Week 8: Noise and Error

Table of Contents

Noise and Probablistic Target

1. Three types of noise
   - Noise in y: Incorrect labeling (e.g. true mis-labeled as false in training data point)
   - Noise in y: Same input value chi with different values of y
   - Noise in x: Input value x is incorrect
2. VC bound accounting for noise in training set
   - Derivation of VC bound in the ideal case assumes that, given x $P(x)$, the probability of y given by target function f being different from that predicted by hypothesis h, or $\|f(x) \neq h(x)$, is *deterministic*
     - Because target function $f(x)$, all by itself, is deterministic
   - When noise is present in traning data, the relationship becomes **probablistic**, due to combination of deterministic target function and random noise

$$\|y \neq h(x)\| \text{ with } y \sim P(y \mid x)$$

   - VC bound **remains valid**, so long as all x and y involved are **i.i.d.**
     - This guarantees that the probabily P can be *estimated* based on available training data
     - In other words, VC holds for

$$\underbrace{x \overset{i.i.d.}{\sim} P(x), y \overset{i.i.d.}{\sim} P(y \mid x)}_{\text{(x,y)} \overset{i.i.d.}{\sim} \text{P(x,y)}}$$

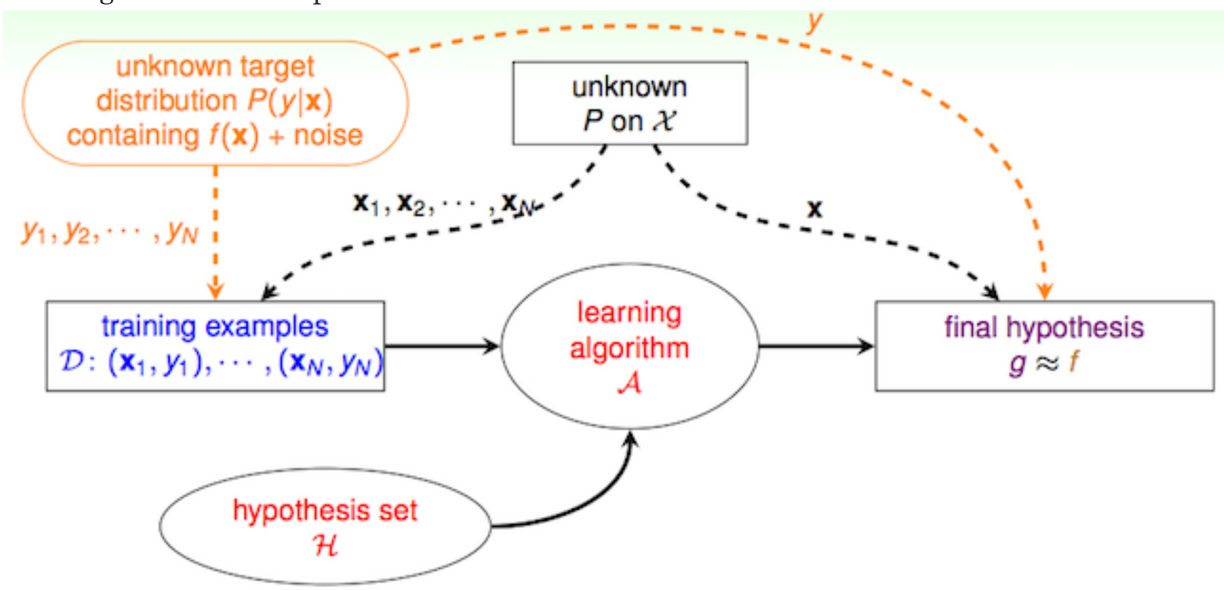
3. Target distribution
   - **Target distribution**, $P(y|x)$, characterizes the behavior of *mini-target* on one x
   - Can be viewed as 'ideal mini-target' + noise, e.g.

