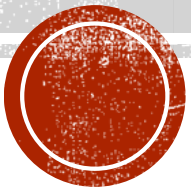
BEYOND BINARY CLASSIFICATION: RANKING



CS 412 Introduction to Machine Learning

Prof. Zheleva

February 13, 2018

Reading Assignment: CIML: 6

LAST LECTURE: PRACTICAL ISSUES

- Bias-variance tradeoff
- How can we take the standard binary classifier and adapt it to handle problems with
  - Imbalanced data distributions
  - Multiclass classification problems (OVA & AVA)
- How to assess statistical significance for classifiers
- Algorithms & guarantees on error rate
- Fundamental ML concepts
  - Reductions

ML APPLICATION DESIGN

1	real world goal	increase revenue
2	real world mechanism	better ad display
3	learning problem	classify click-through
4	data collection	interaction w/ current system
5	collected data	query, ad, click
6	data representation	bow^2, $\pm$ click
7	select model family	decision trees, depth 20
8	select training data	subset from april'16
9	train model & hyperparams	final decision tree
10	predict on test data	subset from may'16
11	evaluate error	zero/one loss for $\pm$ click
12	deploy!	(hope we achieve our goal)

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?

BIAS-VARIANCE TRADE-OFF FOR KNN

$$Err(x) = \left(E[\hat{f}_D(x)] - f(x)\right)^2 + E\left[\left(\hat{f}_D(x) - E[\hat{f}_D(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

Derivation for k-NN: $$Err(x) = \left(f(x) - \frac{1}{k}\sum_{i=1}^{k} f(x_i)\right)^2 + \frac{\sigma_\epsilon^2}{k} + \sigma_\epsilon^2$$

What happens with the bias and variance as k increases?

LEARNING WITH IMBALANCED DATA IS AN EXAMPLE OF REDUCTION

TASK: α-WEIGHTED BINARY CLASSIFICATION

Given:

1. An input space $\mathcal{X}$
2. An unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$
3. A training set D sampled from $\mathcal{D}$

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

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

MULTICLASS CLASSIFICATION

- **Theorem:** Suppose that the average error of the K binary classifiers is ε, then the error rate of the OVA multiclass classifier is at most $(K-1)\,\varepsilon$
- **Theorem:** Suppose that the average error of the K binary classifiers is ε, then the error rate of the AVA multiclass classifier is at most $2(K-1)\,\varepsilon$

EXTENSIONS

- Divide and conquer
  - Organize classes into binary tree structures

Theorem 6 (Tree Error Bound). *Suppose the average binary classifiers error is ϵ. Then the error rate of the tree classifier is* at most $\lceil \log_2 K \rceil \epsilon$.

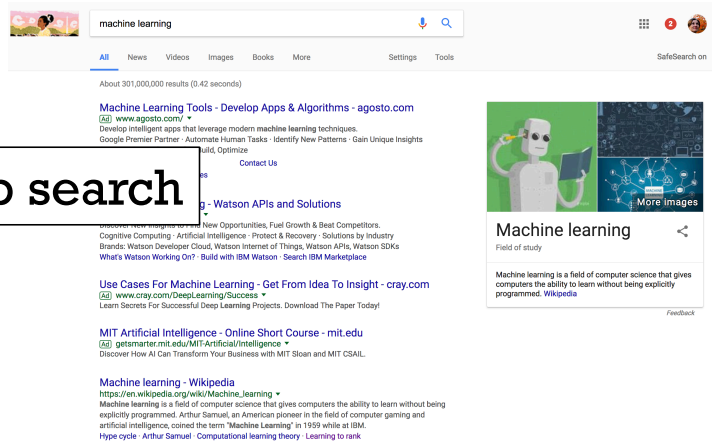
- Use confidence to weight predictions of binary classifiers
  - Instead of using majority vote

