Some nifty notations

mw

October 13, 2008

Abstract

1 The predictions probabilities act like log probabilities

$k$ is number of bins.

$s = (s_k, s_{k-1}, \ldots, s_0)$ is vector of $k + 1$ non-negative integers

$s_i$ number of experts on bin $i$, which contains all experts that have made $k - i$ mistakes.

$s, i$ is $s$ with one experts from bin $i$ moved to bin $i + 1$. If $s_i = 0$, then $s, i = s$.

$p(i|s)$ is probability put on bin $i$ when bin vector is $s$.

The following equality holds:

\[ p(i|s) + L(s, i) = L(s) \]

\[ p(i|s) = L(s) - L(s, i) \]

By summing over $i$ we get

\[ 1 = \sum_i p(i|s) = nL(s) - \sum_i L(s, i). \]

Rewriting this give Jake’s formula

\[ L(s) = \frac{\sum_i L(s, i) + 1}{n}. \]

Some new ideas:

\[ p(i|s) + p(j|s, i) = p(j|s) + p(i|s, j). \]

Unravelling ($y_i$ is a sequence of $i$ moves):

\[ p(y_i|s) = \sum_{j=1}^i p(y_j|s, y_{j-1}). \]

What is going on? The $p(y_j|s, y_{j-1})$ are probabilities, but they act like log probabilities. Note that $0 \leq p(y_j|s, y_{j-1}) \leq 1$, but $p(y_i|s)$ may be larger than one.

1
2 Singleton expert theorem

The algorithm chooses a probability vector on the experts and the adversary choose a bit vector \( r \), indicating which expert will move.

\[
L(s) := \min_{p \in S^n} \max_{r \in \{0,1\}^n} p \cdot r + L(s + r)
\]

Here is Jake’s thm. No complete proof yet. Just can show that is suffices to proof it for the diamond.

**Theorem 2.1.** Moving more than one expert at a time is a suboptimal choice of the adversary, i.e. for all \( s \)

\[
\min_{p \in S^n} \max_{r \in \{0,1\}^n, |r|=1} p \cdot r + L(s + r) < \min_{p \in S^n} \max_{r \in \{0,1\}^n, |r|>1} p \cdot r + L(s + r)
\]

**Proof.** Draw a graph where the nodes \( s \) are drawn at level \( L(s) \) from the baseline. Connect \( s \) and \( s' \) by an edge if the differ by a unit vector and label edge by the prediction probabilies put on the corresponding expert.

If the thm is true then it holds for the diamond below, i.e. the case \(|r| = 2\) is suboptimal.

Here the top node corresponds to some state \( s \) and the bottom node to a state where both experts 1 and 2 were moved. The thm now says that moving both experts at once is incurs less loss than moving the first expert and then the second, i.e.

\[
p_1 + p_2 < p_1 + \tilde{p}_2 \iff p_2 < \tilde{p}_2.
\]

**Claim** If the thm holds for the diamond case \((|r| = 2)\) then it holds for \(|r| > 2\).

**Proof.** Assume we move \( k = |r| > 2 \) at once. We will show that the diamond case implies that moving all \( k \) experts at once is suboptimal for the adversary. Let \( s \) be the start node and \( s + r \) the end node. Let \( p_i \) be the prediction probability for expert \( i \) in \( s \). Moving all \( k \) experts at once incurs loss \( p_1 + p_2 + \ldots + p_k \). Now assume we move expert 1, 2, \ldots, \( k \), one at a time. This corresponds to a path from \( s \) to \( s + r \). Assume this path is labeled with probabilities \( p_1, \tilde{p}_2, \ldots, \tilde{p}_k \). We need to show that

\[
p_1 + p_2 + \ldots + p_k < p_1 + \tilde{p}_2 + \ldots + \tilde{p}_k.
\]

For the sake of concreteness assume \( k = 4 \). We will show that

\[
p_1 + p_2 + p_3 + p_4 < p_1 + \tilde{p}_2 + p_3 + p_4 < p_1 + \tilde{p}_2 + \tilde{p}_3 + p_4 < p_1 + \tilde{p}_2 + \tilde{p}_3 + \tilde{p}_4.
\]
The first inequality follows directly from the diamond case. For the second inequality we need to show that \( p_3 < \hat{p}_3 \). To do this we apply the diamond case twice, once to the diamond ending in the top note and once to a diamond on level down. Finally to show that \( p_4 < \hat{p}_4 \) we apply the diamond case 4 times.

