Practical Issues: Features, Evaluation, Debugging

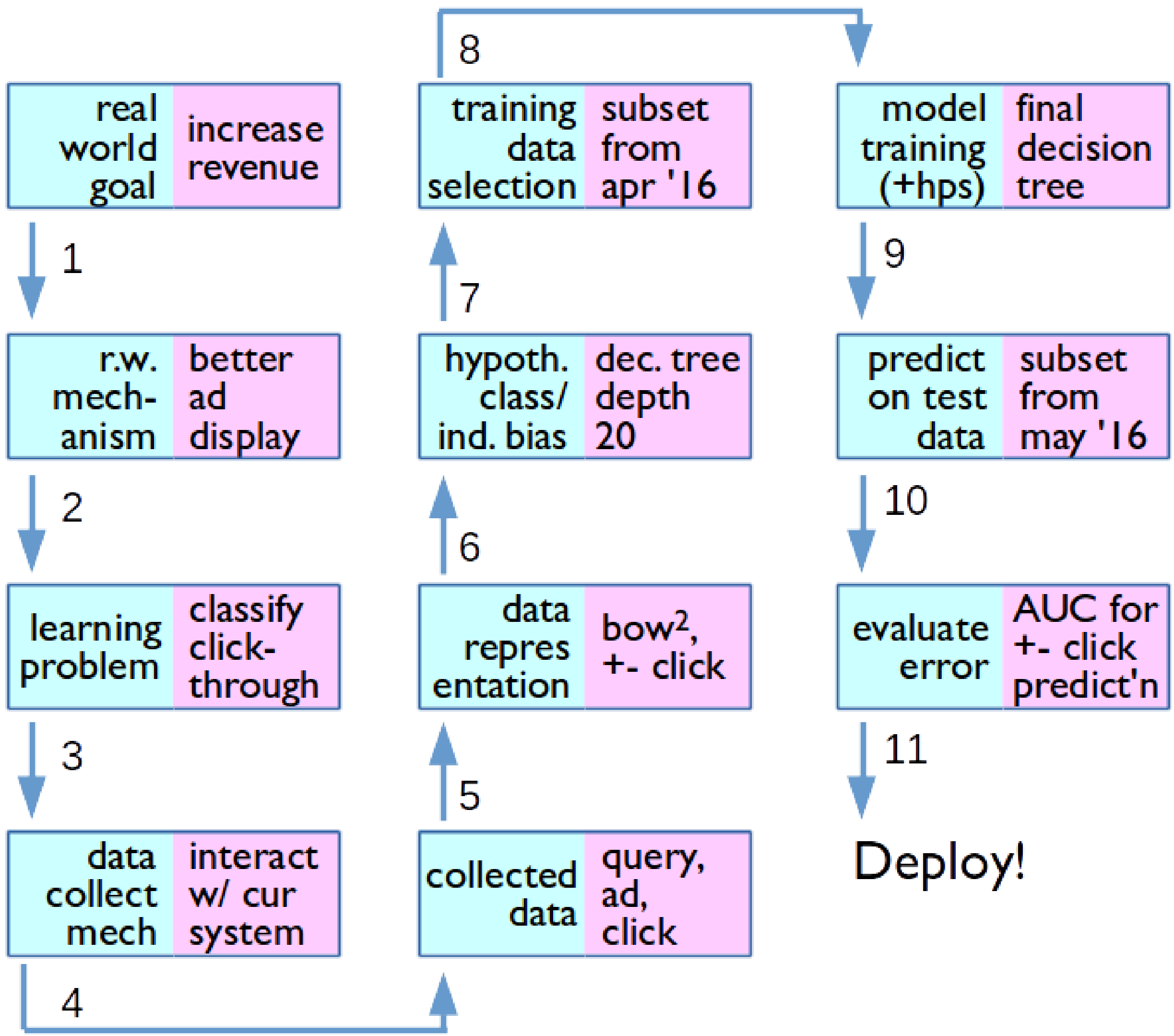
CMSC 422

MARINE CARPUAT
marine@cs.umd.edu

Today: Practical issues

- Learning algorithm is only one of many steps in designing a ML application
- Many things can go wrong, but there are practical strategies for
  - Improving inputs
  - Evaluating
  - Tuning
  - Debugging
- Fundamental ML concepts: estimation vs. approximation error