TODAY: PRACTICAL ISSUES + LINEAR MODELS

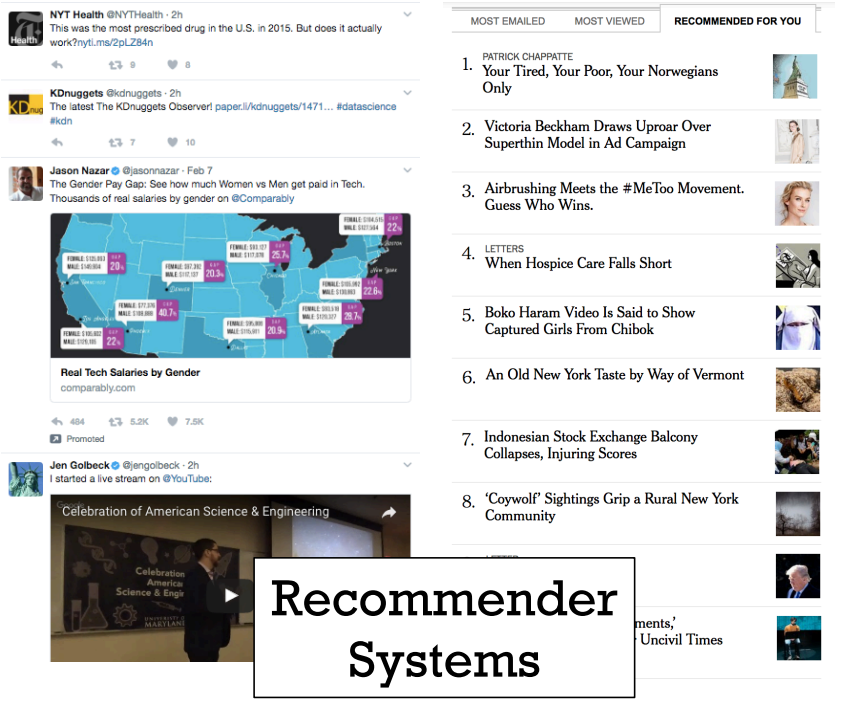
- Ranking
  - Reduction to binary classification
  - Loss functions for ranking
- Linear models

RANKING

- Canonical example: web search
  - Given all the documents on the web
- For a user query, retrieve relevant documents, ranked from most relevant to least relevant



Web search



Recommender Systems

REDUCTIONS

- How can we take the standard binary classifier and adapt it to handle problems with
  - Imbalanced data distributions
  - Multiclass classification problems
  - Ranking

PREFERENCE FUNCTION

- Given a query q and two documents d_i and d_j, the preference function outputs whether
  - d_i should be preferred to d_j
  - Or d_j should be preferred to d_i
- That's a binary classification problem!
  - Combine the features of q, d_i and d_j into one set
  - Class is assigned based on which document is higher in ranking

SPECIFYING THE REDUCTION FROM RANKING TO BINARY CLASSIFICATION

- How to train classifier that predicts preferences?
  - Pairwise ranking
- How to turn the predicted preferences into a ranking?
  - Find the overall permutation

Algorithm 17 NAIVERANKTRAIN(*RankingData*, BINARYTRAIN)

```
1:  D ← [ ]
2:  for n = 1 to N do
3:      for all i, j = 1 to M and i ≠ j do
4:          if i is prefered to j on query n then
5:              D ← D ⊕ (x_nij, +1)
6:          else if j is prefered to i on query n then
7:              D ← D ⊕ (x_nij, −1)
8:          end if
9:      end for
10: end for
11: return BINARYTRAIN(D)
```

N: number of queries
M: number of documents
X_{nij}: features for query n and documents d_i and d_j

Algorithm 18 NAIVERANKTEST(f, $\hat{x}$)

```
1:  score ← ⟨0, 0, ..., 0⟩          // initialize M-many scores to zero
2:  for all i, j = 1 to M and i ≠ j do
3:      y ← f(x̂_ij)                  // get predicted ranking of i and j
4:      score_i ← score_i + y
5:      score_j ← score_j - y
6:  end for
7:  return ARGSORT(score)            // return queries sorted by score
```

documents

NAÏVE APPROACH

- Works well for bipartite ranking problems
  - "is this document relevant or not?"
  - No notion of "is this document more relevant than the other?"
- Not ideal for full ranking problems, because
  - Binary preference problems are not all equally important
  - Separates preference function and sorting