Conjecture: If the algorithm predicts with exponential weights (any fixed learning rate) then the optimal choice of the adversary is still \(|r| = 1\).

3 Lower bounds: from \( n = 2 \) to arbitrary \( n \)

Luckily already for \( n = 2 \) we have a lower bound on the regret of the form \( k + c\sqrt{k} \) where \( k \) is the loss of the best. The point here is that this immediately gives a lower bound of \( k + c\sqrt{k \lg_2 n} \).

Joel’s exact formula for \( n = 2 \) is

\[
\frac{k + 1}{4^{k+1}} \left( \frac{2(k+2)}{k+1} \right).
\]

From plotting it we see that this is always at least \( k + \sqrt{\frac{k}{\pi}} \) (see files ma/lb.*).

The upper bound on the loss of tuned WMR is

\[
k + \sqrt{2k \ln n + \ln n}.
\]

Let’s focus on the square root term. The limit \( \frac{1}{\sqrt{\pi}} \) is about 48 \% of \( \sqrt{2 \ln 2} \). Joel also has an exact formula for the case \( n = 3 \). Its limit is about 57 \% of \( \sqrt{2 \ln 3} \).

**Theorem 3.1.** Any lower bound on the loss for \( n = 2 \) of the form \( k + c_1\sqrt{k} + c_2 \) implies a lower bound of \( k + c_1\sqrt{k \lg_2 n} + c_2 \lg_2 n \) for \( n \) experts.

**Proof.** For the sake of sloppiness we ignore ceilings and floors. Initially split the experts into two blocks. All \( \frac{n}{2} \) experts in each block act alike in the first phase. Use the 2 expert \( \frac{k}{\lg_2 n} \) loss case to force regret \( c_1\sqrt{\frac{k}{\lg_2 n}} + c_2 \). One of the blocks has minimum regret. Split that block again in two and repeat the argument. After \( \lg_2 n \) phases one expert has loss \( k \) and the loss of the algorithm is at least

\[
(\lg_2 n)\left( \frac{k}{\lg_2 n} + c_1\sqrt{\frac{k}{\lg_2 n}} + c_2 \right) = c_1\sqrt{k \lg_2 n} + c_2 \lg_2 n.
\]
4 Improving WM

Inspired by the optimal alg. we can modify the WM algorithm as follows under the assumption that there is an expert which makes \( \leq k \) mistakes:

For mistake vector \( s \) predict with distribution

\[
p_i(s) = \begin{cases} 
\frac{\beta^{s_i}}{Z(s)} & \text{if } s_i \leq k \\
0 & \text{otherwise.}
\end{cases}
\]

Here \( \beta \in [0, 1) \) and \( Z(s) \) is a constant that normalizes the \( p_i(s) \) to one.

The original WM update does not set the weights of experts with more than \( k \) mistakes to zero. It only uses exponential weights.

Define a potential

\[
P(s) = -\ln \sum_i \beta^{s_i}.
\]

It is easy to see that setting weights of experts other than the best expert to zero, increases the potential. So the standard WM bound still holds:

\[\text{total loss of alg} \leq P(s).\]

Clearly \( P(s) \leq V(s) \) because the algorithm plays non-optimal and we are upper bounding its loss.

Don’t know how much better this algorithm is than the original WM alg.

