ML in Practice:

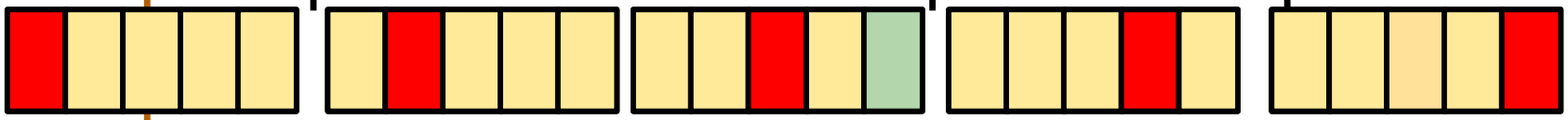
CMSC 422

Slides adapted from Prof. CARPUAT and Prof. Roth

N-fold cross validation

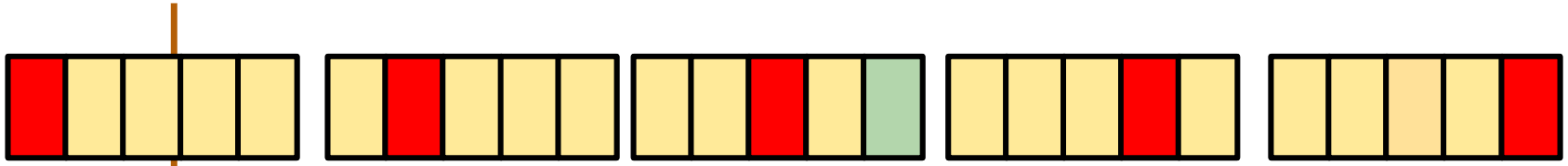
- Instead of a single test-training split:

- Split data into N equal-sized parts



- Train and test N different classifiers
- Report average accuracy and standard deviation of the accuracy

Evaluation: significance tests



- You have two different classifiers, A and B
- You train and test them on the same data set using N-fold cross-validation
- For the n-th fold:

  accuracy(A, n), accuracy(B, n)

  p_n = accuracy(A, n) - accuracy(B, n)
- Is the difference between A and B's accuracies significant?

Hypothesis testing

- You want to show that hypothesis H is true, based on your data
  - (e.g. H = "classifier A and B are different")
- Define a null hypothesis H_0
  - (H_0 is the contrary of what you want to show)
- H_0 defines a distribution $P(m \mid H_0)$ over some statistic
  - e.g. a distribution over the difference in accuracy between A and B
- Can you refute (reject) H_0?

Rejecting H_0

- H_0 defines a distribution $P(M \mid H_0)$ over some statistic M
  - (e.g. M= the difference in accuracy between A and B)
- Select a significance value S
  - (e.g. 0.05, 0.01, etc.)
  - You can only reject H0 if $P(m \mid H_0) \leq S$
- Compute the test statistic m from your data
  - e.g. the average difference in accuracy over your N folds
- Compute $P(m \mid H_0)$
- Refute H_0 with $p \leq S$ if $P(m \mid H_0) \leq S$

Paired t-test

- Null hypothesis (H_0; to be refuted):
  - There is no difference between A and B, i.e. the expected accuracies of A and B are the same
- That is, the expected difference (over all possible data sets) between their accuracies is 0:

  H_0: $E[p_D] = 0$
- We don't know the true $E[p_D]$
- N-fold cross-validation gives us N samples of p_D

Paired t-test

- Null hypothesis H_0: $E[diff_D] = \mu = 0$
- *m*: our estimate of μ based on *N* samples of $diff_D$

$$m = 1/N \sum_n diff_n$$

- The estimated variance S^2:

$$S^2 = 1/(N-1) \sum_{1,N} (diff_n - m)^2$$

- Accept Null hypothesis at significance level *a* if the following statistic lies in $(-t_{a/2, N-1}, +t_{a/2, N-1})$

$$\frac{\sqrt{N}m}{S} \sim t_{N-1}$$

Imbalanced data distributions

- Sometimes training examples are drawn from an imbalanced distribution
- This results in an imbalanced training set
  - "needle in a haystack" problems
  - E.g., find fraudulent transactions in credit card histories
- Why is this a big problem for the ML algorithms we know?

Learning with imbalanced data

- We need to let the learning algorithm know that we care about some examples more than others!