SOLUTION

- Give more weight to higher ranked items
- Define ranking as a function σ
  - Maps the documents to their desired positions
  - If $\sigma_u < \sigma_v$, then document u is preferred over v
  - Given data with observed rankings, we want to learn $\hat{\sigma}$
- Define cost function $\omega(i, j)$
  - Assigns cost to misplacing document from position i to position j
  - Needs to be symmetric: $\omega(i, j) = \omega(j, i)$
  - Monotonic: if i<j<k, then $\omega(i, j) \leq \omega(i, k)$
  - Satisfy triangle inequality: $\omega(i, j) + \omega(j, k) \geq \omega(i, k)$

IMPROVING ON A NAÏVE APPROACH

TASK: ω-RANKING

Given:

1. An input space $\mathcal{X}$
2. An unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \Sigma_M$
3. A training set D sampled from $\mathcal{D}$

Compute: A function $f : \mathcal{X} \to \Sigma_M$ minimizing:

$$\mathbb{E}_{(\boldsymbol{x},\sigma)\sim\mathcal{D}}\left[\sum_{u\neq v}[\sigma_u < \sigma_v]\ [\hat{\sigma}_v < \hat{\sigma}_u]\ \omega(\sigma_u, \sigma_v)\right] \tag{6.7}$$

where $\hat{\sigma} = f(\boldsymbol{x})$

LOSS FUNCTIONS

- Recall: simple loss function for binary classification

$$\text{loss}(y, f(x)) = \begin{cases} 0 \; if \; y = f(x) \\ 1 \quad otherwise \end{cases}$$

- Loss function for ranking

$$\omega(i, j) = \begin{cases} 1 & if \; \min\{i, j\} \leq K \; and \; i \neq j \\ 0 & otherwise \end{cases}$$

  - Only care about getting top K results right

RECAP: PRECISION/RECALL

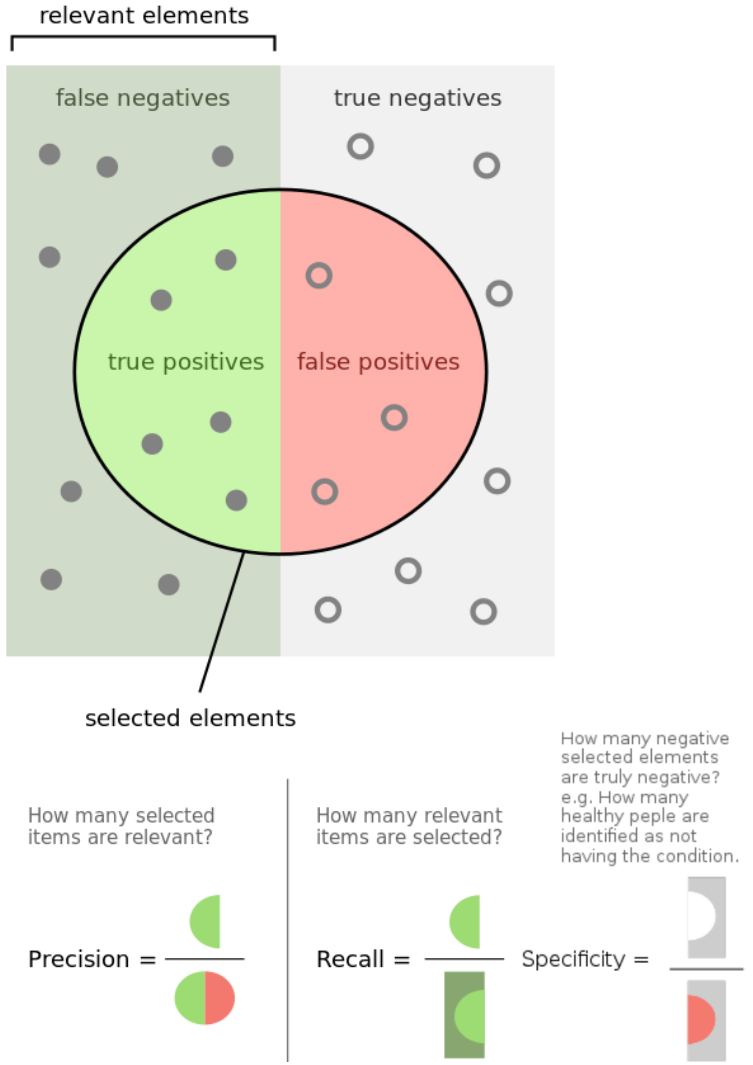
- Precision $Precision = \frac{TP}{TP + FP}$
- Recall = sensitivity = true positive rate