5 Multiplicative algs

Consider the following template of an algorithm:

For mistake vector \( s \) predict with distribution

\[
p_i(s) = \frac{f(s_i)}{\sum_j f(s_j)}
\]

Here \( f : \mathbb{N} \to [0, \infty) \) is any non-increasing function. For the original WM, \( f(x) = \beta^x \).

Lemma 5.1. For any template algorithm, the best choice of the adversary is to choose \( r \) as a unit vector.

Proof. The adversary maximizes. The essential part of the proof is to distinguish the following two cases: (A) one trial with \( r = e_i + e_j \) (where \( i \neq j \)), versus (B) one trial with \( r = e_i \) followed by a second trial with \( r = e_j \). We will show that the adversary makes the algorithm incur at least as much loss in case (B).

Let \( s \) be an arbitrary state vector and let \( Z(s) = \sum_q f(s_q) \). In case (A) the loss is

\[
\frac{f(s_i)}{Z(s)} + \frac{f(s_j)}{Z(s)}.
\]

In case (B) the loss is

\[
\frac{f(s_i)}{Z(s)} + \frac{f(s_j)}{Z(s + e_i)}.
\]
Since $Z(s + e_i) \leq Z(s)$, the loss in case (B) is at least as much as the loss in case (A).

\[ \square \]

**Lemma 5.2.** For the templet algorithm, the best sequence of unit vectors that moves 0 to any state is to always pick the expert with smallest loss if possible.

There is a counter example in ma/path.mw

Might still be true for exponential weights. For any $f$ and $s$, what is the best path with the largest total probability along it.

6 Balancing WM

Assume you have an estimate of the value $V(s)$ into a potential $P(s)$. An example is the above potential defined for WMR. We now choose the outgoing probs $p_i(s)$ out of $s$ to $s + e_i$ so that all $p_i(s) + P(s + e_i)$ are all equal to some fixed value $X(s)$. By summing over $i$ and enforcing the constraint that the $p_i(s)$ sum to one, we get

\[
\sum_{i} p_i(s) + \sum_{i} P(s + e_i) = nX(s).
\]

This implies that $X(s) = \frac{1 + \sum_{i} P(s + e_i)}{n}$ and $p_i(s) = X(s) - P(s + e_i)$. Of course there are versions that set all probabilities to non-live successors to zero. In that case we only average over live successors.

Anyway! What properties of the potential $P(s)$ are needed so that the resulting $p_i(s)$ lie in $[0,1]$?

1. It seems that we need that $|P(s + e_i) - P(s + e_j)| \leq 1$, for any $i, j$. Is this condition necessary and sufficient?

2. Does this property hold for

\[
P(s) = -\ln \frac{\sum_{i=1}^{n} \beta^{s_i}}{1 - \beta}
\]

and/or for

\[
P(s) = -\ln \frac{\sum_{i \in \lambda(s)} \beta^{s_i}}{1 - \beta}
\]

3. This scheme of balancing one step ahead should improve WM. Can we balance $q$ steps ahead, for $q > 1$. So now we estimate the value of the game for all $q$ away successors of $s$ and then estimate the local probs by balancing. Intuitively this method morphs WMR into the optimal alg
7 Multinomial formula for \( p_i(s) \)

Let \( s \) be a state vector with \( 0 \leq s_i \leq k + 1 \). We showed that \( p_i(s) = V(s) - V(s+e_i) \). Following Joel’s thinking, I now give a concise formula for this gap. It is simply the probability of all paths thru the final state \( i \) to the sink state. Note that there are no transitions from the sink state and the last move on a path thru final state \( i \) is an \( e_i \) move. All moves are chosen uniformly among the \( n \) experts and the probability of a path with \( q \) moves is \( (\frac{1}{n})^q \). We are computing the probability of all paths to the sink where the last move is an \( e_i \) move. In the formula, \( m_j \geq 0 \) is the number of \( e_j \) moves in a path. The formula is a sum, where each summand accounts for all paths with the same move count \((m_1, \ldots, m_n)\). The multinomial is the number of such paths. The number of moves in these paths is \( 1 + \sum_j m_j \), where the 1 accounts for the last move (the \((k+1)st\) \( e_i \) move. Which move count vector sends a path to the \( i \)th final state? Well, we must have \( m_j \geq k + 1 - s_j \) for all \( j \neq i \), and \( m_i = k - s_i \). Again, the last move is fixed to be an \( e_i \) move from the \( i \)th final state to the sink.