$$\left.\begin{array}{r} \text{idea mini-target} \quad f(x) = 1 \\ \text{'flipping' noise } level = 0.3 \end{array}\right\} \Rightarrow \begin{array}{l} P(+1 \mid x) = 0.7 \\ P(-1 \mid x) = 0.3 \end{array}$$

   - Deterministic target f is just a special case of target distribution where the conditional probabilities are 1 and 0, respectively

$$P(y \mid x) = 1 \text{ for } y = f(x)$$
$$P(y \mid x) = 0 \text{ for } y \neq f(x)$$

4. Learning when noise is present
    - Goal: Predict **ideal mini-target** (w.r.t. $P(y \mid x)$), on **often-seen inputs** (w.r.t. P(x))
        - Out-of-sample error driven by often-seen inputs
        - Inproving E_in (while guaranteeing small probability of bad sample) on often-seen inputs improves performance of the model on populatin overall, assuming data is i.i.d.
    - Learning flow with noise present



Error Measure

1. The effectiveness of a model g in predicting the underlying target distribution f is measured by **error measure**, $E(g, f)$
2. Natural choices for error measure:
    - *Out-of-sample*: Averaged over unknown x $E_{out}(g) = \epsilon_{x \sim P} \| g(x) \neq f(x) \|$
    - *Pointwise*: Evaluate error on one specific x
    - *Classification*: Incidents where prediction differs from target
        - Also callsed **0/1 error**, as it's often used to evaluate binary classification models
3. Pointwise error measure

    - Error measure if often expressed as average error across the data set, $E(g, f) =$ averaged $err(g(x), f(x))$, or

$$E_{out}(g) = \epsilon_{x \sim P} \underbrace{\| g(x) \neq f(x) \|}_{\text{err(g\{x\},f(x))}}$$

where err is the *pointwise error measure*

- **in-sample**

  $$E_{in}(g) = \frac{1}{N}\sum_{n=1}^{N} \text{err}(g(\mathbf{x}_n), f(\mathbf{x}_n))$$

  out-of-sample

  $$E_{out}(g) = \underset{\mathbf{x}\sim P}{\mathcal{E}} \text{err}(g(\mathbf{x}), f(\mathbf{x}))$$

- Two important pointwise error measures

$$err(\underbrace{g(x)}_{\hat{y}}, \underbrace{f(x)}_{y})$$

  - **0/1 error**: $err(\hat{y}, y) = \|\hat{y} \neq y\|$
    - Is prediction made by the model correct or incorrect?
    - Classification
  - **Squared error**: $err(\hat{y}, y) = (\hat{y} - y)^2$
    - In absolute measure, how far is $\hat{y}$ from y ?
    - Regression

4. Ideal mini-target
   - Interplay between *noise*, $P(y|x)$ and *error*, *err*, defines mini-target $f(x)$
     - Achieved by *optimizing error* (according to optimization target), while *accounting for noise*
     - Depending on error measure, the ideal mini-target could be different even when probabilistic noise is the same

$$P(y=1|\mathbf{x}) = 0.2, P(y=2|\mathbf{x}) = 0.7, P(y=3|\mathbf{x}) = 0.1$$

$$\text{err}(\tilde{y}, y) = [\![\tilde{y} \neq y]\!]$$

$$\tilde{y} = \begin{cases} 1 & \text{avg. err } 0.8 \\ 2 & \text{avg. err } 0.3(*) \\ 3 & \text{avg. err } 0.9 \\ 1.9 & \text{avg. err } 1.0 \text{(really? :-))} \end{cases}$$

$$f(\mathbf{x}) = \underset{y\in\mathcal{Y}}{\text{argmax}}\, P(y|\mathbf{x})$$

$$\text{err}(\tilde{y}, y) = (\tilde{y} - y)^2$$

$$\begin{cases} 1 & \text{avg. err } 1.1 \\ 2 & \text{avg. err } 0.3 \\ 3 & \text{avg. err } 1.5 \\ 1.9 & \text{avg. err } 0.29(*) \end{cases}$$