$$Recall = \frac{TP}{TP + FN}$$

- Specificity = true negative rate

$$Specificity = \frac{TN}{TN + FP}$$

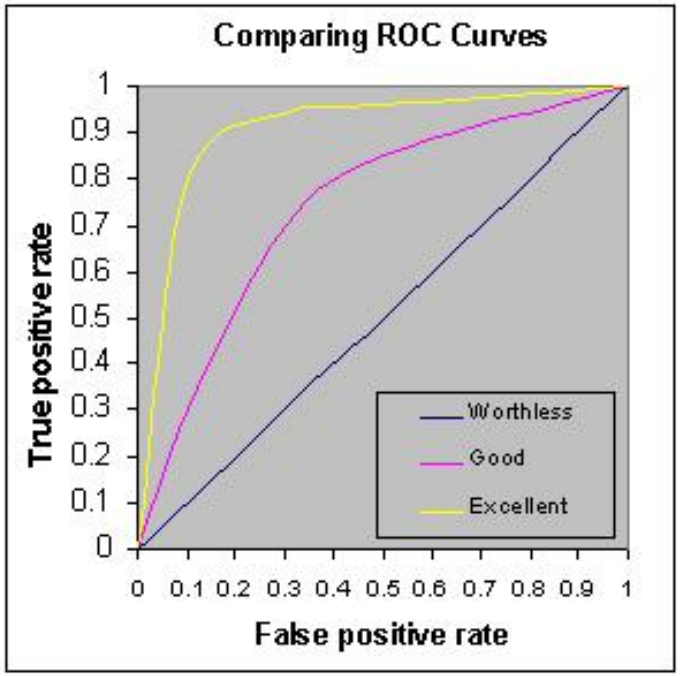
- False positive rate = 1-specificity



Source: https://commons.wikimedia.org/wiki/File:Precisionrecall.svg

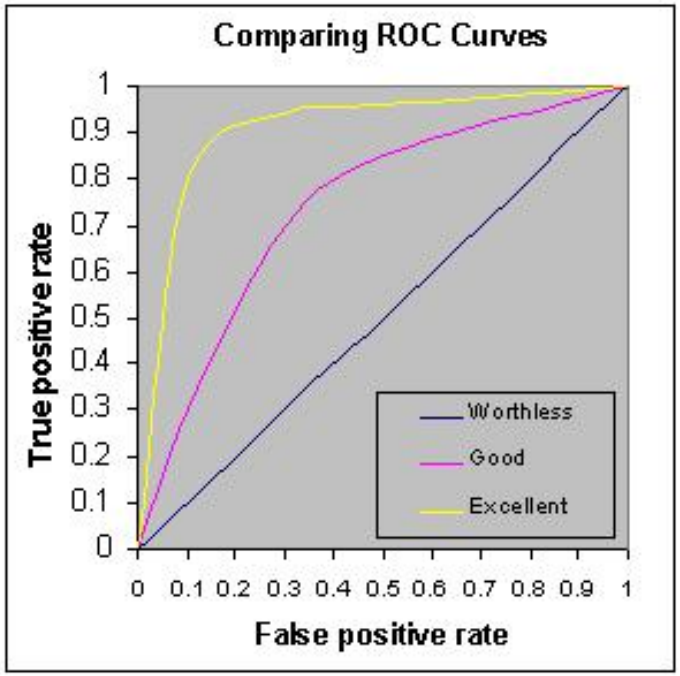
ROC AND AUC

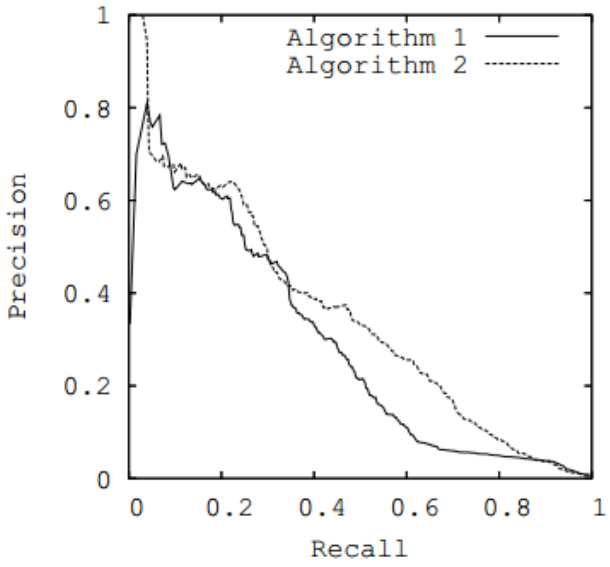
- ROC(receiver operating characteristic) curve shows tradeoff between
  - True positive rate (=sensitivity, recall)
  - False positive rate (=1-specificity)
- AUC: Area under the ROC curve
- Why AUC?
  - Important statistical property: AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance



Source: http://gim.unmc.edu/dxtests/roc3.htm

CONTRAST AND COMPARE AUC VS. F-SCORE





$$Pr = \frac{TP}{TP + FP}$$

$$Re = \frac{TP}{TP + FN}$$

$$F = \frac{2 * Pr * Re}{Pr + Re}$$

$$Re = \frac{FP}{FP + TN}$$

What does it mean to have high AUC but low F score?

LOSS FUNCTIONS

- Loss function for ranking

$$\omega(\mathrm{i},\mathrm{j})=\begin{cases}1 & if\ \min\{i,j\}\le K\ and\ i\neq j\\ 0 & otherwise\end{cases}$$

- Can we use AUC loss function for bipartite ranking case?

$$\omega(i,j)=\frac{\binom{M}{2}}{M^{+}(M-M^{+})}\times\begin{cases}1 & \text{if } i\le M^{+} \text{ and } j>M^{+}\\ 1 & \text{if } j\le M^{+} \text{ and } i>M^{+}\\ 0 & \text{otherwise}\end{cases}$$

  - M objects to be ranked
  - M+ objects that are actually relevant

RESULTING RANKING ALGORITHM

