Learning when there is no perfect hypothesis

- Hoeffding/Chernoff bounds: minimizing training error will approximately minimize true error: just need $O(1/\varepsilon^2)$ samples versus $O(1/\varepsilon)$.
- What about polynomial-time algorithms? Seems harder.
  - Given data set $S$, finding apx best conjunction is NP-hard.
  - Can do other things, like minimize hinge-loss, maxent type loss, but not directly connected to error rate.
- One way to make progress: make assumptions on the "noise" in the data. E.g., Random Classification Noise model.

Learning from Random Classification Noise

- PAC model, target $f \in C$, but assume labels from noisy channel.
- "noisy" Oracle $EX^\eta(f,D)$, $\eta$ is the noise rate.
  - Example $x$ is drawn from $D$.
  - With probability $1-\eta$ see label $l(x) = f(x)$.
  - With probability $\eta$ see label $l(x) = 1-f(x)$.
- E.g., if $h$ has non-noisy error $p$, what is the noisy error rate?
  - $p(1-\eta) + (1-p)\eta = \eta + p(1-2\eta)$.

Algorithm A PAC-learns $C$ from random classification noise if for any $f \in C$, any distrib $D$, any $\eta < 1/2$, any $\varepsilon, \delta > 0$, given access to $EX^\eta(f,D)$, $A$ finds a hyp $h$ that is $\varepsilon$-close to $f$, with probability $\geq 1-\delta$.

Allowed time poly$(1/\varepsilon, 1/\delta, 1/(1-2\eta), n, \text{size}(f))$

- Q: is this a plausible goal? We are asking the learner to get closer to $f$ than the data is.
- A: OK because noisy error rate is linear in true error rate (squashed by $1-2\eta$)

Notation

- Use "Pr[...]"] for probability with respect to non-noisy distribution.
- Use "Pr$_\eta$[...]"] for probability with respect to noisy distribution.

Learning OR-functions (assume monotone)

- Let's assume noise rate $\eta$ is known. Any ideas?
- Say $p_i = \Pr[f(x) = 0 \land x_i = 1]$
- Any $h$ that includes all $x_i$ such that $p_i = 0$ and no $x_i$ such that $p_i > \varepsilon/n$ is good.
- So, just need to estimate $p_i$ to $\pm \varepsilon/n$.
  - Rewrite as $p_i = \Pr[f(x) = 0|x_i = 1] \times \Pr[x_i = 1]$.
  - 2nd part unaffected by noise (and if tiny, can ignore $x_i$). Define $q_i$ as 1st part.
  - Then $\Pr_h[l(x) = 0|x_i = 1] = q_i(1-\eta) + (1-q_i)\eta = \eta + q_i(1-2\eta)$.
  - So, enough to approx LHS to $\pm O((\varepsilon/n)(1-2\eta))$. 
Learning OR-functions (assume monotone)
• If noise rate not known, can estimate with smallest value of \( \Pr_{\eta}[\eta(x)=0|x_i=1] \).

\[ \begin{array}{c|c|c}
0 & 0 & 1 \\
0 & 1 & \eta \\
1 & 1 & 1 \\
\end{array} \]

Generalizing the algorithm
Basic idea of algorithm was:
• See how can learn in non-noisy model by asking about probabilities of certain events with some "slop".
• Try to learn in noisy model by breaking events into:
  - Parts predictably affected by noise.
  - Parts unaffected by noise.

Let's formalize this in notion of "statistical query" (SQ) algorithm. Will see how to convert any SQ alg to work with noise.

The Statistical Query Model
• No noise.
• Algorithm asks: "what is the probability a labeled example will have property \( \chi \)? Please tell me up to additive error \( \tau \)."

- Formally, \( \chi: X \times \{0,1\} \rightarrow \{0,1\} \). Must be poly-time computable. \( \tau \geq 1/poly(\ldots) \).
- Let \( P_{\chi} = \Pr[\chi(x,f(x))=1] \).
- World responds with \( P_{\chi} \in [P_{\chi}-\tau, P_{\chi}+\tau] \).

[can extend to \([0,1]\)-valued or vector-valued \( \chi \)]

• May repeat poly(\ldots) times. Can also ask for unlabeled data. Must output \( h \) of error \( \leq \varepsilon \).
  No \( \delta \) in this model.

