Online Learning
And Other Cool Stuff

Your guide:
Avrim Blum
Carnegie Mellon University

[Machine Learning Summer School 2012]

Itinerary

• Stop 1: Minimizing regret and combining advice.
  - Randomized Wtd Majority / Multiplicative Weights alg
  - Connections to game theory

• Stop 2: Extensions
  - Online learning from limited feedback (bandit algs)
  - Algorithms for large action spaces, sleeping experts

• Stop 3: Powerful online LTF algorithms
  - Winnow, Perceptron

• Stop 4: Powerful tools for using these algorithms
  - Kernels and Similarity functions

• Stop 5: Something completely different
  - Distributed machine learning

Powerful tools for learning:
Kernels and Similarity Functions

2-minute version

• Suppose we are given a set of images , and want to learn a rule to distinguish men from women. Problem: pixel representation not so good.

• A powerful technique for such settings is to use a kernel: a special kind of pairwise function .

  ➢ Can think about & analyze kernels in terms of implicit mappings, building on margin analysis we just did for Perceptron (and similar for SVMs).

  ➢ Can also directly analyze directly as similarity functions, building on analysis we just did for Winnow. [Balcan-B’06] [Balcan-B-Srebro’08]

Kernel functions and Learning

• Back to our generic classification problem. E.g., given a set of images , labeled by gender, learn a rule to distinguish men from women. [Goal: do well on new data]

• Problem: our best algorithms learn linear separators, but might not be good for data in its natural representation.

  ➢ Old approach: use a more complex class of functions.

  ➢ More recent approach: use a kernel.
What's a kernel?

- A kernel $K$ is a legal def of dot-product: fn s.t. there exists an implicit mapping $\Phi_K$ such that $K(x,y) = \Phi_K(x) \cdot \Phi_K(y)$.
- E.g., $K(x,y) = (x \cdot y + 1)^d$.
- $\Phi_K: \mathbb{R}^n \to \mathbb{R}^d$.
- Point is: many learning algos can be written so only interact with data via dot-products.
  - E.g., Perceptron: $w = x^{(1)} + x^{(2)} - x^{(5)} + x^{(9)}$.
  - $w \cdot x = (x^{(1)} + x^{(2)} - x^{(5)} + x^{(9)}) \cdot x$.
- If replace $x \cdot y$ with $K(x,y)$, it acts implicitly as if data was in higher-dimensional $\Phi$-space.

Kernel should be pos. semi-definite (PSD)

- E.g., for the case of $n=2$, $d=2$, the kernel $K(x,y) = (1 + x \cdot y)^d$ corresponds to the mapping:

```
Example
```

Moreover, generalize well if good margin

- If data is lin. separable by margin $\gamma$ in $\Phi$-space, then need sample size only $\tilde{O}(1/\gamma^2)$ to get confidence in generalization.
  
  Assume $|\Phi(x)| \leq 1$.

- E.g., follows directly from mistake bound we proved for Perceptron.

- Kernels found to be useful in practice for dealing with many, many different kinds of data.

Defn satisfying (1) and (2):

- Say have a learning problem $P$ (distribution $D$ over examples labeled by unknown target $f$).
- Sim fn $K(x,y) \to [-1,1]$ is $(\epsilon,\gamma)$-good for $P$ if at least a $1-\epsilon$ fraction of examples $x$ satisfy:

```
Goal: notion of "good similarity function"
```

for a learning problem that...

1. Talks in terms of more intuitive properties (no implicit high-diml spaces, no requirement of positive-semidefiniteness, etc)
2. If $K$ satisfies these properties for our given problem, then has implications to learning
3. Includes usual notion of "good kernel" (one that induces a large margin separator in $\Phi$-space).

```
Moreover, generalize well if good margin
```

But there is a little bit of a disconnect...

