Machine Learning Coms-4771

Reductions between Machine Learning Problems

Lecture 5

Basic difficulty: The problem you want to solve is not the problem solved by standard algorithms

Solutions:

1. Design new algorithms
2. Think about how to reuse old ones

This lecture is about the mathematics of 2.

Asymmetric Binary Classification

Homework problem 1: binary problem, but false negatives and false positives have very different costs (diagnosis, spam filtering)

More generally, there may be a different cost associated with mislabeling each example:

- Distribution D over $X \times Y \times C$, where X is some feature space, $Y = \{0, 1\}$ is the binary label, $C \subset [0, \infty)$ is the mislabeling cost.
- Goal: learn a classifier $h : X \to Y$ minimizing the expected cost

$$\mathbf{E}_{(x,y,c)\sim D}[c \cdot \mathbf{1}(h(x) \neq y)].$$

- The costs of test examples are generally not available. If they are, they can be embedded into x.

k-class classification

- Distribution D over $X \times Y$, where $Y = \{1, \ldots, k\}$.
- Goal: learn a classifier $h : X \to Y$ minimizing the multiclass error rate

$$\mathbf{E}_{(x,y)\sim D}\mathbf{1}(h(x) \neq y)$$

Cost-sensitive k-class: Each prediction has a different associated loss.

- Distribution D over $X \times [0, \infty)^k$, where a vector in $[0, \infty)^k$ specifies the cost of each of the k choices.
- Goal: find a classifier $h : X \to \{1, \ldots, k\}$ minimizing the expected cost

$$\mathbf{E}_{(x,c)\sim D}[c_{h(x)}].$$

  The costs are generally not available at test time.
- Very general: can encode *any* loss function on individual examples.

Example of a cost-sensitive problem

Problem: Predict which route to take to Columbia.
Three routes:

- Route A: takes 30 minutes with probability 1/2, and 2 hours with probability 1/2. (Expected time 1 hour 15 minutes.)
- Route B: takes 2 hours whenever A takes 30 minutes, and 30 minutes whenever A takes 2 hours.
- Route C: takes 45 minutes every time. (Expected time 45 minutes.)

Is it a 3-class classification problem (predicting the best route)?

No: Ignoring costs makes the problem harder.
The 3-class classification problem is very noisy (noise rate 1/2) and route C (best expected time) is never chosen!

Reductions are easy (= you can do them too)

Basic problem:

- We want to solve a problem defined by a distribution D and loss function ℓ.
- We have a learning algorithm $\mathcal{A}$ for optimizing another loss function ℓ' (on any distribution we feed it).

Basic idea:

- Design goal: Transform D into another distribution D' (by transforming samples) so that a solution minimizing ℓ' on D' can be translated into a solution minimizing ℓ on the original distribution D.
- Transform your problem into a problem $\mathcal{A}$ knows how to solve, apply $\mathcal{A}$ to get a solution, and then transform this solution into your final solution.
- If $\mathcal{A}$ does well on the induced problem, our solution is guaranteed to do well on the original problem.

Why reductions?

- They work.
- Can reuse highly optimized learning algorithms and code. Plug and play: N different algorithms for a simple problem + one reduction = N different algorithms for a complex problem.
- For a number of learning algorithms (e.g., neural networks, SVMs), it's not so easy to design good extensions that deliver the efficiency of the binary algorithm.
- The theory of learning has focused mostly on simple problems (binary classification or regression)
- Reductions compose: much of science is about decomposing big problems into small subproblems, solving them, and composing the solutions.
- Many successful algorithms can be seen as reductions (e.g., boosting and bagging)

Reductions in Complexity Theory

"A reduces to B" can be interpreted as:

- If B is easy, so is A (maching learning).
- If A is hard, so is B (complexity theory).

Both statements are relative (relative hardness or relative easiness).
Absolute statements are hard!

From k-class to Binary

One-Against-All: create one binary problem for each of the k classes

- Training: The classifier for class i is trained to predict "Is the label i or not?" Example (x, y) is transformed into k examples of the form $(x, \mathbf{1}(y = i))$, for $i \in \{1, \ldots, k\}$.
- Testing: Evaluate all k classifiers and randomize over those which predict "yes" (or over all k labels if all answers are "no").

(draw a picture in class)

The One Classifier Trick

We learn k classifiers in parallel. The *could* be one classifier.

1. Let $S' = \bigcup_i \{(\langle x, i \rangle, \mathbf{1}(i = y)) : (x, y) \in S\}$
2. Run the binary learner on S' to get a classifier $h : X \times \{1, \ldots, k\} \to \{0, 1\}$.
3. Let $h_i(x) = h(\langle x, i \rangle)$.

Handy for analysis: we can think about drawing from the induced distribution D' by drawing $(x, y) \sim D$ and i uniformly in $\{1, \ldots, k\}$ to get a sample $(\langle x, i \rangle, \mathbf{1}(i = y))$ from D'.