\[
p_i(s) = \sum_{m_j \geq k+1-s_j \text{ for } j \neq i \text{ and } m_i = k-s_i} \left( \sum_{j=1}^{n} m_j \right) \left( \sum_{j=1}^{n} m_j \right) \left( \frac{1}{n} \right)^{\sum_{j=1}^{n} m_j}
\]

Current conjectured formula for the value of the game (no proof):

\[
V(s) = \text{expected path length for all paths from } s \text{ to the sink state.}
\]

Something else I tried:

\[
V(s) = \sum_{m_j \geq k+1-s_j} \left( \sum_{j=1}^{n} m_j \right) \left( \sum_{j=1}^{n} m_j \right) \left( \frac{1}{n} \right)^{\sum_{j=1}^{n} m_j}
\]

Check:

\[
p_i(s) = V(s) - V(s+e_i)
\]

\[
= \sum_{m_j \geq k+1-s_j} \left( \sum_{j=1}^{n} m_j \right) \left( \sum_{j=1}^{n} m_j \right) \left( \frac{1}{n} \right)^{\sum_{j=1}^{n} m_j}
- \sum_{m_j \geq k+1-s_j \text{ for } j \neq i \text{ and } m_i \geq k+1-s_i-1} \left( \sum_{j=1}^{n} m_j \right) \left( \frac{1}{n} \right)^{\sum_{j=1}^{n} m_j}

= \sum_{m_j \geq k+1-s_j \text{ for } j \neq i \text{ and } m_i = k-s_i} \left( \sum_{j=1}^{n} m_j \right) \left( \frac{1}{n} \right)^{\sum_{j=1}^{n} m_j}
\]

The above is not quite \( p_i(s) \). Previous def of \( p_i(s) \): probability of all paths starting from \( s \) thru the final state \( i \) to the sink state.

Alternate def: probability of all paths starting from \( s \) with an \( e_i \) move and ending at \((s_1 > k, \ldots, s_{i-1} > k, s_i = k, s_{i+1} > k, \ldots, s_n > k)\).

Note that there are no transitions from the sink state and the last move on a path thru final state \( i \) is an \( e_i \) move.
8 Combinatorial Sums

For any state $s \in S$,

$$
\hat{p}_i(s) = \sum_{r: s + r = o_i} \binom{|r|}{r_1, r_2, \ldots, r_n} \left( \frac{1}{n} \right)^{|r|+1}.
$$

Since $\hat{p}_i(s)$ is a probability distribution, summing over $i$ is one:

$$
\sum_{i=1}^{n} \sum_{r: s + r = o_i} \binom{|r|}{r_1, r_2, \ldots, r_n} \left( \frac{1}{n} \right)^{|r|+1} = 1.
$$

Since $V(s)$ can be written as an expected path length, we can obtain a similar expression as a sum of multinomials for $V(s)$:

$$
V(s) = \frac{1}{n} \sum_{i=1}^{n} \sum_{r: s + r = o_i} (|r| + 1) \binom{|r|}{r_1, r_2, \ldots, r_n} \left( \frac{1}{n} \right)^{|r|+1}
$$

I think we forgot the $\frac{1}{n}$ in the paper.