- In practice, kernels constructed by viewing as a measure of similarity: $K(x,y) \in [-1,1]$, with some extra reqts.
  - $K(x,y) = \Phi(x) \cdot \Phi(y)$.
  - Can we give an explanation for desirable properties of a similarity function that doesn't use implicit spaces?
  - And even remove the PSD requirement?

```
Moreover, generalize well if good margin
```

"most $x$ are on average more similar to points $y$ of their own type than to points $y$ of the other type"
Defn satisfying (1) and (2):
• Say have a learning problem $P$ (distribution $D$ over examples labeled by unknown target $f$).
• Sim fn $K: (x,y) \rightarrow [-1,1]$ is $(\varepsilon, \gamma)$-good for $P$ if at least a $1-\varepsilon$ fraction of examples $x$ satisfy:

\[
E_{y \sim D}[K(x,y) | \ell(y) = \ell(x)] \geq E_{y \sim D}[K(x,y) | \ell(y) \neq \ell(x)] + \gamma
\]

Note: it’s possible to satisfy this and not be PSD.

How to use it
At least a $1-\varepsilon$ prob mass of $x$ satisfy:

\[
E_{y \sim D}[K(x,y) | \ell(y) = \ell(x)] \geq E_{y \sim D}[K(x,y) | \ell(y) \neq \ell(x)] + \gamma
\]

Algorithm
• Draw sets $S^+$, $S^-$ of positive and negative examples.
• Classify $x$ based on average similarity to $S^+$ versus to $S^-$. 

Theorem If $|S^+|$ and $|S^-|$ are $\Omega(1/\varepsilon^2 \ln(1/\delta^2))$, then with probability $\geq 1-\delta$, error $\leq \varepsilon + \delta$.

But not broad enough
• $K(x,y) = x \cdot y$ has good separator but doesn’t satisfy defn. (half of positives are more similar to negs that to typical pos)
Broader defn...

- Ask that exists a set $R$ of "reasonable" $y$ (allow probabilistic) s.t. almost all $x$ satisfy

$$E_y[K(x,y)|\ell(y)\neq\ell(x),R(y)] > E_y[K(x,y)|\ell(y)=\ell(x),R(y)] + \gamma$$

- Formally, say $K$ is $(\epsilon,\gamma)$.good if have hinge-loss $\epsilon$, and $Pr(R_x,Pr(R_x) \geq \tau_x$.

- Claim 1: this is a legitimate way to think about good (large margin) kernels:
  - If $\gamma$-good kernel then $(\tau,\gamma^2)$-good here.
  - If $\gamma$-good here and PSD then $\gamma$-good kernel

How to use such a sim fn?

- Ask that exists a set $R$ of "reasonable" $y$ (allow probabilistic) s.t. almost all $x$ satisfy

$$E_y[K(x,y)|\ell(y)\neq\ell(x),R(y)] > E_y[K(x,y)|\ell(y)=\ell(x),R(y)] + \gamma$$

- Draw $S = \{y_1, ..., y_n\}$, $n=1/(\gamma^2 \tau)$ — could be unlabeled
- View as "landmarks", use to map new data:
  $$F(x) = [K(x,y_1), ..., K(x,y_n)]$$

- Whp, exists separator of good $L_1$ margin in this space: $w^* = [0,0,1/n,1/n,0,0,0,-1/n,0,0]$ — could be unlabeled

- So, take new set of examples, project to this space, and run good $L_1$ alg (e.g., Winnow)

Learning with Multiple Similarity Functions

- Let $K_1, ..., K_r$ be similarity functions s. t. some (unknown) convex combination of them is $(\epsilon,\gamma)$-good.

Algorithm

- Draw $S = \{y_1, ..., y_n\}$ set of landmarks. Concatenate features.
  $$F(x) = [K_1(x,y_1), ..., K_1(x,y_n), ..., K_r(x,y_1), ..., K_r(x,y_n)]$$

- Run some $L_1$ optimization algorithm as before (or Winnow) in this new feature space.

Broader defn...