$$f(\mathbf{x}) = \sum_{y\in\mathcal{Y}} y \cdot P(y|\mathbf{x})$$

Algorithmic Error Measure

1. Depending on the context where a model is used, **errors might not cost the same**
   - *err* is **application/user-dependent**

two types of error: false accept and false reject

		g: +1	g: -1
f	+1	no error	false reject
	-1	false accept	no error

		g: +1	g: -1
f	+1	0	10
	-1	1	0

- supermarket: fingerprint for discount
- false reject: **very unhappy customer, lose future business**
- false accept: give away a minor discount, intruder left fingerprint :-)

two types of error: false accept and false reject

		g: +1	g: -1
f	+1	no error	false reject
	-1	false accept	no error

		g: +1	g: -1
f	+1	0	1
	-1	1000	0

- CIA: fingerprint for entrance
- false accept: **very serious consequences!**
- false reject: unhappy employee, but so what? :-)

2. The true error measure err is not always possible to find
   - Might not be name the exact cost of each error
   - Use **algorithmic error measure** ($\widehat{err}$) as best-effort approximate
3. Choices of algorithmic error measure
   - Plausible (well-defined, can be used with learning algorithm)
     - 0/1
       - Minimizing "flipping noise"
       - NP-hard to optimize
     - Square
       - Minimum Gaussian noise
   - Friendly (easily optimize in learning algorithm (A))
     - Closed-form solution
     - Convex objective function

Weighted Classification

1. **Weighted classification**: Depending on the use case, give different weight/importance to each data point
2. Considering the above CIA example, its weight classification errors can be expressed as follows:

out-of-sample

$$E_{\text{out}}(h) = \underset{(\mathbf{x},y)\sim P}{\mathcal{E}} \left\{ \begin{array}{ll} 1 & \text{if } y = +1 \\ 1000 & \text{if } y = -1 \end{array} \right\} \cdot [\![y \neq h(\mathbf{x})]\!]$$

in-sample

$$E_{\text{in}}(h) = \frac{1}{N} \sum_{n=1}^{N} \left\{ \begin{array}{ll} 1 & \text{if } y_n = +1 \\ 1000 & \text{if } y_n = -1 \end{array} \right\} \cdot [\![y_n \neq h(\mathbf{x}_n)]\!]$$

3. Minimize in-sample error for weighted classification

- PLA: Works as is and converges if sample set is linear separable
- Pocket: Requires modification to pocket weight replacement rule
    - If w_{t+1} reaches smaller **weighted** in-sample error E_{in}^{w} than current pocket weight $\hat{w}$, replace $\hat{w}$ by w_{t+1}

4. Proof of Pocket algorithm guarantee on E_{in}^{w}

- Account for difference in weight by copying the *more costly* samples n times

original problem

		$h(\mathbf{x})$ +1	$h(\mathbf{x})$ -1
y	+1	0	1
	-1	1000	0

$$\mathcal{D}: \begin{array}{c} (\mathbf{x}_1, +1) \\ (\mathbf{x}_2, -1) \\ (\mathbf{x}_3, -1) \\ \ldots \\ (\mathbf{x}_{N-1}, +1) \\ (\mathbf{x}_N, +1) \end{array}$$

equivalent problem

		$h(\mathbf{x})$ +1	$h(\mathbf{x})$ -1
y	+1	0	1
	-1	1	0

$$\begin{array}{c} (\mathbf{x}_1, +1) \\ (\mathbf{x}_2, -1), (\mathbf{x}_2, -1), \ldots, (\mathbf{x}_2, -1) \\ (\mathbf{x}_3, -1), (\mathbf{x}_3, -1), \ldots, (\mathbf{x}_3, -1) \\ \ldots \\ (\mathbf{x}_{N-1}, +1) \\ (\mathbf{x}_N, +1) \end{array}$$

- Using *"virtual copying"*, **weighted pocket algorithm** includes:
    - Weighted PLA
        - Randomly picking samples, only that the more costly samples are now n times more likely to be chosen
    - Weighted pocket replacement