15-859(M) Randomized Algorithms

An Intro to Machine Learning

Avrim Blum

Lecture #12

Plan for today

- Machine Learning intro: models and basic issues
- How much data do I need to see to be confident in generalizations I make from it?
- Connections of this to notion of Occam's razor
- A cool idea: "shatter coefficients", VC-dimension, and a very nice probabilistic argument.

Plan for Monday

- An interesting algorithm for online decision making. Problem of "combining expert advice"
- Algorithms for online decision making from very limited feedback. The "multi-armed bandit problem"

Machine learning can be used to...

- recognize speech,
- identify patterns in data,
- steer a car,
- play games,
- adapt programs to users,
- improve web search, ...

From a scientific perspective: can we develop models to understand learning as a computational problem, and what types of guarantees might we hope to achieve?

A typical setting

- Imagine you want a computer program to help filter which email messages are spam and which are important.
- Might represent each message by n features. (e.g., return address, keywords, spelling, etc.)
- Take sample S of data, labeled according to whether they were/weren't spam.
- Goal of algorithm is to use data seen so far produce good prediction rule (a "hypothesis") *h(x)* for future data.

The concept learning setting

E.g.,

money	pills	Mr.	bad spelling	known-sender	spam?

The concept learning setting

E.g.,

money	pills	Mr.	bad spelling	known-sender	spam?
Y	N	Y	Y	N	Y
N	N	N	Y	Y	N
N	Y	N	N	N	Y
Y	N	N	N	Y	N
N	N	Y	N	Y	N
Y	N	N	Y	N	Y
N	N	Y	N	N	N
N	Y	N	Y	N	Y

a positive example

a negative example

Given data, some reasonable rules might be:

- Predict SPAM if ¬known AND (money OR pills)
- Predict SPAM if money + pills – known > 0.
- ...

Big questions

(A) How might we automatically generate rules that do well on observed data?

[algorithm design]

(B) What kind of confidence do we have that they will do well in the future?

[confidence bound / sample complexity]

for a given learning alg, how much data do we need...

Natural formalization (PAC)

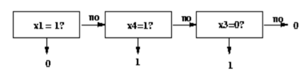
Email msg

Spam or not?

- We are given sample S = {(x,y)}.
  - View labels y as being produced by some target function f.
- Alg does optimization over S to produce some hypothesis (prediction rule) h.
- Assume S is a random sample from some probability distribution D. Goal is for h to do well on new examples also from D.

I.e., $Pr_D[h(x) \neq f(x)] < \varepsilon$.

Example of analysis: Decision Lists



Say we suspect there might be a good prediction rule of this form.

1. Design an efficient algorithm A that will find a consistent DL if one exists.
2. Show that if sample S is of reasonable size, Pr[exists consistent DL h with err(h) > ε] < δ.
3. This means that A is a good algorithm to use if f is, in fact, a DL.

(a bit of a toy example since would want to extend to "mostly consistent" DL)

How can we find a consistent DL?

x_1	x_2	x_3	x_4	x_5	label
1	0	0	1	1	+
~~0~~	~~1~~	~~1~~	~~0~~	~~0~~	~~–~~
~~1~~	~~1~~	~~1~~	~~0~~	~~0~~	~~+~~
~~0~~	~~0~~	~~0~~	~~1~~	~~0~~	~~–~~
~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~+~~
1	0	0	0	1	–

if (x_1=0) then -, else
if (x_2=1) then +, else
if (x_4=1) then +, else -

Decision List algorithm

- Start with empty list.
- Find if-then rule consistent with data.
  (and satisfied by at least one example)
- Put rule at bottom of list so far, and cross off examples covered. Repeat until no examples remain.

If this fails, then:

- No rule consistent with remaining data.
- So no DL consistent with remaining data.
- So, no DL consistent with original data.

OK, fine. Now why should we expect it to do well on future data?

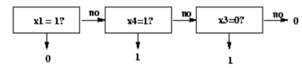
Confidence/sample-complexity

- Consider some DL h with err(h)>ε, that we're worried might fool us.
- Chance that h survives |S| examples is at most $(1-\varepsilon)^{|S|}$.
- Let |H| = number of DLs over n Boolean features. |H| < (4n+2)! . (really crude bound)

So, Pr[some DL h with err(h)>ε is consistent] < $|H|(1-\varepsilon)^{|S|}$.