Algorithm 19 RANKTRAIN($\mathbf{D}^{rank}$, ω, BINARYTRAIN)

1: $\mathbf{D}^{bin} \leftarrow [\,]$
2: **for all** $(x, \sigma) \in \mathbf{D}^{rank}$ **do**
3: **for all** $u \neq v$ **do**
4: $y \leftarrow \text{SIGN}(\sigma_v - \sigma_u)$ // y is +1 if u is prefered to v
5: $w \leftarrow \omega(\sigma_u, \sigma_v)$ // w is the cost of misclassification
6: $\mathbf{D}^{bin} \leftarrow \mathbf{D}^{bin} \oplus (y, w, x_{uv})$
7: **end for**
8: **end for**
9: **return** BINARYTRAIN($\mathbf{D}^{bin}$)

Discussed last time how to train a weighted binary classifier

Algorithm 20 RANKTEST(f, $\hat{\boldsymbol{x}}$, obj)

```
1:  if obj contains 0 or 1 elements then
2:      return obj
3:  else
4:      p ← randomly chosen object in obj                          // pick pivot
5:      left ← [ ]                                  // elements that seem smaller than p
6:      right ← [ ]                                 // elements that seem larger than p
7:      for all u ∈ obj \{p} do
8:          ŷ ← f(x_up)                  // what is the probability that u precedes p
9:          if uniform random variable < ŷ then
10:             left ← left ⊕ u
11:         else
12:             right ← right ⊕ u
13:         end if
14:     end for
15:     left ← RANKTEST(f, x̂, left)                          // sort earlier elements
16:     right ← RANKTEST(f, x̂, right)                        // sort later elements
17:     return left ⊕ ⟨p⟩ ⊕ right
18: end if
```