8.1 Formulas for the symmetric case

We are mostly interesting formulas for $V(0)$, i.e. the value of the game when all experts have made zero mistakes. In this case all all summands fo the outer sum of the formula for $V(0)$ are the same and therefore:

$$
V(s) = \sum_{r: s + r = o} (|r| + 1) \binom{|r|}{r_1, r_2, \ldots, r_n} \left( \frac{1}{n} \right)^{|r|+1}.
$$

$$
= \sum_{r_1 \geq k+1, \ldots, r_{n-1} \geq k+1} (k+1 + \sum_{i=1}^{n-1} r_i) \binom{k + \sum_{i=1}^{n-1} r_i}{r_1, r_2, \ldots, r_{n-1}, k} \left( \frac{1}{n} \right)^{k+1+\sum_{i=1}^{n-1}}.
$$

9 Short paper

- Discussion of termination issues as in current paper
- Recursive of $V(s)$ as in current paper
- Let the adversary pick its loss vector probabilistically and rewrite the recurrence accordingly and point out that the value is not increased by giving the adversary more choices.
- Fix the strategy of the adversary to be uniform on $e_i$, where $i \in \lambda(s)$. Call the value of the game when the adversary uses this strategy and the learner plays optimally $V_A(s)$. Clearly, $V(s) \geq V_A(s)$, since the adversary maximizes and the
chosen strategy might not be optimal. Independent of the choice of the learner we end up with the following recurrence:

\[
V_A(s) = \frac{1 + \sum_{i \in \lambda(s)} V_A(s + e_i)}{|\lambda(s)|},
\]

for \( s \neq d \) and \( V_A(d) = 0 \). We now define a different quantity that has the same recurrence. Let \( \tau(s) \) be the expected number of steps in a random walk on the state lattice \( S \) that starts at \( s \) and continues until the dead state \( d \) is reached. Note that \( \tau(s) \) has the recurrence

\[
\tau(s) = 1 + \sum_{i=1}^{n} \frac{\tau(s + e_i)}{n},
\]

which can be rewritten as the expected “waiting time” until the first live event is drawn plus and average over the successors when a live event is drawn:

\[
\tau(s) = \sum_{q=1}^{\infty} \left( \frac{1 - |\lambda(s)|}{n} \right)^{q-1} \cdot \left( \frac{|\lambda(s)|}{n} \right) + \frac{\sum_{i \in \lambda(s)} \tau(s + e_i)}{|\lambda(s)|}.
\]

Clearly \( \frac{\tau(s)}{n} \) and \( V_A(s) \) have the same recurrence and value 0 at \( s = d \). We conclude that \( V_A(s) = \frac{\tau(s)}{n} \).

- We now fix a strategy for the learner to \( \hat{p}_i(s) := \frac{\tau(s) - \tau(s + e_i)}{n} \), for \( s \neq d \). For convenience we define \( \hat{p}_i(s) \) for \( s = d \) the same way. Note that \( \hat{p}_i(d) = 0 \) for all \( i \). For \( s \neq d \), \( \hat{p}_i(s) \) has the following recurrence:

\[
\hat{p}_i(s) = \frac{\tau(s) - \tau(s + e_i)}{n} = \frac{1}{n^2} \sum_{j} \tau(s + e_j) - \frac{1}{n^2} \sum_{j} \tau(s + e_j + e_i)
\]

\[
= \frac{1}{n} \sum_{j=1}^{n} \hat{p}_i(s + e_j) - \frac{1}{n} \sum_{j=1}^{n} \hat{p}_i(s + e_j + e_i)
\]

We claim that \( \hat{p}_i(s) \) can be interpreted as the probability that a random walk starting at \( s \) reaches the dead state via \( o_i \). To prove that this alternate interpretation is correct, first observe that once you are in the dead state then the probability for reaching any of the \( o_i \) is zero and this is consistent with \( \hat{p}_i(d) = 0 \). Also for \( s \neq d \) then a successor \( s + e_i \) of \( s \) is chosen by picking \( e_i \) uniformly with
probability $\frac{1}{n}$. Thus the probability of reaching the dead state from $s$ via $o_i$ is the average of the probability of reaching the dead state from any of the successors $s + e_i$ via $o_i$. The latter is consistent with the recurrence  
\[ \hat{p}_i(s) = \frac{1}{n} \sum_{j=1}^n \hat{p}_i(s + e_j). \]

If $s \neq d$ then any path starting at $s$ enters the dead state via one of the $o_i$. Thus in this case the vector $\hat{p}_n(s)$ is always a probability distribution and constitutes a legal choice for the learner.