- This is <0.01 for |S| > $(1/\varepsilon)[\ln(|H|) + \ln(100)]$ or about $(1/\varepsilon)[n \ln n + \ln(100)]$

Example of analysis: Decision Lists



Say we suspect there might be a good prediction rule of this form.

1. Design an efficient algorithm A that will find a consistent DL if one exists. DONE
2. Show that if |S| is of reasonable size, then Pr[exists consistent DL h with err(h) > ε] < δ. DONE
3. So, if f is in fact a DL, then whp A's hypothesis will be approximately correct. "PAC model"

Confidence/sample-complexity

- What's great is there was nothing special about DLs in our argument.
- All we said was: "if there are not *too* many rules to choose from, then it's unlikely one will have fooled us just by chance."
- And in particular, the number of examples needs to only be proportional to log(|H|).

  (the "log" is important here)

Occam's razor

William of Occam (~1320 AD):

> "entities should not be multiplied unnecessarily" (in Latin)

Which we interpret as: "in general, prefer simpler explanations".

Why? Is this a good policy? What if we have different notions of what's simpler?

Occam's razor (contd)

A computer-science-ish way of looking at it:

- Say "simple" = "short description".
- At most 2^s explanations can be < s bits long.
- So, if the number of examples satisfies:

$$m > (1/\varepsilon)[s \ln(2) + \ln(100)]$$

Think of as 10x #bits to write down h.

Then it's unlikely a bad simple explanation will fool you just by chance.

Occam's razor (contd)^2

Nice interpretation:

- Even if we have different notions of what's simpler (e.g., different representation languages), we can both use Occam's razor.
- Of course, there's no guarantee there will be a short explanation for the data. That depends on your representation.

Extensions

We said: if $|S| \geq (1/\varepsilon)[\ln(|H|) + \ln(1/\delta)]$, then with probability $\geq 1-\delta$, all $h \in H$ with $err_D(h) \geq \varepsilon$ have $err_S(h) > 0$.

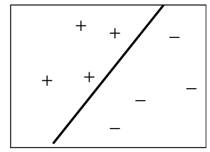
What if no perfect rule, and best we find is rule with error (say) 10% on training set? What can we say?

Thm: If $|S| \geq (1/(2\varepsilon^2))[\ln(|H|) + \ln(2/\delta)]$, then with prob $\geq 1-\delta$, all $h \in H$ have $|err_D(h) - err_S(h)| < \varepsilon$.

Proof: apply Hoeffding bounds.
- Chance of failure at most $2|H|e^{-2|S|\varepsilon^2}$.
- Set to δ and solve.

One more extension

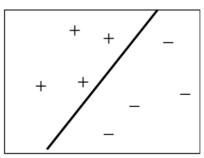
- What about something like the class H of linear separators? What is |H|?



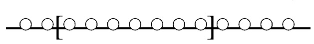
  - There are infinitely many linear separators, but not that many really different ones.
  - Union bound is too weak.

A cool idea: shatter coefficient

- Let H[S] be the number of ways of splitting set S using functions in H.
- Let $H[m] = \max_{|S|=m} H[S]$.

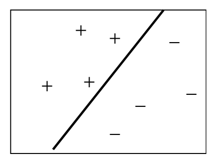


- E.g., linear separators in R^d: $H[m] = O(m^d)$.
- E.g., intervals on a line: $H[m] = O(m^2)$.



A cool idea: shatter coefficient

- Let H[S] be the number of ways of splitting set S using functions in H.
- Let $H[m] = \max_{|S|=m} H[S]$.



- E.g., linear separators in R^d: $H[m] = O(m^d)$.

Thm: if $m=|S| \geq (2/\varepsilon)[\lg(2H[2m]) + \lg(1/\delta)]$, then with probability $\geq 1-\delta$, all $h \in H$ with $err_D(h) \geq \varepsilon$ have $err_S(h) > 0$.

A cool idea: shatter coefficient

Thm: if $m \geq (2/\varepsilon)[\lg(2H[2m]) + \lg(1/\delta)]$, then with probability $\geq 1-\delta$, all $h \in H$ with $err_D(h) \geq \varepsilon$ have $err_S(h) > 0$.

Note 1: For linear separators in R^d, $H[2m] = O(m^d)$, so bound is $O(1/\varepsilon)[d \lg(1/\varepsilon) + \lg(1/\delta)]$

Note 2: VC-dimension(H) = max value m such that $H[m] = 2^m$

Sauer's lemma: $H[m] = O(m^{VCdim(H)})$.