- Ask that exists a set $R$ of "reasonable" $y$ (allow probabilistic) s.t. almost all $x$ satisfy

$$E_y[K(x,y)|\ell(y)\neq\ell(x),R(y)] > E_y[K(x,y)|\ell(y)=\ell(x),R(y)] + \gamma$$

- Formally, say $K$ is $(\epsilon,\gamma)$.good if have hinge-loss $\epsilon$, and $Pr(R_x,Pr(R_x) \geq \tau_x$.

- Claim 2: even if not PSD, can still use for learning.
  - So, don't need to have implicit-space interpretation to be useful for learning.
  - But, maybe not with SVM/Perceptron directly...

How to use such a sim fn?

If $K$ is $(\epsilon,\gamma)$.good, then can learn to error $\epsilon' = O(\epsilon)$ with $O((1/(\epsilon' y^2)) \log(n))$ labeled examples.

$$\text{minimize} \sum_{i=1}^n \left[ 1 - \sum_{j=1}^n \alpha_j f(x_i) K(x_i, x_j) \right] +$$

$$\text{s.t.} \sum_{j=1}^n [\alpha_j] \leq 1/\gamma$$

- Whp, exists separator of good $L_1$ margin in this space: $w^* = [0,0,1/n,1/n,0,0,0,-1/n,0,0]$ — could be unlabeled

- So, take new set of examples, project to this space, and run good $L_1$ alg (e.g., Winnow)

Learning with Multiple Similarity Functions

- Let $K_1, ..., K_r$ be similarity functions s. t. some (unknown) convex combination of them is $(\epsilon,\gamma)$-good.

Algorithm

- Draw $S = \{y_1, ..., y_n\}$ set of landmarks. Concatenate features.
  $$F(x) = [K_1(x,y_1), ..., K_1(x,y_n), ..., K_r(x,y_1), ..., K_r(x,y_n)]$$

Guarantee: Whp the induced distribution $F(P)$ in $R^n$ has a separator of error $\leq \epsilon + \delta$ at $L_1$ margin at least $t/4$.

Sample complexity is roughly: $O((1/(\epsilon' y^2)) \log(nr))$

Only increases by log($r$) factor!
Learning with Multiple Similarity Functions

- Interesting fact: because property defined in terms of $L_1$, no change in margin.
  - Only $\log(r)$ penalty for concatenating feature spaces.
  - If $L_2$, margin would drop by factor $r^{1/2}$, giving $O(r)$ penalty in sample complexity.
- Algorithm is also very simple (just concatenate).

Applications/extensions

  - If use directly this way rather than converting to PSD kernel, comparable performance and models much sparser. (They use $L_1$-normalized SVM).
- Bellet, A.; Habrard, A.; Sebban, M. MLJ 2012, ICML 2012: efficient algorithms for learning $(\epsilon,\gamma,\tau)$-good similarity functions in different contexts.

Summary

- Kernels and similarity functions are powerful tools for learning.
  - Can analyze kernels using theory of $L_2$ margins, plug in to Perceptron or SVM
  - Can also analyze more general similarity fns (not nec. PSD) without implicit spaces, connecting with $L_1$ margins and Winnow, $L_1$-SVM.
  - Second notion includes 1st notion as well (modulo some loss in parameters).
  - Potentially other interesting suffic. conditions too. E.g., [WangYangFeng07] motivated by boosting.

Itinerary

- Stop 1: Minimizing regret and combining advice.
  - Randomized Wtd Majority / Multiplicative Weights alg
  - Connections to game theory
- Stop 2: Extensions
  - Online learning from limited feedback (bandit alg)
  - Algorithms for large action spaces, sleeping experts
- Stop 3: Powerful online LTF algorithms
  - Winnow, Perceptron
- Stop 4: Powerful tools for using these algorithms
  - Kernels and Similarity functions
- Stop 5: Something completely different
  - Distributed machine learning

Distributed PAC Learning

Distributed Learning

Many ML problems today involve massive amounts of data distributed across multiple locations.

Maria-Florina Balcan
Avrim Blum
Shai Fine
Yishay Mansour

Georgia Tech
CMU
IBM
Tel-Aviv

[In COLT 2012]
Distributed Learning
Many ML problems today involve massive amounts of data distributed across multiple locations.

