CS 461: Machine Learning
Lecture 9

Dr. Kiri Wagstaff
kiri.wagstaff@calstatela.edu

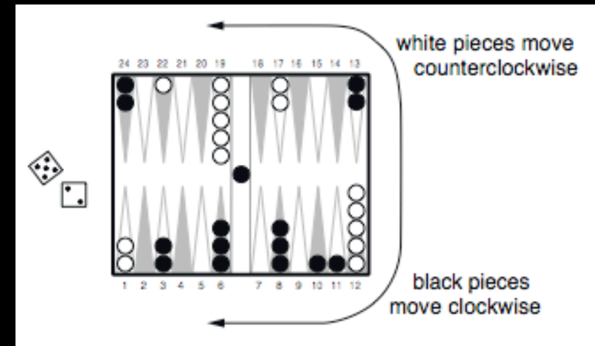
Plan for Today

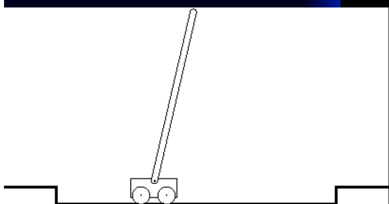
- Review Reinforcement Learning

- Ensemble Learning
  - How to combine forces?
  - Voting
  - Error-Correcting Output Codes
  - Bagging
  - Boosting

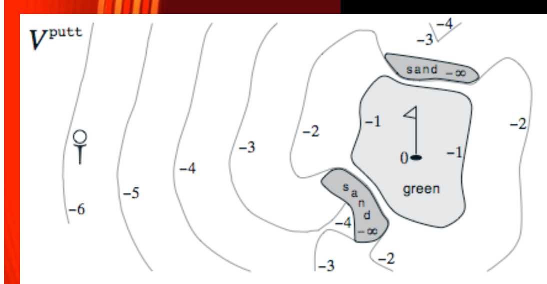
- Homework 5
- Evaluations

Review from Lecture 8





- Reinforcement Learning
  - How different from supervised, unsupervised?
- Key components
  - Actions, states, transition probs, rewards
  - Markov Decision Process
  - Episodic vs. continuing tasks
  - Value functions, optimal value functions
- Learn: policy (based on V, Q)
  - Model-based: value iteration, policy iteration
  - TD learning
    - Deterministic: backup rules (max)
    - Nondeterministic: TD learning, Q-learning (running avg)



Ensemble Learning

Chapter 15

What is Ensemble Learning?

- "No Free Lunch" Theorem
  - No single algorithm wins all the time!
- Ensemble: collection of base learners
  - Combine the strengths of each to make a super-learner
  - Also considered "meta-learning"
- How can you get different learners?

- How can you combine learners?

Where do Learners come from?

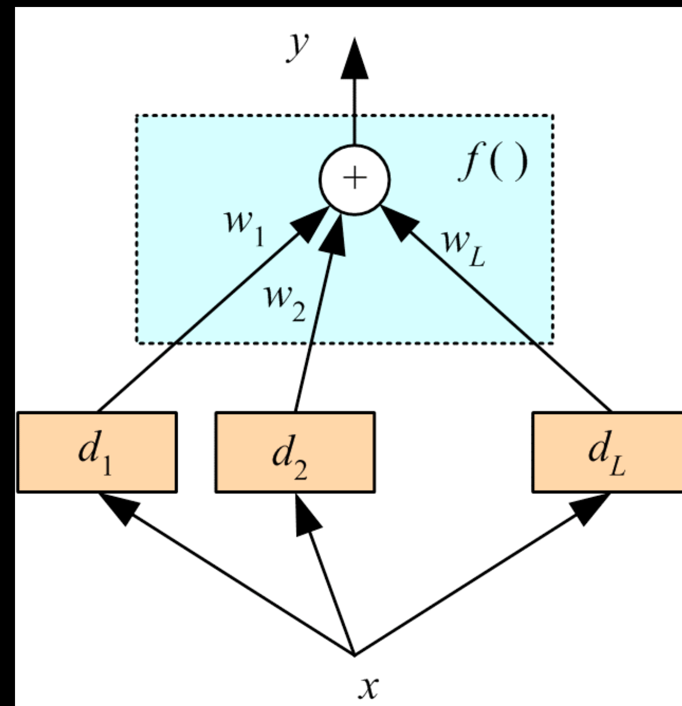
- Different learning algorithms
- Algorithms with different choice for parameters
- Data set with different features
- Data set = different subsets
- Different sub-tasks