A cool idea: shatter coefficient

Thm: if $m \geq (2/\varepsilon)[\lg(2H[2m]) + \lg(1/\delta)]$, then with probability $\geq 1-\delta$, all $h \in H$ with $err_D(h) \geq \varepsilon$ have $err_S(h) > 0$.

Proof of Thm:
- Consider drawing 2 sets S, S' of m examples each.
- Let A be the event: exists $h \in H$ with $err_D(h) \geq \varepsilon$ and $err_S(h) = 0$.
- Let B be the event: exists $h \in H$ with $err_{S'}(h) \geq \varepsilon/2$ and $err_S(h) = 0$.
- Claim 1: $\Pr[A]/2 \leq \Pr[B]$ (because $\Pr[B|A] \geq \frac{1}{2}$)
- So, just need to show Pr[B] is low.

A cool idea: shatter coefficient

Thm: if $m \geq (2/\varepsilon)[\lg(2H[2m]) + \lg(1/\delta)]$, then with probability $\geq 1-\delta$, all $h \in H$ with $err_D(h) \geq \varepsilon$ have $err_S(h) > 0$.

Proof cont'd:

- Consider drawing 2 sets S, S' of m examples each.
- Let B be the event: exists $h \in H$ with $err_{S'}(h) \geq \varepsilon/2$ and $err_S(h)=0$. Suffices to show Pr[B] is low.
- Now, define T, T' as follows:
  - For i=1 to m, flip a fair coin:
    - If heads, put ith element of S into T and ith element of S' into T'.
    - If tails, do it other way around.

A cool idea: shatter coefficient

Thm: if $m \geq (2/\varepsilon)[\lg(2H[2m]) + \lg(1/\delta)]$, then with probability $\geq 1-\delta$, all $h \in H$ with $err_D(h) \geq \varepsilon$ have $err_S(h) > 0$.

Proof cont'd:

- Consider drawing 2 sets S, S' of m examples each.
- Let *C* be the event: exists $h \in H$ with $err_{T'}(h) \geq \varepsilon/2$ and $err_T(h)=0$. Suffices to show Pr[*C*] is low.
- Now, define T, T' as follows:
  - For i=1 to m, flip a fair coin:
    - If heads, put ith element of S into T and ith element of S' into T'.
    - If tails, do it other way around.

A cool idea: shatter coefficient

Thm: if $m \geq (2/\varepsilon)[\lg(2H[2m]) + \lg(1/\delta)]$, then with probability $\geq 1-\delta$, all $h \in H$ with $err_D(h) \geq \varepsilon$ have $err_S(h) > 0$.

Proof cont'd:

- Will show that for all S,S', $Pr_{swap}[C]$ is low.
- Let *C* be the event: exists $h \in H$ with $err_{T'}(h) \geq \varepsilon/2$ and $err_T(h)=0$. Suffices to show Pr[*C*] is low.
- Now, define T, T' as follows:
  - For i=1 to m, flip a fair coin:
    - If heads, put ith element of S into T and ith element of S' into T'.
    - If tails, do it other way around.

A cool idea: shatter coefficient

Thm: if $m \geq (2/\varepsilon)[\lg(2H[2m]) + \lg(1/\delta)]$, then with probability $\geq 1-\delta$, all $h \in H$ with $err_D(h) \geq \varepsilon$ have $err_S(h) > 0$.

Proof cont'd:

- Will show that for all S,S', $Pr_{swap}[C]$ is low.
- Let *C* be the event: exists $h \in H$ with $err_{T'}(h) \geq \varepsilon/2$ and $err_T(h)=0$. Suffices to show Pr[*C*] is low.
- Fix some splitting h of $S \cup S'$ (at most H[2m])
- If for any i, h makes mistake on ith element of both S and S', then $Pr[C_h]=0$. Also, if h makes fewer than $\varepsilon m/2$ mistakes on $S \cup S'$, then $Pr[C_h]=0$.
- Else, $Pr[C_h] \leq 2^{-\varepsilon m/2}$. Set $H[2m] \times 2^{-\varepsilon m/2} = \delta/2$. Done!

Online learning

- What if we don't want to make assumption that data is coming from some fixed distribution? Or any assumptions on data?
- Can no longer talk about past performance predicting future results.
- Can we ... esting at all??

Idea: re...
➢Show t... rly as well as best p... s.

Using "expert" advice

Say we want to predict the stock market.

