$h(x) = \text{sgn}\left(\left\langle \sum_{i=1}^{m} \alpha_i y_i x_i, x \right\rangle\right)$$

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \xrightarrow{\text{replace}} \begin{bmatrix} f_{x_1} \\ \vdots \\ f_{x_m} \end{bmatrix}$$

$X X^T \to K$ (Gram matrix)

Then, $h(x) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i k(x, x_i) \right)$

Symmetric: $k(x, x') = k(x', x)$

"Kernelized" SVM. Advantages of kernelizing:

→ Inner products in infinite-dimensional space can be computed simply by evaluating k

↳ No need to explicitly specify the features

→ Solving ERMs on RKHS
→ reduced to finite-dimensional optimization problems.