Combine Learners: Voting

- Linear combination (weighted vote)

$$y = \sum_{j=1}^{L} w_j d_j$$

$$w_j \geq 0 \text{ and } \sum_{j=1}^{L} w_j = 1$$



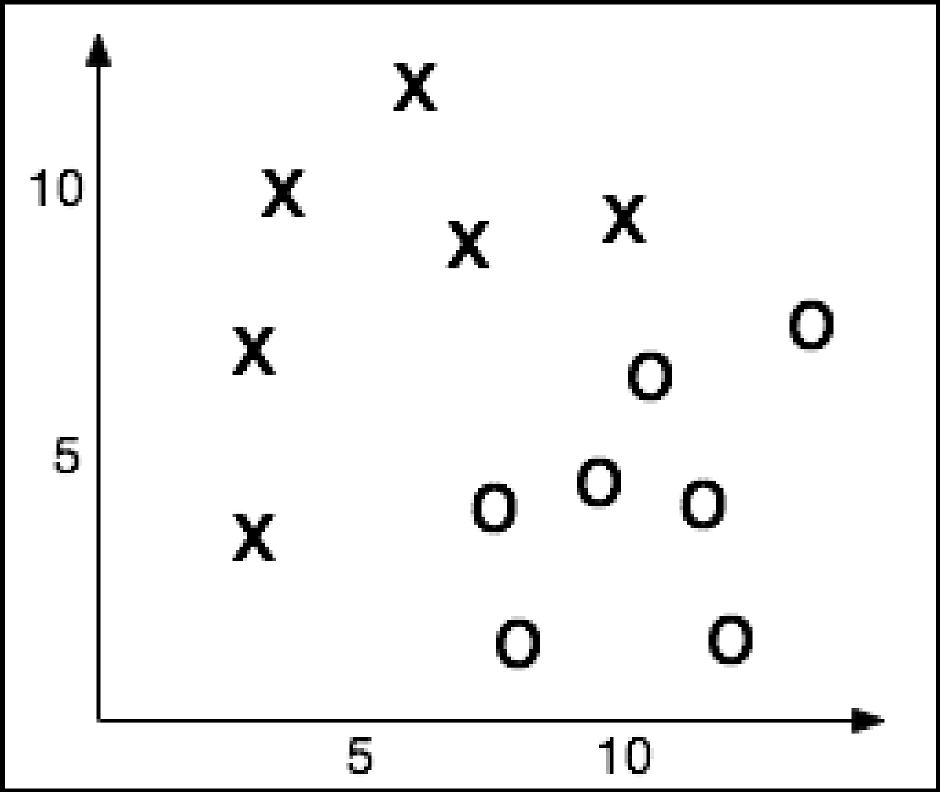
- Classification

$$y_i = \sum_{j=1}^{L} w_j d_{ji}$$

Bayesian

$$P(C_i \mid x) = \sum_{\text{all models } \mathcal{M}_j} P(C_i \mid x, \mathcal{M}_j) P(\mathcal{M}_j)$$

Exercise: x's and o's



Different Learners: ECOC

- Error-Correcting Output Code
  - = how to define sub-tasks to get different learners
  - Maybe use the same base learner, maybe not
  - Key: want to be able to detect errors!
- Example: dance steps to convey secret command
  - Three valid commands

Attack	Retreat	Wait
R L R	L L R	R R R

Not an ECOC

Attack	Retreat	Wait
R L R	L L L	R R L

ECOC

Error-Correcting Output Code

- Specifies how to interpret (and detect errors in) learner outputs
- K classes, L learners
- One learner per class, L=K