Typical Design Process for an ML Application



Practical Issues

- “garbage in, garbage out”
  - Learning algorithms can’t compensate for useless training examples
    - E.g., if all features are irrelevant
- Feature design can have bigger impact on performance than tweaking the learning algorithm
    - E.g., feature combination

Improving Input Representations

- Feature pruning
- Feature normalization

$$\text{Centering:} \quad x_{n,d} \leftarrow x_{n,d} - \mu_d \tag{5.1}$$

$$\text{Variance Scaling:} \quad x_{n,d} \leftarrow x_{n,d} / \sigma_d \tag{5.2}$$

$$\text{Absolute Scaling:} \quad x_{n,d} \leftarrow x_{n,d} / r_d \tag{5.3}$$

$$\textit{where:} \quad \mu_d = \frac{1}{N} \sum_n x_{n,d} \tag{5.4}$$

$$\sigma_d = \sqrt{\frac{1}{N-1} \sum_n (x_{n,d} - \mu_d)^2} \tag{5.5}$$

$$r_d = \max_n \left| x_{n,d} \right| \tag{5.6}$$

- Example normalization $\boldsymbol{x}_n \leftarrow \boldsymbol{x}_n / \, ||\boldsymbol{x}_n||$

See CIML 5.3

Practical Issues: Evaluation

- So far we've measured classification performance using **accuracy**

- But this is not a good metric when some errors matter mode than others
  - Given medical record, predict whether patient has cancer or not
  - Given a document collection and a query, find documents in collection that are relevant to query

The 2-by-2 contingency table

Imagine we are addressing a document retrieval task for a given query, where
+1 means that the document is relevant
-1 means that the document is not relevant

We can categorize predictions as:
- true/false positives
- true/false negatives

	Gold label = +1	Gold label = -1
Prediction = +1	tp	fp
Prediction = -1	fn	tn

Precision and recall

- **Precision**: % of positive predictions that are correct
- **Recall**: % of positive gold labels that are found

	Gold label = +1	Gold label = -1
Prediction = +1	tp	fp
Prediction = -1	fn	tn

Practical Issues: hyperparameter tuning with dev set vs. cross-validation

Algorithm 8 CROSSVALIDATE(*LearningAlgorithm*, *Data*, K)

$\hat{\epsilon} \leftarrow \infty$ // store lowest error encountered so far
$\hat{\alpha} \leftarrow$ unknown // store the hyperparameter setting that yielded it
for all hyperparameter settings α **do**
  $err \leftarrow [\ ]$ // keep track of the K-many error estimates
  for $k = 1$ **to** K **do**
    $train \leftarrow \{(x_n, y_n) \in Data : n \mod K \neq k - 1\}$
    $test \leftarrow \{(x_n, y_n) \in Data : n \mod K = k - 1\}$ // test every Kth example
    $model \leftarrow$ Run *LearningAlgorithm* on *train*
    $err \leftarrow err \oplus$ error of *model* on *test* // add current error to list of errors
  end for
  $avgErr \leftarrow$ mean of set err
  if $avgErr < \hat{\epsilon}$ **then**
    $\hat{\epsilon} \leftarrow avgErr$ // remember these settings
    $\hat{\alpha} \leftarrow \alpha$ // because they're the best so far
  end if
end for

Practical Issues: Debugging!

- You've implemented a learning algorithm,
- You try it on some train/dev/test data
- You get really bad performance

- What's going on?
  - Is the data too noisy?
  - Is the learning problem too hard?
  - Is the implementation of the learning algorithm buggy?

Strategies for Isolating Causes of Errors

- Is the problem with **generalization** to test data?
  - Can learner fit the training data?
  - Yes: problem is in generalization to test data
  - No: problem is in representation (need better features or better data)

- **Train/test mismatch**?
  - Try reselecting train/test by shuffling training data and test together

Strategies for Isolating Causes of Errors

- Is algorithm **implementation correct**?
  - Measure loss rather than accuracy
  - Hand-craft a toy dataset
- **Is representation adequate?**
  - Can you learn if you add a cheating feature that perfectly correlates with correct class?
- Do you have **enough data**?
  - Try training on 80% of the training set, how much does it hurt performance?

Formalizing Errors

The learned classifier

$\mathcal{F}$ set of all possible classifiers using a fixed representation

$$\mathrm{error}(f) = \underbrace{\left[\mathrm{error}(f) - \min_{f^* \in \mathcal{F}} \mathrm{error}(f^*)\right]}_{\textbf{estimation error}} + \underbrace{\left[\min_{f^* \in \mathcal{F}} \mathrm{error}(f^*)\right]}_{\textbf{approximation error}}$$

How far is the learned classifier f from the optimal classifier f*?