- We solicit n "experts" for their advice. (Will the market go up or down?)
- We then want to use their advice somehow to make our prediction. E.g.,

Expt 1	Expt 2	Expt 3	neighbor's dog	truth
down	up	up	up	up
down	up	up	down	down
…	…	…	…	…

Basic question: Is there a strategy that allows us to do nearly as well as best of these in hindsight?

["expert" = someone with an opinion. Not necessarily someone who knows anything.]

Simpler question

- We have n “experts”.
- One of these is perfect (never makes a mistake). We just don’t know which one.
- Can we find a strategy that makes no more than lg(n) mistakes?

Answer: sure. Just take majority vote over all experts that have been correct so far.

➢Each mistake cuts # available by factor of 2.

➢Note: this means ok for n to be very large.

What if no expert is perfect?

Intuition: Making a mistake doesn't completely disqualify an expert. So, instead of crossing off, just lower its weight.

Weighted Majority Alg:

- Start with all experts having weight 1.
- Predict based on weighted majority vote.
- Penalize mistakes by cutting weight in half.

					prediction	correct
weights	1	1	1	1		
predictions	Y	Y	Y	N	Y	Y
weights	1	1	1	.5		
predictions	Y	N	N	Y	N	Y
weights	1	.5	.5	.5		

Analysis: do nearly as well as best expert in hindsight

- M = # mistakes we've made so far.
- m = # mistakes best expert has made so far.
- W = total weight (starts at n).
- After each mistake, W drops by at least 25%. So, after M mistakes, W is at most $n(3/4)^M$.
- Weight of best expert is $(1/2)^m$. So,

$$\begin{aligned}(1/2)^m &\leq n(3/4)^M \\ (4/3)^M &\leq n2^m \\ M &\leq 2.4(m + \lg n)\end{aligned}$$

So, if m is small, then M is pretty small too.

Randomized Weighted Majority

2.4(m + lg n) not so good if the best expert makes a mistake 20% of the time. Can we do better? Yes.

- Instead of taking majority vote, use weights as probabilities. (e.g., if 70% on up, 30% on down, then pick 70:30) Idea: smooth out the worst case.
- Also, generalize ½ to 1- ε.

Solves to: $M \leq \frac{-m\ln(1-\varepsilon)+\ln(n)}{\varepsilon} \approx (1+\varepsilon/2)m + \frac{1}{\varepsilon}\ln(n)$

M = expected #mistakes

$\boxed{M \leq 1.39m + 2\ln n \quad \leftarrow \varepsilon = 1/2}$

$\boxed{M \leq 1.15m + 4\ln n \quad \leftarrow \varepsilon = 1/4}$

$\boxed{M \leq 1.07m + 8\ln n \quad \leftarrow \varepsilon = 1/8}$

Analysis

- Say at time t we have fraction F_t of weight on experts that made mistake.
- So, we have probability F_t of making a mistake, and we remove an εF_t fraction of the total weight.
  - $W_{final} = n(1-\varepsilon F_1)(1 - \varepsilon F_2)\ldots$
  - $\ln(W_{final}) = \ln(n) + \sum_t [\ln(1 - \varepsilon F_t)] \leq \ln(n) - \varepsilon \sum_t F_t$ (using $\ln(1-x) < -x$)
    $= \ln(n) - \varepsilon M$. ($\sum F_t$ = E[# mistakes])
- If best expert makes m mistakes, then $\ln(W_{final}) > \ln((1-\varepsilon)^m)$.
- Now solve: $\ln(n) - \varepsilon M > m \ln(1-\varepsilon)$.

$$M \leq \frac{-m\ln(1-\varepsilon)+\ln(n)}{\varepsilon} \approx (1+\varepsilon/2)m + \frac{1}{\varepsilon}\log(n)$$

Additive regret

- So, have M ≤ OPT + εOPT + 1/ε log(n).
- Say we know we will play for T time steps. Then can set $\varepsilon = (\log(n)/T)^{1/2}$. Get $M \leq OPT + 2(T * \log(n))^{1/2}$.
- If we don’t know T in advance, can guess and double.
- These are called “additive regret” bounds.

$$M \leq \frac{-m\ln(1-\varepsilon)+\ln(n)}{\varepsilon} \approx (1+\varepsilon/2)m + \frac{1}{\varepsilon}\log(n)$$

Extensions

- What if experts are actions? (rows in a matrix game, choice of deterministic alg to run,…)
- At each time t, each has a loss (cost) in {0,1}.
- Can still run the algorithm
  - Rather than viewing as "pick a prediction with prob proportional to its weight" ,
  - View as "pick an expert with probability proportional to its weight"
- Same analysis applies.