COMPARE TO NAÏVE VERSION

Algorithm 18 NAIVERANKTEST(f, $\hat{x}$)

```
1: score ← ⟨0, 0, ..., 0⟩          // initialize M-many scores to zero
2: for all i, j = 1 to M and i ≠ j do
3:     y ← f(x̂_ij)                  // get predicted ranking of i and j
4:     score_i ← score_i + y
5:     score_j ← score_j - y
6: end for
7: return ARGSORT(score)            // return documents sorted by score
```

RANKTEST

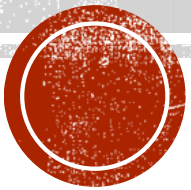
- A probabilistic version of the quicksort algorithm
- Only $O(M\log_2 M)$ calls to f in expectation
- Better error bound than naïve algorithm

> **Theorem 8** (Rank Error Bound). *Suppose the average binary error of f is ϵ. Then the ranking algorithm achieves a test error of at most 2ϵ in the general case, and ϵ in the bipartite case.*

SUMMARY

- What are reductions and why they are useful
- Implement, analyze and prove error bounds of algorithms for
  - Weighted binary classification
  - Multiclass classification (OVA, AVA)
  - Ranking

LINEAR MODELS



CS 412 Introduction to Machine Learning

Prof. Zheleva

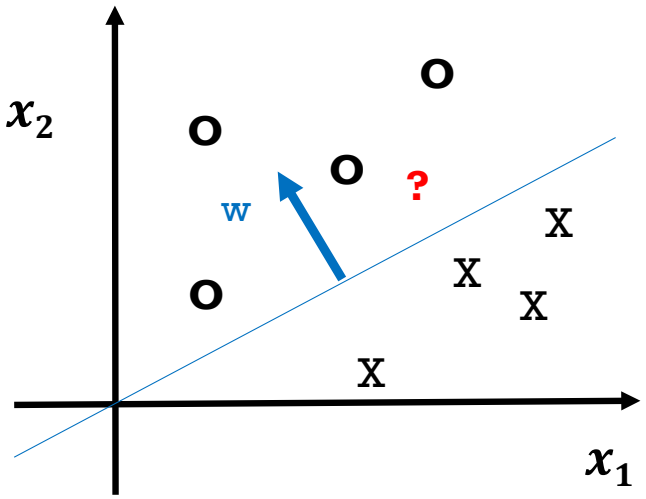
Reading Assignment: CIML: 7

BINARY CLASSIFICATION VIA HYPERPLANES

- A classifier is a hyperplane that separates positive from negative examples (**w**,b)
- At test time, we check on what side of the hyperplane examples fall

$$\hat{y} = sign(\boldsymbol{w}^T \boldsymbol{x} + \boldsymbol{b})$$

- This is a linear classifier model
  - Because the prediction is a linear combination of feature values x
  - Perceptron is one example



TASK: BINARY CLASSIFICATION

Given:

1. An input space $\mathcal{X}$
2. An unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$
3. A training set D sampled from $\mathcal{D}$

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

EXAMPLE ALGORITHM FOR LINEAR CLASSIFICATION: PERCEPTRON

Algorithm 5 PERCEPTRONTRAIN(**D**, *MaxIter*)

1: $w_d \leftarrow 0$, for all $d = 1 \ldots D$ // initialize weights
2: $b \leftarrow 0$ // initialize bias
3: **for** $iter = 1 \ldots MaxIter$ **do**
4: **for all** $(x,y) \in \mathbf{D}$ **do**
5: $a \leftarrow \sum_{d=1}^{D} w_d \, x_d + b$ // compute activation for this example
6: **if** $ya \leq 0$ **then**
7: $w_d \leftarrow w_d + yx_d$, for all $d = 1 \ldots D$ // update weights
8: $b \leftarrow b + y$ // update bias
9: **end if**
10: **end for**
11: **end for**
12: **return** $w_0, w_1, \ldots, w_D, b$

LEARNING A LINEAR CLASSIFIER AS AN OPTIMIZATION PROBLEM

- Not all data is linearly separable
- Goal: minimize the number of misclassified examples

Indicator function: 1 if (.) is true, 0 otherwise

$$\min_{w,b} \sum_{n} \mathbf{1}[y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 0]$$