- Denote the value achieved when the adversary plays optimally against this strategy of the learner as $V_L(s)$. Clearly $V_L(s) \geq V(s)$ since the learner minimizes but its strategy might not be optimal. Thus to show that $V(s) = \frac{r(s)}{n}$ it suffices to show that $V_L(s) = \frac{r(s)}{n}$.

- If the adversary plays unit vectors then the cost $\hat{p}_i(s)$ summed along any path from $s$ to $d$ telescopes and is $\frac{r(s)}{n} - \frac{r(d)}{n} = \frac{r(s)}{n}$. We now show that choosing non-unit losses is non-optimal for the adversary. We begin with the diamond lemma. Note that the proof is based on the alternate interpretation of the $\hat{p}_i(s)$.

Lemma 9.1. For any state $s$ and distinct events $i, j \in \lambda(s)$, we have  
\[ \hat{p}_i(s) < \hat{p}_i(s + e_j). \]

This fact is intuitive: if losses are randomly assigned then the probability that the $i$th event will survive last strictly increases when another event suffers a loss. We prove this precisely below.

Proof. To show that $\hat{p}_i(s) \leq \hat{p}_i(s + e_j)$ is straightforward. Any sequence $S_0, S_1, S_2, \ldots$ that brings $s$ to the one-live state $o_i$ also brings $s + e_j$ to $o_i$. Indeed, if $s + S_t = o_i$ for some $t$ then certainly $(s + e_j) + S_t = o_i$ as well.

To show that this inequality is strict, we need only find one random sequence for which $s + e_j$ is brought to $o_i$ but not $s$. Take any sequence $S_0, S_1, \ldots$ such that $s + S_t = d - e_i - e_j$ (where the only events remaining are $i$ and $j$) and where $S_{t+1} = S_t + e_i$. Then $(s + e_j) + S_t = o_i$ but $s + S_{t+1} = s + (S_t + e_i) = o_j$.  

Adapt the below to how that non-unit losses are non-optimal: Assume indeed that $|\ell^*| > 1$, i.e. it admits a decomposition $\ell^* = e_i + \tilde{\ell}$ for some $i$ and bit vector $\tilde{\ell} \neq 0$ with $\tilde{\ell}_i = 0$. Applying Lemma 9.1 repeatedly, we have that $\hat{p}_i(s) < \hat{p}_i(s + \ell)$ and therefore  
\[ \hat{p}(s) \cdot \ell^* + \hat{V}(s + \ell^*) = \hat{p}_i(s) + \hat{p}(s) \cdot \tilde{\ell} + \hat{V}(s + \ell^*) < \hat{p}_i(s + \ell) + \hat{p}(s) \cdot \tilde{\ell} + \hat{V}(s + \ell^*) \]

(Lem. 9.1)  
\[ = \hat{V}(s + \ell) - \hat{V}(s + \ell^*) + \hat{p}(s) \cdot \tilde{\ell} + \hat{V}(s + \ell^*) \]

(Cor. 7.7)  
\[ = \hat{p}(s) \cdot \tilde{\ell} + \hat{V}(s + \ell). \]
But the statement $\hat{p}(s) \cdot \ell^* + \hat{V}(s + \ell^*) < \hat{p}(s) \cdot \ell + \hat{V}(s + \ell)$ implies $\ell^*$ is a non-optimal choice for the Casino and this contradicts our assumption that $\ell^*$ was optimum.

- From the above it follows that adversary should never play a non-unit vectors. Also if alg. does not play $\hat{p}_i(s)$ then not all paths are balanced and adversary can achieve gain.