Extensions

- What if losses (costs) in [0,1]?
- Here is a simple way to extend the results.
- Given cost vector c, view c_i as bias of coin. Flip to create boolean vector c', s.t. $E[c'_i] = c_i$. Feed c' to alg A. $\boxed{\text{world}} \xrightarrow{c} \boxed{\$} \xrightarrow{c'} \boxed{A}$
- For any sequence of vectors c', we have: Cost' = cost on c' vectors
  - $E_A[\text{cost}'(A)] \leq \min_i \text{cost}'(i) + [\text{regret term}]$
  - So, $E_\$[E_A[\text{cost}'(A)]] \leq E_\$[\min_i \text{cost}'(i)] + [\text{regret term}]$
- LHS is $E_A[\text{cost}(A)]$.
- RHS $\leq \min_i E_\$[\text{cost}'(i)] + [\text{r.t.}] = \min_i[\text{cost}(i)] + [\text{r.t.}]$

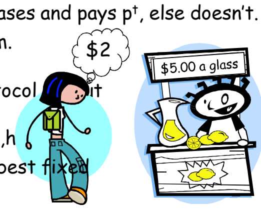
In other words, costs between 0 and 1 just make the problem easier…

What can we use this for?

- Can use to combine multiple algorithms to do nearly as well as best in hindsight.
  - E.g., do nearly as well as best strategy in hindsight in repeated play of matrix game.
- Extension: "sleeping experts". E.g., one for each possible keyword. Try to do nearly as well as best "coalition".
- More extensions: "bandit problem", movement costs.

Online pricing

- Say you are selling lemonade (or a cool new software tool, or bottles of water at the world expo).
- Protocol #1: for t=1,2,…T
  - Seller sets price p^t
  - Buyer arrives with valuation v^t
  - If $v^t \geq p^t$, buyer purchases and pays p^t, else doesn't.
  - v^t revealed to algorithm.
  - repeat
- Protocol #2: same as protocol … it without v^t revealed.
- Assume all valuations in [1,h …
- Goal: do nearly as well as best fixed price in hindsight.



Online pricing

- Say you are selling lemonade (or a cool new software tool, or bottles of water at the world expo).
- Protocol #1: for t=1,2,…T
  - Seller sets price p^t
  - Buyer arrives with valuation v^t
  - If $v^t \geq p^t$, buyer purchases and pays p^t, else doesn't.
  - v^t revealed to algorithm.
- Bad algorithm: "best price in past"
  - What if sequence of buyers = 1, h, 1, …, 1, h, 1, …, 1, h, …
  - Alg makes T/h, OPT makes T.

Factor of h worse!

Online pricing

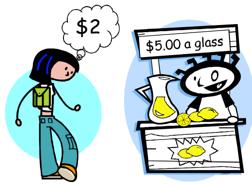
- Say you are selling lemonade (or a cool new software tool, or bottles of water at the world expo).
- Protocol #1: for t=1,2,…T
  - Seller sets price p^t
  - Buyer arrives with valuation v^t
  - If $v^t \geq p^t$, buyer purchases and pays p^t, else doesn't.
  - v^t revealed to algorithm.
- Good algorithm: Randomized Weighted Majority!
  - Define one expert for each price $p = (1+\epsilon)^i \in [1,h]$.
  - Best price of this form gives profit $\geq OPT/(1+\epsilon)$.
  - Run RWM algorithm. Get expected gain at least:

$$(\text{best expert})/(1+\epsilon) - O(\epsilon^{-1}\, h \log n)$$
$$= OPT/(1+\epsilon)^2 - O(\epsilon^{-1}\, h \log(\epsilon^{-1} \log h))$$

[extra factor of h coming from range of gains]

Online pricing

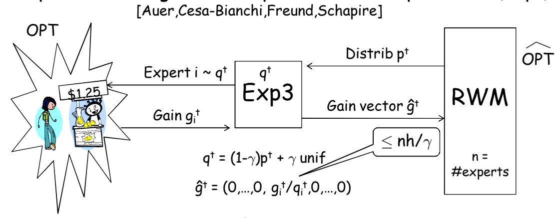
- Say you are selling lemonade (or a cool new software tool, or bottles of water at the world expo).
- What about Protocol #2? [just see accept/reject decision]
  - Now we can't run RWM directly since we don't know how to penalize the experts!
  - Called the "adversarial multiarmed bandit problem"
  - How can we solve that?



