Validation and Testing

COMPSCI 371D — Machine Learning

Outline

Training and Testing

- Empirical risk is average loss over training set:
  $L_T(h) \stackrel{\text{def}}{=} \frac{1}{|T|} \sum_{(\mathbf{x},y)\in T} \ell(y, h(\mathbf{x}))$
- Training is Empirical Risk Minimization:
  $\text{ERM}_T(\mathcal{H}) \in \arg\min_{h\in\mathcal{H}} L_T(h)$
  (A fitting problem)
- **Not enough for machine learning: Must *generalize***
- **Small risk on "previously unseen data"**
- How do we know? Evaluate on a separate *test set* S
- This is called *testing* the predictor
- How do we know that S and T are "related?"

Model Selection

- Hyper-parameters: Degree k for polynomials, number k of neighbors in k-NN
- How to choose? Why not just include with parameters, and train?
- Difficulty 0: k-NN has no training! No big deal
- Difficulty 1: $k \in \mathbb{N}$, while $\mathbf{v} \in \mathbb{R}^m$ for some predictors. Hybrid optimization. Medium deal, just technical difficulty
- **Difficulty 2: *Answer from training would be trivial!***
- Can always achieve minimal risk on T
- So k must be chosen separately from training. **It tunes generalization**
- This is what makes it a *hyper-parameter*
- Choosing hyper-parameters is called *model selection*
- Evaluate choices on a separate *validation set* V

Model Selection, Training, Testing

- *Warning: We use "model" with two different meanings in the same slide deck!* [Sorry, that's the literature]
- "Model" in "model selection" is $\mathcal{H}$
- Given a (hyper-)parametric family of hypothesis spaces, model selection selects one particular member of the family
- Given a specific hypothesis space (hyper-parameter), training selects one particular predictor out of it
- Use V to select model, T to train, S to test
- Train on cats and test on horses?
- V, T, S are mutually disjoint but "related"
- What does "related" mean?

A Generative Data Model

- What does "related" mean?
- *Every* sample $(\mathbf{x}, y)$ comes from a joint probability distribution $p(\mathbf{x}, y)$, the "data model" ("model" number 2)
- True for training, validation, and test data, and for data seen during deployment
- For the latter, y is "out there" but unknown
- The goal of machine learning:
  - Define the *(statistical) risk* $L_p(h) = \mathbb{E}_p[\ell(y, h(\mathbf{x}))] = \iint \ell(y, h(\mathbf{x}))p(\mathbf{x}, y)d\mathbf{x}dy$
  - Learning performs *(Statistical) Risk Minimization*: $\text{RM}_p(\mathcal{H}) \in \arg\min_{h\in\mathcal{H}} L_p(h)$
- *Lowest risk on* $\mathcal{H}$: $L_p(\mathcal{H}) \stackrel{\text{def}}{=} \min_{h\in\mathcal{H}} L_p(h)$

p is Unknown

- $\mathrm{RM}_p(\mathcal{H}) \in \arg\min_{h\in\mathcal{H}} \iint \ell(y, h(\mathbf{x}))p(\mathbf{x}, y)d\mathbf{x}dy$
- So, we don't need training data anymore?
- **We typically do not know $p(\mathbf{x}, y)$**
- $\mathbf{x} =$ image? Or sentence?
- Can we not estimate p?
- The curse of dimensionality, again
- We typically cannot find $\mathrm{RM}_p(\mathcal{H})$ or $L_p(\mathcal{H})$
- That's the goal all the same

So Why Talk About It?

- Why talk about $p(\mathbf{x}, y)$ if we cannot know it?
- $L_p(h)$ is a mean, and we can *estimate* means
- We can sandwich $L_p(h)$ or $L_p(\mathcal{H})$ between *bounds over all possible choices of p*
- What else would we do anyway?
- p is conceptually clean and simple
- The unattainable holy grail
- Think of p as an oracle that sells samples from $X \times Y$
- She knows p, we don't
- Samples cost money and effort!
  [Example: MNIST Database]

Even More Importantly...

- We know what "related" means:
  T, V, S are all drawn independently from $p(\mathbf{x}, y)$
- We know what "generalize" means:
  Find $\mathrm{RM}_p(\mathcal{H}) \in \arg\min_{h \in \mathcal{H}} L_p(h)$
- We know the goal of machine learning