The Statistical Query Model
• Examples of queries:
  - What is the probability that \( x_i=1 \) and label is negative?
  - What is the error rate of my current hypothesis \( h \)? \( \Pr \chi(x,f(x))=1 \iff h(x) \neq f(x) \)

• Get back answer to \( \pm \tau \). Can simulate from \( \approx 1/\tau^2 \) examples. [That's why need \( \tau \geq 1/poly(\ldots) \).]
• To learn OR-functions, ask for \( \Pr[x_i=1 \land f(x)=0] \) with \( \tau = \varepsilon/(2n) \). Produce OR of all \( x_i \) such that \( P_{\chi} \leq \varepsilon/(2n) \).

The Statistical Query Model
• Many algorithms can be simulated with statistical queries:
  - Perceptron: ask for \( \mathbb{E}[f(x)x : h(x) \neq f(x)] \) (formally define vector-valued \( \chi = f(x)x \) if \( h(x) = f(x) \), and 0 otherwise. Then divide by \( \Pr[h(x)=f(x)] \).
  - Hill-climbing type algorithms: what is error rate of \( h \)? What would it be if I made this tweak?

• Properties of SQ model:
  - Can automatically convert to work in presence of classification noise.
  - Can give a nice characterization of what can and cannot be learned in it.

SQ-learnable \( \Rightarrow \) (PAC+Noise)-learnable
• Given query \( \chi \), need to estimate from noisy data. Idea:

- Break into part predictably affected by noise, and part unaffected.
- Estimate these parts separately.
- Can draw fresh examples for each query or estimate many queries from same sample if VCDim of query space is small.

• Running example: \( \chi(x,\ell) = 1 \iff x_i=1 \land \ell=0 \).
How to estimate $\Pr[\chi(x, f(x)) = 1]$?

- Let $\text{CLEAN} = \{ x : \chi(x, 0) = \chi(x, 1) \}$
- Let $\text{NOISY} = \{ x : \chi(x, 0) \neq \chi(x, 1) \}$
  - What are these for $\chi(x, l) = 1$ iff $x_i = 1$ \& $l = 0$ ?
- Now we can write:
  - $\Pr[\chi(x, f(x)) = 1] = \Pr[\chi(x, f(x)) = 1 \land x \in \text{CLEAN}] + \Pr[\chi(x, f(x)) = 1 \land x \in \text{NOISY}]$.

Step 1: first part is easy to estimate from noisy data (easy to tell if $x \in \text{CLEAN}$).
- What about the 2nd part?

How to estimate $\Pr[\chi(x, f(x)) = 1]$?

- Let $\text{CLEAN} = \{ x : \chi(x, 0) = \chi(x, 1) \}$
- Let $\text{NOISY} = \{ x : \chi(x, 0) \neq \chi(x, 1) \}$
  - What are these for $\chi(x, l) = 1$ iff $x_i = 1$ \& $l = 0$ ?
- Now we can write:
  - $\Pr[\chi(x, f(x)) = 1] = \Pr[\chi(x, f(x)) = 1 \land x \in \text{CLEAN}] + \Pr[\chi(x, f(x)) = 1 \land x \in \text{NOISY}]$.

Can estimate $\Pr[x \in \text{NOISY}]$.
- Also estimate $P_\eta \equiv \Pr[\chi(x, l) = 1 \mid x \in \text{NOISY}]$.
- Want $P \equiv \Pr[\chi(x, f(x)) = 1 \mid x \in \text{NOISY}]$.
- Write $P_\eta = P(1-\eta) \ast (1-P)\eta = \eta + P(1-2\eta)$.
- So, $P = (P_\eta - \eta) / (1-2\eta)$.
  - Just need to estimate $P_\eta$ to additive error $(1-2\eta)$.
  - If don’t know $\eta$, can have “guess and check” wrapper around entire algorithm.

Characterizing what’s learnable using SQ algorithms

- Key tool: Fourier analysis of boolean functions.
- Sounds scary but it’s a cool idea!
- Let’s think of functions from $\{0,1\}^n \rightarrow \{-1,1\}$.
- View function $f$ as a vector of $2^n$ entries: $(D[000]^1/2 f(000), D[001]^1/2 f(001), \ldots, D[x]^1/2 f(x), \ldots)$
- What is $\langle f, f \rangle$? What is $\langle f, g \rangle$?
- What is an orthonormal basis? Will see connection to SQ algos next time...