Multi-armed bandit problem

Exponential Weights for Exploration and Exploitation ($\exp^3$)

[Auer,Cesa-Bianchi,Freund,Schapire]

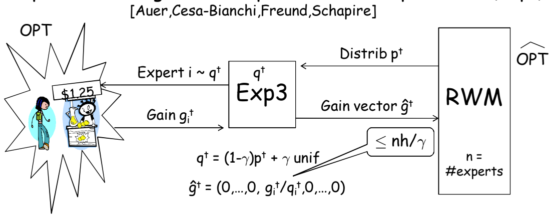


1. RWM believes gain is: $p^t \cdot \hat{g}^t = p_i^t(g_i^t/q_i^t) \equiv g^t_{RWM}$
2. $\sum_t g^t_{RWM} \geq \widehat{OPT}/(1+\epsilon) - O(\epsilon^{-1}\, nh/\gamma \log n)$
3. Actual gain is: $g_i^t = g^t_{RWM}(q_i^t/p_i^t) \geq g^t_{RWM}(1-\gamma)$
4. $E[\widehat{OPT}] \geq OPT$. Because $E[\hat{g}_j^t] = (1-q_j^t)0 + q_j^t(g_j^t/q_j^t) = g_j^t$,
   so $E[\max_j[\sum_t \hat{g}_j^t]] \geq \max_j [\, E[\sum_t \hat{g}_j^t]\,] = OPT$.

Multi-armed bandit problem

Exponential Weights for Exploration and Exploitation ($\exp^3$)

[Auer,Cesa-Bianchi,Freund,Schapire]

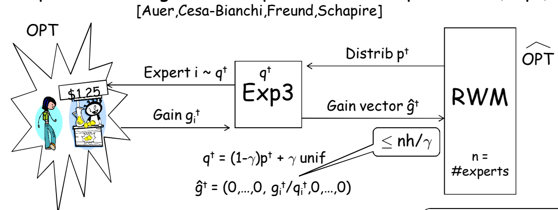


2. $\sum_t g^t_{RWM} \geq \widehat{OPT}/(1+\epsilon) - O(\epsilon^{-1}\, nh/\gamma \log n)$
3. Actual gain is: $g_i^t = g^t_{RWM}(q_i^t/p_i^t) \geq g^t_{RWM}(1-\gamma)$
4. $E[\widehat{OPT}] \geq OPT$. Because $E[\hat{g}_j^t] = (1-q_j^t)0 + q_j^t(g_j^t/q_j^t) = g_j^t$,
   so $E[\max_j[\sum_t \hat{g}_j^t]] \geq \max_j [\, E[\sum_t \hat{g}_j^t]\,] = OPT$.

Multi-armed bandit problem

Exponential Weights for Exploration and Exploitation ($\exp^3$)

[Auer,Cesa-Bianchi,Freund,Schapire]



Conclusion ($\gamma = \epsilon$):

$E[\text{Exp3}] \geq OPT/(1+\epsilon)^3 - O(\epsilon^{-2}\, h\, n \log(n))$

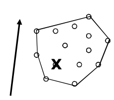
Can even reduce ϵ^{-2} to ϵ^{-1} with more care in analysis.

Almost as good as protocol 1!

Extensions (of expert or bandit problem)

[KV] setting:

- Implicit set S of feasible points in R^m. (E.g., m=#edges, S={indicator vectors 011010010 for possible paths})
- Assume have oracle for offline problem: given vector c, find $x \in S$ to minimize $c \cdot x$. (E.g., shortest path algorithm)
- Use to solve online problem: on day t, must pick $x_t \in S$ before c_t is given.
- $(c_1 \cdot x_1 + \ldots + c_T \cdot x_T)/T \rightarrow \min_{x \in S} x \cdot (c_1 + \ldots + c_T)/T$.



[Z] setting:

- Assume S is convex.
- Allow c(x) to be a convex function over S.
- Assume given any y not in S, can algorithmically find nearest $x \in S$.

Other models in learning

Lots of other models considered as well for different kinds of problems.

- "Active learning": have large unlabeled sample and alg may choose among these.
  - E.g., web pages, image databases.
- "Membership query learning": Algorithm can construct its own examples.
  - E.g., features represent variable-settings in some experiment, label represents outcome.
- "Semi-supervised learning": use of labeled+unlabeled data in passive setting.