Validation

- Hyper-parametric family of hypothesis spaces $\mathcal{H} = \bigcup_{\pi \in \Pi} \mathcal{H}_\pi$
- Finding a good vector $\hat{\pi}$ of hyper-parameters is called *model selection*
- A popular method is called *validation*
- Use a *validation set V* separate from T
- Pick a hyper-parameter vector for which the predictor trained on the *training* set minimizes the *validation* risk

$$\hat{\pi} = \arg\min_{\pi \in \Pi} L_V(\text{ERM}_T(\mathcal{H}_\pi))$$

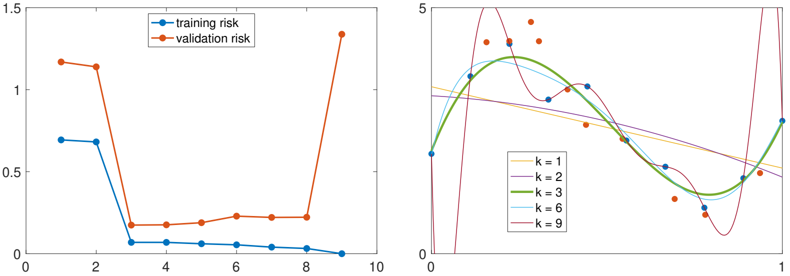
- When the set Π of hyper-parameters is finite, try them all

Validation Algorithm

procedure $\textsc{Validation}(\mathcal{H}, \Pi, T, V, \ell)$
  $\hat{L} = \infty$ ▷ Stores the best risk so far on V
  for $\pi \in \Pi$ **do**
    $h \in \arg\min_{h' \in \mathcal{H}_\pi} L_T(h')$ ▷ Use loss ℓ to compute best predictor $\text{ERM}_T(\mathcal{H}_{\boldsymbol{\pi}})$ on T
    $L = L_V(h)$ ▷ Use loss ℓ to evaluate the predictor's risk on V
    if $L < \hat{L}$ **then**
      $(\hat{\pi}, \hat{h}, \hat{L}) = (\pi, h, L)$ ▷ Keep track of the best hyper-parameters, predictor, and risk
    end if
  end for
  return $(\hat{\pi}, \hat{h}, \hat{L})$ ▷ Return best hyper-parameters, predictor, and risk estimate
end procedure

Validation for Infinite Sets

- When Π is not finite, scan and find a local minimum
- Example: Polynomial degree



- When Π is not countable, scan a grid and find a local minimum

Resampling Methods for Validation

- Validation is good but expensive: needs separate data
- A pity not to use V as part of T!
- Resampling methods split T into T_k and V_k for $k = 1, \ldots, K$
- (Nothing to do with number of classes or polynomial degree!)
- For each π, for each round k, train on T_k, test on V_k to measure performance
- Average performance over k taken as validation risk for π
- Let $\hat{\pi}$ be the best π
- When done, train the predictor in $\mathcal{H}_{\hat{\pi}}$ and on all of T
- *Cross-validation* and the *bootstrap* differ on how splits are made

K-Fold Cross-Validation

- $V_1, \ldots, V_K$ are a partition of T into approximately equal-sized sets
- $T_k = T \setminus V_k$
- For $\pi \in \Pi$

  For $k = 1, \ldots, K$:

  train on T_k, measure performance on V_k

  Average performance over k is validation risk for π
- Pick $\hat{\pi}$ as the π with best average performance
- When done, train the predictor in $\mathcal{H}_{\hat{\pi}}$ and on all of T
- Since performance is an average, we also get a variance!
- We don't have that for standard validation

How big should K be?

- T_k has $|T|(K-1)/K$ samples, so the predictor in each fold is a bit worse than the final predictor
- Smaller K: More pessimistic risk estimate
  (upward bias b/c we train on smaller T_k)
- Bigger K decreases bias of risk estimate
  (training on bigger T_k)
- Why not $K = N$?
- LOOCV (Leave-One-Out Cross-Validation)
- Train on all but one data point, validate on that data point, repeat
- Any issue?
- Nadeau and Bengio recommend $K = 15$

The Bootstrap

- *Bag* or *multiset*: A set that allows for multiple instances
- $\{a, a, b, b, b, c\}$ has cardinality 6
- *Multiplicities*: 2 for a, 3 for b, and 1 for c
- A set is also a bag: $\{a, b, c\}$
- Bootstrap: Same as CV, except
  - T_k: N samples drawn uniformly at random from T, *with replacement*
  - $V_k = T \setminus T_k$
- T_k is a bag, V_k is a set

How Many Elements are in V_k?

- Fix attention on one sample s

  $\mathbb{P}[s \text{ is drawn in one draw}] = 1/N$

  $\mathbb{P}[s \text{ is not drawn in one draw}] = 1 - 1/N$

  $\mathbb{P}[s \text{ is not drawn ever}] = (1 - 1/N)^N$
- Average fraction of missing elements $(1 - 1/N)^N$
- For large N, this is about

  $\lim_{N\to\infty} \left(1 - \frac{1}{N}\right)^N = \frac{1}{e} \approx 0.37$
- Good approximation: $(1 - 1/24)^{24} \approx 0.36$
- 37 % of elements are missing from T_k on average and make it into V_k
- 63 % of elements end up in T_k on average

Cross-Validation vs Bootstrap

- Bootstrap estimates are good
- Typically somewhat more biased than CV (because $|T_k| \approx 0.63|T|$)
- CV is method of choice for model selection
- Bootstrap leads to *random decision forests*