Objective function
In this case: simple 0/1 loss

Variables over which we are optimizing the objective function

LEARNING A LINEAR CLASSIFIER AS AN OPTIMIZATION PROBLEM

- If linearly separable, efficient algorithms exist
- If not linearly separable, problem is NP-hard
- What about approximate optimization (small constant worse than optimal)?
  - Problem is still NP-hard
- This is just optimizing training error!
  - We care about generalization

LEARNING A LINEAR CLASSIFIER AS AN OPTIMIZATION PROBLEM

- Structural risk minimization framework
  - Tradeoff between low training error and a solution that is simple

$$\min_{w,b} \sum_n \mathbf{1}[y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 0] + \lambda R(\boldsymbol{w}, b)$$

Variables over which we are optimizing the objective function

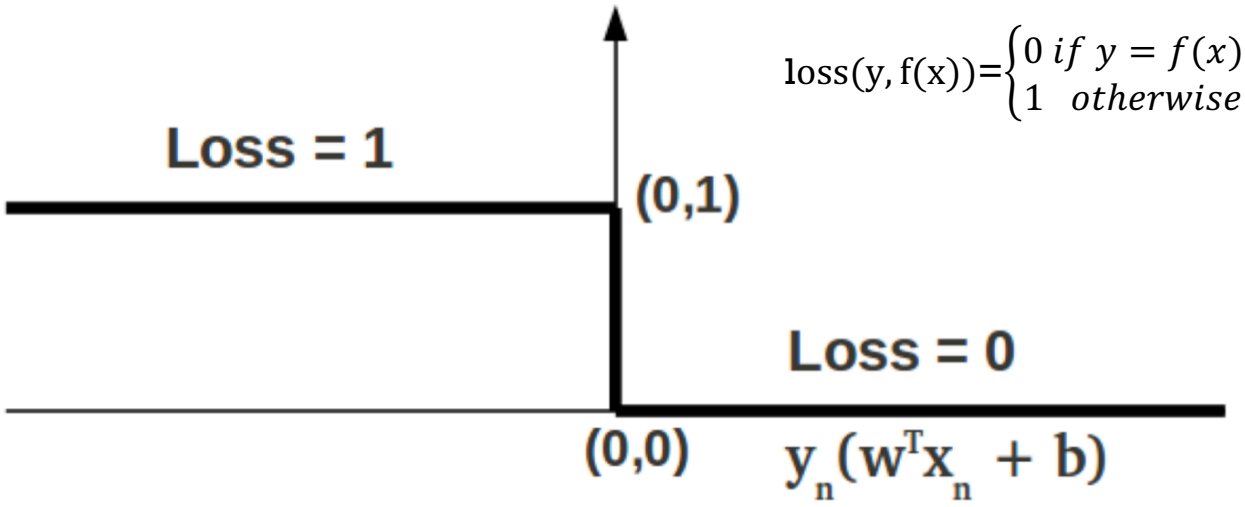
Objective function
In this case: simple 0/1 loss

Regularizing function
Helps with avoiding overfitting
Penalizes complex functions
λ is a hyperparameter

KEY QUESTIONS

- How can we adjust the optimization problem so that there *are* efficient algorithms for solving it?
- What are good regularizers $R(\boldsymbol{w}, b)$ for hyperplanes?
- Assuming we can adjust the optimization problem appropriately, what algorithms exist for efficiently solving this regularized optimization problem?

THE 0/1 LOSS FUNCTION



$$\text{loss}(y, f(x)) = \begin{cases} 0 \; if \; y = f(x) \\ 1 \quad otherwise \end{cases}$$

- Small changes in w,b can lead to big changes in the loss
- 0/1 loss is non-smooth, non-convex

ANNOUNCEMENTS

- Most regrading requests have been addressed
  - If we haven't, then submit another one through Regrade option on gradescope
- Mark your regrading requests on Piazza with tag "regrade" and "hw#" where # corresponds to the homework number
  - Easier for us to find
- Reminder: HW2 is due in a week
  - Highly encourage to take advantage of office hours

ACKNOWLEDGEMENTS

- These slides use materials by Marine Carpuat