One Against All (OAA): Error Efficiency

Theorem

For any distribution D over $X \times \{1, \ldots, k\}$ and any binary classifier $h : X \to \{0, 1\}$,

$$\underbrace{err_D(f)}_{\text{multiclass}} \leq (k-1) \underbrace{err_{D'}(h)}_{\text{binary}},$$

where $f(x) = \text{argmax}_i h(\langle x, i \rangle)$ is the multiclass classifier returned by OAA.

Proof.

Three failure modes:

- A false negative, no false positives: h induces an error rate of $(k-1)/k$ with a *single* binary mistake, so h's efficiency is $(k-1)/k$ per mistake (f is choosing randomly between k labels, only one of which is correct).
- $q > 0$ false positives, no false negative: h induces an error rate of $q/(q+1)$ with q binary mistakes, so the efficiency is $1/(q+1)$ per mistake.
- $q > 0$ false positives + false negative: f errs all the time, so h's efficiency in inducing errors is $1/(q+1)$ per mistake.

The maximum efficiency is $(k-1)/k$. Multiplying by k (since there are k opportunities to err), we get the theorem. □

One Against All (OAA)

Does anyone see a problem?

A false negative (predicting "no" when the correct label is "yes") is much more disastrous than a false positive. (Especially bad for noisy problems where there is no majority class i with $D(y = i \mid x) > 1/2$.)

Probability of correct prediction (if there are no other errors):

- False negative: $1/k$ (randomizing over all labels)
- False positive: $1/2$

Here is where your homework problem comes in!

Compose two reductions: Multiclass $\xrightarrow{(1)}$ asymmetric binary $\xrightarrow{(2)}$ binary

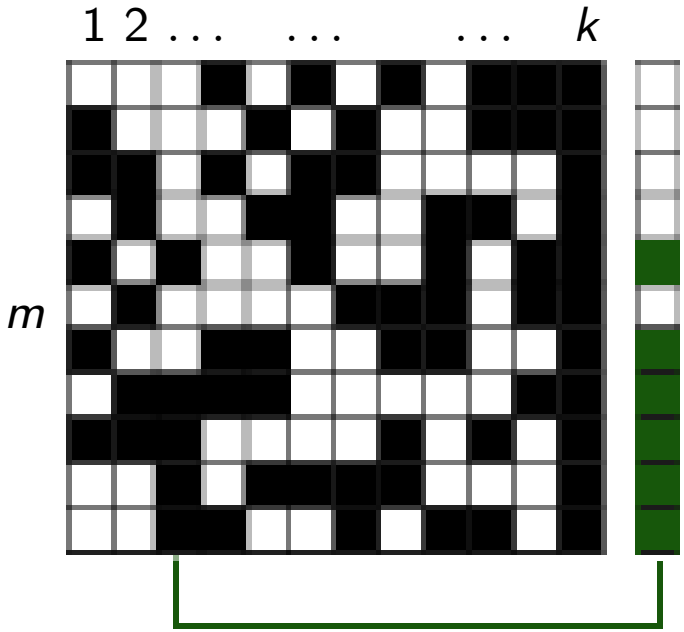
The best known statement for (2):

$$\text{Asymmetric error} \leq \text{binary error} \cdot \text{expected cost}$$

Composing (1) + (2) with costs $c_0 = k/2$ and $c_1 = k - 1$ gives a factor of 2 improvement:

$$\text{Multiclass error} \leq \frac{k}{2} \cdot \text{binary error}$$

Error Correcting Output Codes: Multiclass to Binary



decoding the vector of predictions to the closest column

Each column corresponds to a class.

Every two columns differ in d positions.

Each row j defines a binary problem: "Is the label white or black?"

Example $(x, y) \mapsto m$ examples $(x, y, C(j, y))$, where $C(j, y)$ is 1 is the entry (j, y) is black and 0 otherwise ($j \in \{1, \ldots, m\}$).

Notice: OAA corresponds to ECOC with a diagonal matrix ($m = k$, $d = 2$).