Column = defines task for learner *l*

Row = encoding of class *k*

$$W = \begin{bmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{bmatrix}$$

ECOC: Pairwise Classification

- L = K(K-1)/2
- 0 = "don't care"

$$
W = \begin{bmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}
$$

ECOC: Full Code

- Total # columns = 2^(K-1) - 1
  - For K=4:

$$\mathbf{W} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & +1 & +1 & +1 & +1 \\ -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 \end{bmatrix}$$

- Goal: choose L sub-tasks (columns)
  - Maximize row dist: detect errors
  - Maximize column dist: different sub-tasks
- Combine outputs by weighted voting

$$y_i = \sum_{j=1}^{L} w_j d_{ji}$$

Different Learners: Bagging

- Bagging = "bootstrap aggregation"
  - Bootstrap: draw N items from X with replacement
- Want "unstable" learners
  - Unstable: high variance
  - Decision trees and ANNs are unstable
  - K-NN is stable
- Bagging
  - Train L learners on L bootstrap samples
  - Combine outputs by voting

Different Learners: Boosting

- Boosting: train next learner on mistakes made by previous learner(s)
- Want "weak" learners
  - Weak: P(correct) > 50%, but not necessarily by a lot
  - Idea: solve easy problems with simple model
  - Save complex model for hard problems

Original Boosting

1. Split data X into {X1, X2, X3}
2. Train L1 on X1
   - Test L1 on X2
3. Train L2 on L1's mistakes on X2 (plus some right)
   - Test L1 and L2 on X3
4. Train L3 on disagreements between L1 and L2

- Testing: apply L1 and L2; if disagree, use L3

- Drawback: need large X

AdaBoost = Adaptive Boosting

- Arbitrary number of base learners
- Re-use data set (like bagging)
- Use errors to adjust probability of drawing samples for next learner
  - Reduce probability if it's correct
- Testing: vote, weighted by training accuracy

- Key difference from bagging:
  - Data sets not chosen by chance; instead use performance of previous learners to select data

AdaBoost

Training:

For all $\{x^t, r^t\}_{t=1}^N \in \mathcal{X}$, initialize $p_1^t = 1/N$

For all base-learners $j = 1, \ldots, L$

Randomly draw $\mathcal{X}_j$ from $\mathcal{X}$ with probabilities p_j^t

Train d_j using $\mathcal{X}_j$

For each (x^t, r^t), calculate $y_j^t \leftarrow d_j(x^t)$

Calculate error rate: $\epsilon_j \leftarrow \sum_t p_j^t \cdot 1(y_j^t \neq r^t)$

If $\epsilon_j > 1/2$, then $L \leftarrow j - 1$; stop

$\beta_j \leftarrow \epsilon_j / (1 - \epsilon_j)$

For each (x^t, r^t), decrease probabilities if correct:

If $y_j^t = r^t$ $p_{j+1}^t \leftarrow \beta_j p_j^t$ Else $p_{j+1}^t \leftarrow p_j^t$

Normalize probabilities:

$Z_j \leftarrow \sum_t p_{j+1}^t; \quad p_{j+1}^t \leftarrow p_{j+1}^t / Z_j$

Testing:

Given x, calculate $d_j(x), j = 1, \ldots, L$

Calculate class outputs, $i = 1, \ldots, K$:

$$y_i = \sum_{j=1}^L \left(\log \frac{1}{\beta_j}\right) d_{ji}(x)$$

AdaBoost Applet

- http://www.cs.ucsd.edu/~yfreund/adaboost/index.html

Summary: Key Points for Today

- No Free Lunch theorem
- Ensemble: combine learners
- Voting
- Error-Correcting Output Codes
- Bagging
- Boosting

Homework 5

Next Time

- Final Project Presentations (no reading assignment!)
  - Use order on website
- Submit slides on CSNS by midnight March 7
  - No, really
  - You may not be able to present if you don't
- Reports are due to CSNS midnight March 8
  - Early submission: March 1