10 Sets of size $\leq m$

$n$ - number of events

$1 \leq m \leq n$ - size of sets

$m$-set $Q$ - a set of events of size $m$

$S = \{0, \ldots, n\}$ - state space

$s \in S$ - total losses of the $n$ events

$\ell \in \{0, 1\}^n$ - loss vector

$k$ - upper bound the total loss of the best set

An $m$-set $Q$ is live, if its total loss $\sum_{i \in Q} s_i$ is at most $k$. An event $i$ is live, if it occurs in some live $m$-set and we let $\lambda(s)$ denote the set of live events at state $s$. A state vector $s$ is live if it contains at least one live event, i.e. $|\lambda(s)| \geq 1$. Similarly a state vector $s$ is dead if $|\lambda(s)| = 0$.

Clearly $s$ is live if its $m$-smallest components sum to at most $k$ and $s$ is dead if its $m$-smallest components sum to at least $k + 1$. Also an event $i \in \lambda(s)$, if in some sorted order of $s$, $s_i$ either appears in position $m$ or below, or $s_i$ appears in position $m + 1$ or higher and $s_i$ plus the total of the $m - 1$ smallest is at most $k$.

In the def's of $V(s)$, we again restrict the Gambler to only put positive weight on live events and the Casino to only give non-zero losses to live events. We also assume that the loss vector $\ell$ is not equal zero. The def of $V$ is the same as before except that the set of dead states has changed.

Markov process: At all live states $s$, a random unit vector $e_i$ is chosen uniformly at random and we advance to state $s + e_i$. The process stops as soon as you reach a dead state. A walk/path always starts at a initial state $s$ and stops as soon as a dead state is reached. So the last move on any path always is a move from a live state to a dead state. The probability of a random walk/path is again $\frac{1}{n}$ sup its length (number of moves).

The proof follows the same outline as the short proof for the genereric setting.

- Again we show that $V(s) = \frac{\tau(s)}{n}$. The $\geq$ direction is achieved by fixing the adversary’s strategy to a uniform choice over the live units.

- The $\geq$ direction is achieved by fixing the learner’s strategy to

$$\hat{p}_i(s) := \frac{\tau(s) - \tau(s + e_i)}{n}, \text{ if } s \text{ live.}$$

We again show that

$$\hat{p}_i(s) = Pr(\text{all paths with last transition } e_i).$$
• Diamond lemma holds again because of the absorbing property. Therefore the adversary must choose uniformly from the live units.

11 Permututations

Very similar to the set case. The unit vectors $e_i$ are square matrices of zeros with a one in a single position. That is these matrices have the form $e_i e'_j$.

12 The general case

Partition of $\{0, 1, 2, \ldots \}^n$ into two disjoint sets $\mathcal{L}$ and $\mathcal{D}$ of live and dead states.

Same uniform markov process from all states $s$ in $\mathcal{L}$, i.e. $s$ has $n$ edges emanating from it. All dead state are sink states.

Minimal properties of the partition:

• For all $s \in \mathcal{L}$, the shortest path (sequence of units) from $s$ to any state in $\mathcal{D}$ is finite.

• Absorbing property of $\mathcal{D}$: For all $s \in \mathcal{D}$ and any $e_i$, $s + e_i \in \mathcal{D}$ as well.

• $\lambda(s)$ consists of all events that s.t. $s + e_i$ has a different set of paths that send this state to $\mathcal{D}$ than $s$ does.

For any $s \in \mathcal{L}$,

$$\hat{p}_i(s) = Pr(\text{last move of path starting at } s \text{ is } e_i).$$

• This is all related to Section 1. We are building a probability theory based on addition and subtraction instead of multiplication and division. Our stuff is an alternate to Bayes! Pls read Section 1 again.

• The only reason we need live and dead events is because we need the recursion to advance. Without the live/dead event thing the proof would be even shorter.