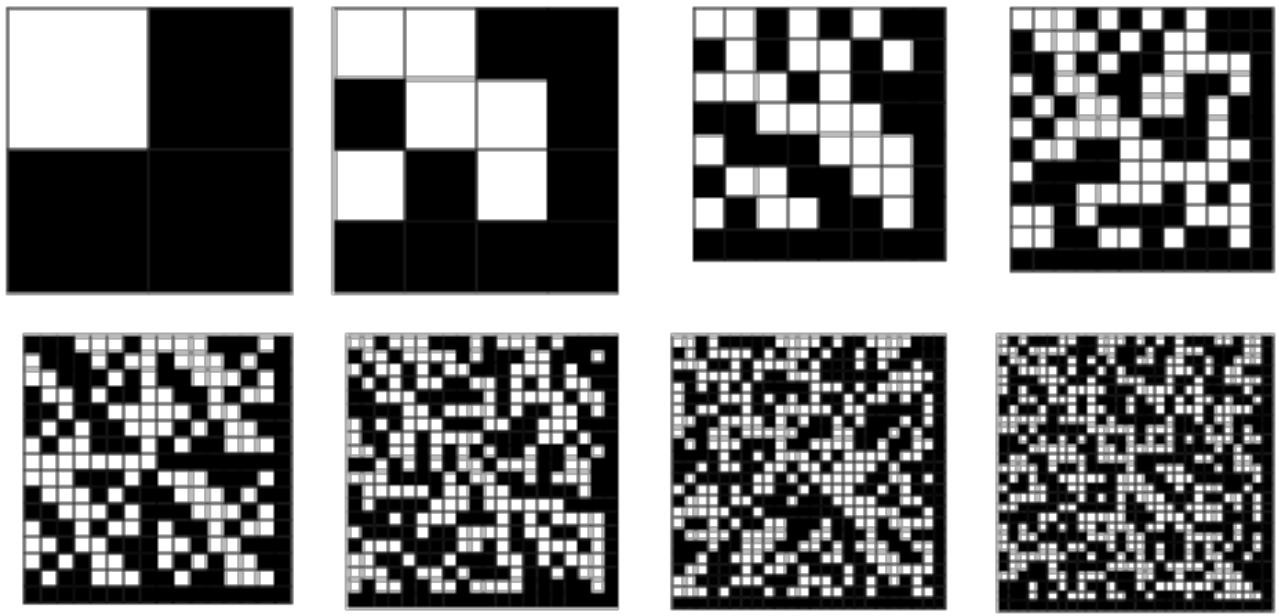
Theorem: For all multiclass problems, for all binary classifiers, ECOC has loss rate less than $2m\epsilon/d$, where ϵ is the average binary error rate.

Proof: At least $d/2$ binary classifiers must err to cause a multiclass error.

There are m opportunities to err, to ϵ binary loss can induce at most $\epsilon m/(d/2) = 2m\epsilon/d$ multiclass loss.

ECOC with Hadamard Matrices

In Hadamard Matrices, $d = m/2$, thus the multiclass loss rate is at most 4ϵ.
The number of rows m is the smallest power of 2 bigger than k, so $m \leq 2k$.



Recursive construction of H_{2k}:

$$H_{2k} = \begin{pmatrix} \overline{H_k} & H_k \\ H_k & H_k \end{pmatrix}$$

Regret

Concern: A statement about transformation of losses between problems may be vacuous if the created binary problems are hard. ECOC may create unnatural partitions which may be hard (i.e., no algorithm can do well). If $\epsilon = .25$, the theorem says that the multiclass error rate is at most 1 (not so useful).

Regret of classifier h on distribution D with respect to loss function ℓ is defined as

$$r_D(h) = \ell_D(h) - \underbrace{\min_{h'} \ell_D(h')}_{\text{Bayes error rate}}$$

Thus regret separates unremovable noise.

A more convincing result is a bound on the transformation of regrets instead of losses. (But this doesn't deal with a possible increase in the bias.)

Regret Transforms for Multiclass $\mapsto$ Binary

	Error	Regret
OAA with reweighting	$\sim \frac{k}{2}e$	none
All Pairs	$(k-1)e$	$(k-1)r$
Tree	$\lceil \log_2 k \rceil e$	none
ECOC	$4e$	none
Tournament methods	const $\cdot e$	const $\cdot r$

none = given a Bayes optimal binary classifier, the reduction doesn't produce an optimal multiclass classifier

- All-Pairs: learn a classifier for each pair of classes, to predict which of the two classes is more probable than the other. Multiclass predictions are made by running all pairwise classifiers and outputting the label maximizing the number of pairwise "wins", with ties broken arbitrarily.
- Tree: split the set of lables in half, learn a binary classifier to distinguish between the subsets; recurse. at test time, descend the tree.

Example showing why ECOC and OAA don't produce an optimal classifier, given an optimal binary classifier (for any $\delta > 0$):

Conditional probabilities of 3 classes			
$\frac{1}{2} - \delta$	$\frac{1}{4} + \frac{\delta}{2}$	$\frac{1}{4} + \frac{\delta}{2}$	optimal predictions
1	0	0	no
0	1	0	no
0	0	1	no

Randomizing over the three labels results in multiclass regret $\frac{2}{3}(\frac{1}{2} - \delta - (\frac{1}{4} + \frac{\delta}{2})) > \frac{1}{8}$

Final Thoughts

- Bagging (multiple runs of the same algorithm on the same learning problem are combined using voting): self-reduction of classification to classification, which can reduce variance (dependence on the exact training set) but not bias.
- Voting methods which run different algorithms on the same problem can decrease both bias and variance.
- OAA and ECOC apply the same learning algorithm but on different binary problems—can decrease both bias and variance (bias = mismatch between the problem and the algorithm).
- Boosting (coming up in a couple of lectures) = a reduction of strong learning to weak learning.

Conclusion

What else can we do with binary classification?

- Ranking
- Squared loss regression
- Quantile regression
- Structured prediction (e.g., for natural language processing)
- Dealing with partial feedback in cost-sensitive learning

See references at the blog/web-page.