- 2 heuristics to balance the training data
  - Subsampling
  - Weighting

Recall: Machine Learning as Function Approximation

Problem setting

- Set of possible instances X
- Unknown target function $f: X \rightarrow Y$
- Set of function hypotheses $H = \{h \mid h: X \rightarrow Y\}$

Input

- Training examples $\{(x^{(1)}, y^{(1)}), \ldots (x^{(N)}, y^{(N)})\}$ of unknown target function f

Output

- Hypothesis $h \in H$ that best approximates target function f

Recall: Loss Function

$l(y, f(x))$ where y is the truth and $f(x)$ is the system's prediction

$$\text{e.g. } l(y, f(x)) = \begin{cases} 0 & if\ y = f(x) \\ 1 & otherwise \end{cases}$$

Captures our notion of what is important to learn

Recall: Expected loss

- f should make good predictions
  - as measured by loss l
  - on **future** examples that are also drawn from D

- Formally
  - ε , the expected loss of f over D with respect to l should be small

$$\varepsilon \triangleq \mathbb{E}_{(x,y)\sim D}\{l(y, f(x))\} = \sum_{(x,y)} D(x,y)\, l(y, f(x))$$

Task: Binary Classification

Given:

1. An input space $\mathcal{X}$
2. An unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$

Compute: A function f minimizing: $\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[f(\boldsymbol{x}) \neq y\right]$

TASK: α-WEIGHTED BINARY CLASSIFICATION

Given:

1. An input space $\mathcal{X}$
2. An unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$

Compute: A function f minimizing: $\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\alpha^{y=1}\left[f(\boldsymbol{x}) \neq y\right]\right]$

Given a good algorithm for solving the binary classification problem, how can I solve the α-weighted binary classification problem?

We define cost of misprediction as:

α > 1 for y = +1

1 for y = -1

Solution: Train a binary classifier on an induced distribution

Algorithm 11 SUBSAMPLEMAP($\mathcal{D}^{weighted}$, α)

while *true* **do**
  $(x, y) \sim \mathcal{D}^{weighted}$ // draw an example from the weighted distribution
  $u \sim$ uniform random variable in $[0, 1]$
  if $y = +1$ **or** $u < \frac{1}{\alpha}$ **then**
    return (x, y)
  end if
end while

Subsampling optimality

- **Theorem:** If the binary classifier achieves a binary error rate of ε, then the error rate of the α-weighted classifier is α ε

- Proof (CIML 6.1)

Strategies for inducing a new binary distribution

- Undersample the negative class
- Oversample the positive class

Strategies for inducing a new binary distribution

- Undersample the negative class
  - More computationally efficient
- Oversample the positive class
  - Base binary classifier might do better with more training examples
  - Efficient implementations incorporate weight in algorithm, instead of explicitly duplicating data!

Algorithm 1 DecisionTreeTrain(*data*, *remaining features*)

```
guess ← most frequent answer in data                    // default answer for this data
if the labels in data are unambiguous then
    return Leaf(guess)                                  // base case: no need to split further
else if remaining features is empty then
    return Leaf(guess)                                  // base case: cannot split further
else                                                    // we need to query more features
    for all f ∈ remaining features do
        NO ← the subset of data on which f=no
        YES ← the subset of data on which f=yes
        score[f] ← # of majority vote answers in NO
                   + # of majority vote answers in YES
                                                        // the accuracy we would get if we only queried on f
    end for
    f ← the feature with maximal score(f)
    NO ← the subset of data on which f=no
    YES ← the subset of data on which f=yes
    left ← DecisionTreeTrain(NO, remaining features \ {f})
    right ← DecisionTreeTrain(YES, remaining features \ {f})
    return Node(f, left, right)
end if
```

Topics

Given an arbitrary method for binary classification, how can we learn to make multiclass predictions?

Fundamental ML concept: reductions

Multiclass classification

- Real world problems often have multiple classes (text, speech, image, biological sequences…)

- How can we perform multiclass classification?
  - Straightforward with decision trees or KNN
  - Can we use the perceptron algorithm?

Reductions

- Idea is to re-use simple and efficient algorithms for binary classification to perform more complex tasks

- Works great in practice:
  - E.g., Vowpal Wabbit

One Example of Reduction: Learning with Imbalanced Data

TASK: α-WEIGHTED BINARY CLASSIFICATION

Given:

1. An input space $\mathcal{X}$
2. An unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$

Compute: A function f minimizing: $\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\alpha^{y=1}\left[f(x) \neq y\right]\right]$

Subsampling Optimality Theorem:
If the binary classifier achieves a binary error rate of ε, then the error rate of the α-weighted classifier is α ε

What you should know

- Be aware of practical issues when applying ML techniques to new problems
- How to select an appropriate evaluation metric for imbalanced learning problems
- How to learn from imbalanced data using α-weighted binary classification, and what the error guarantees are