Click data

Customer data

Scientific data

Each has only a piece of the overall data pie

In order to learn over the combined D, holders will need to communicate.

Classic ML question: how much data is needed to learn a given class of functions well?
Distributed Learning
Many ML problems today involve massive amounts of data distributed across multiple locations.

These settings bring up a new question: how much communication? Plus issues like privacy, etc.

The distributed PAC learning model
- Goal is to learn unknown function $f \in C$ given labeled data from some distribution $D$.
- However, $D$ is arbitrarily partitioned among $k$ entities (players) $1, 2, \ldots, k$. ($k=2$ is interesting)

Players can sample $(x, f(x))$ from their own $D_i$.

The distributed PAC learning model
Assume learning a class $C$ of VC-dimension $d$.
Some simple baselines. [viewing $k \ll d$]
- Baseline #1: based on fact that can learn any class of VC-dim $d$ to error $\epsilon$ from $O(d/\epsilon \log 1/\epsilon)$ samples
  - Each player sends $1/k$ fraction to player 1.
  - Player 1 finds consistent $h \in C$, whp has error $\leq \epsilon$ with respect to $D$. Sends $h$ to others.
  - Total: 1 round, $O(d/\epsilon \log 1/\epsilon)$ examples communcate.
The distributed PAC learning model

- **Baseline #2:**
  - Suppose \( C \) is learnable by an online algorithm \( A \) with mistake-bound \( M \).
  - Player 1 runs \( A \), broadcasts current hypothesis.
  - If any player has a counterexample, sends to player 1. Player 1 updates, re-broadcasts.
  - At most \( M \) examples and hypotheses communicated.

Dependence on \( 1/\epsilon \)

- Had linear dependence in \( d \) and \( 1/\epsilon \), or \( M \) and no dependence on \( 1/\epsilon \).

- Can you get \( O(d \log 1/\epsilon) \) examples of communication?
  - Yes! Distributed boosting.

Recap of Adaboost

- Weak learning algorithm \( A \).
  - For \( t=1,2, \ldots, T \)
    - Construct \( D_t \) on \( \{x_1, \ldots, x_m\} \)
    - Run \( A \) on \( D_t \) producing \( h_t \)
  - \( D_1 \) uniform on \( \{x_1, \ldots, x_m\} \)
  - \( D_t+1 \) increases weight on \( x_i \) if \( h_t \) makes a mistake on \( x_i \); decreases it on \( x_i \) if \( h_t \) correct.

Key points:
- \( D_{t+1}(x_i) \) depends on \( h_1(x_i), \ldots, h_t(x_i) \) and normalization factor that can be communicated efficiently.
- To achieve weak learning it suffices to use \( O(d) \) examples.

Distributed Adaboost

- Each player has a sample \( S_i \) from \( D_i \).
  - For \( t=1,2, \ldots, T \)
    - Each player sends player 1, enough data to produce hypothesis \( h_t \) of error \( 1/2 \). (For \( t=1, O(d/k) \) examples each.)
    - Player 1 broadcasts \( h_t \) to all other players.
    - Each player reweights its own distribution on \( S_i \) using \( h_t \) and sends the sum of its weights \( w_{i,t} \) to player 1.
      - \( h_t \) may do better on some than others
    - Player 1 determines the #of samples to request next from each \( i \) (samples \( O(d) \) times from the multinomial given by \( w_{i,t}/W_t \)).

Final result:
- \( O(d) \) examples of communication per round
  - \( O(k \log d) \) extra bits to send weights & request
  - 1 hypothesis sent per round
  - \( O(\log 1/\epsilon) \) rounds of communication.
- So, \( O(d \log 1/\epsilon) \) examples of communication in total plus low order extra info.