Quality of the model family

The bias/variance trade-off

- Trade-off between
  - approximation error (error due to bias)
  - estimation error (error due to variance)

- Example:
  - Consider the always positive classifier
    - Strongly biased toward predicting +1 no matter what the input
    - Low variance as a function of a random draw of the training set

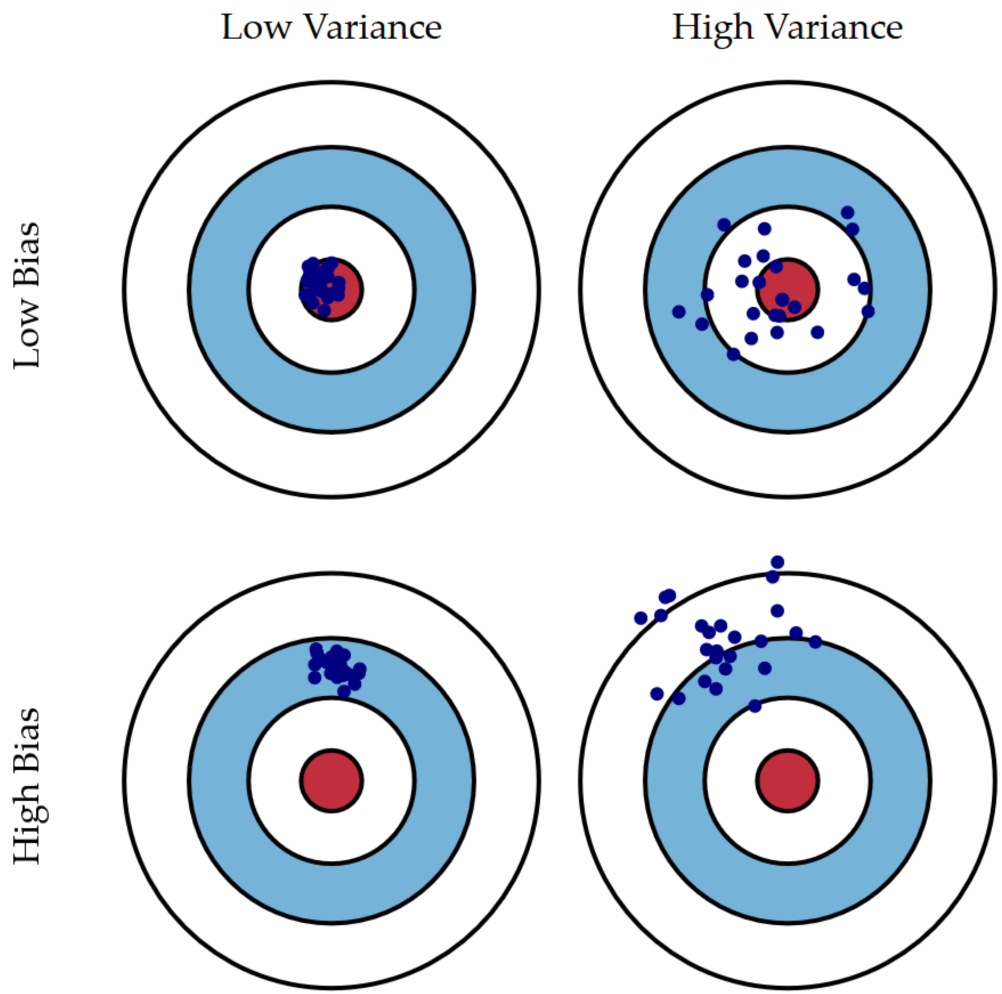
The bias/variance trade-off illustrated

Assume

Center of the target = model that perfectly predicts the correct values.

As we move away from the bulls-eye, predictions get worse and worse.

Each hit represents a classifier trained on a sample from data generating distribution.



http://scott.fortmann-roe.com/docs/BiasVariance.html

Recap: practical issues

- Learning algorithm is only one of many steps in designing a ML application
- Many things can go wrong, but there are practical strategies for
  - Improving inputs
  - Evaluating
  - Tuning
  - Debugging
- Fundamental ML concepts: estimation vs. approximation error, bias/variance trade-off