Agnostic learning

- Recent result of [Balcan-Hanneke] gives robust halving alg that can be implemented in distributed setting.
  - Get error \( 2 \text{OPT}(C) + \epsilon \) using total of only \( O(k \log|C| \log(1/\epsilon)) \) examples.
  - Not computationally efficient in general, but says \( O(\log(1/\epsilon)) \) possible in principle.
Can we do better for specific classes of interest?
E.g., conjunctions over \( \{0,1 \}^d \). \( f(x) = x_2 x_5 x_9 x_{15} \)
• These generic methods give \( O(d) \) examples, or \( O(d^2) \) bits total. Can you do better?
• Again, thinking of \( k \ll d \).

Can we do better for specific classes of interest?
E.g., conjunctions over \( \{0,1 \}^d \). \( f(x) = x_2 x_5 x_9 x_{15} \)
• These generic methods give \( O(d) \) examples, or \( O(d^2) \) bits total. Can you do better?
• Sure: each entity intersects its positives. Sends to player 1.
• Player 1 intersects & broadcasts.

Can we do better for specific classes of interest?
E.g., conjunctions over \( \{0,1 \}^d \). \( f(x) = x_2 x_5 x_9 x_{15} \)
• These generic methods give \( O(d) \) examples, or \( O(d^2) \) bits total. Can you do better?

Only \( O(k) \) examples sent. \( O(kd) \) bits.

Can we do better for specific classes of interest?
General principle: can learn any intersection closed class (well-defined "tightest wrapper" around positives) this way.

Interesting class: parity functions
Examples \( x \in \{0,1 \}^d \). \( f(x) = x \cdot v_f \mod 2 \), for unknown \( v_f \).
• Interesting for \( k=2 \).
• Classic communication LB for determining if two subspaces intersect.
• Implies \( O(d^2) \) bits LB for proper learning.
• What if we allow hyps that aren’t parities?

Interesting class: parity functions
Examples \( x \in \{0,1 \}^d \). \( f(x) = x \cdot v_f \mod 2 \), for unknown \( v_f \).
• Parity has interesting property that:
  (a) Can be properly PAC-learned. [Given dataset \( S \) of size \( O(d^2) \), just solve the linear system]
  (b) Can be non-properly learned in reliable-useful model of Rivest–Sloan’88. [if \( x \) in subspace spanned by \( S \), predict accordingly, else say ??]
**Interesting class: parity functions**

Examples $x \in \{0,1\}^d$. $f(x) = x \cdot v_f \mod 2$, for unknown $v_f$.

- **Algorithm:**
  - Each player $i$ properly PAC-learns over $D_i$ to get parity function $g_i$. Also improperly R-U learns to get rule $h_i$. Sends $g_i$ to other player.
  - Uses rule: "if $h_i$ predicts, use it; else use $g_{3i}$.
  - Can one extend to $k=3$??

**Linear Separators**

Thm: Over any non-concentrated $D$ [density bounded by $c \cdot \text{unif}$], can achieve #vectors communicated of $O((d \log d)^{1/2})$ rather than $O(d)$ (for constant $k$, $\epsilon$).

- **Algorithm:**
  - Run a margin-version of perceptron in round-robin.
    - Player $i$ receives $h$ from prev player.
    - If $\text{err}(h) \geq \epsilon$ on $D_i$, then update until $f(x)(w \cdot x) \geq 1$ for most $x$ from $D_i$.
    - Then pass to next player.

**Proof idea:**

- Non-concentrated $D \Rightarrow$ examples nearly-orthogonal whp
  
  \[|\text{cos}(x,x')| = O((\log(d)/d)^{1/2})\]

- So updates by player $j$ don’t hurt $i$ too much: after player $i$ finishes, if less than $(d/\log(d))^{1/2}$ updates by others, player $i$ is still happy.

- Implies at most $O((d \log d)^{1/2})$ rounds.

**Conclusions and Open Questions**

As we move to large distributed datasets, communication becomes increasingly crucial.

- Rather than only ask “how much data is needed to learn well”, we ask “how much communication do we need?”
- Also issues like privacy become more central.
  (Didn’t discuss here, but see paper)

**Open questions:**

- Linear separators of margin $\gamma$ in general?
- Other classes? [parity with $k=3$?]
- Incentives?