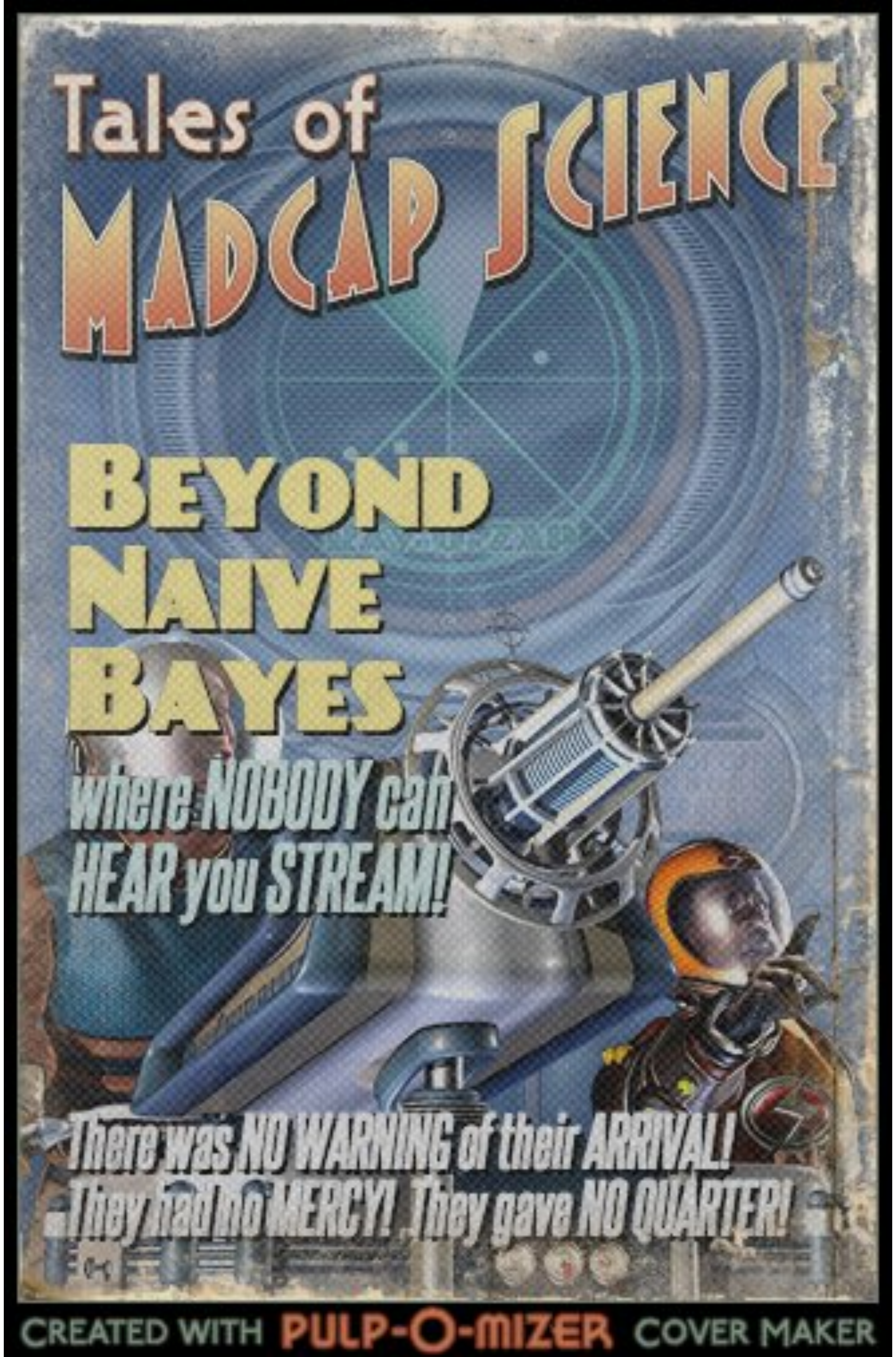
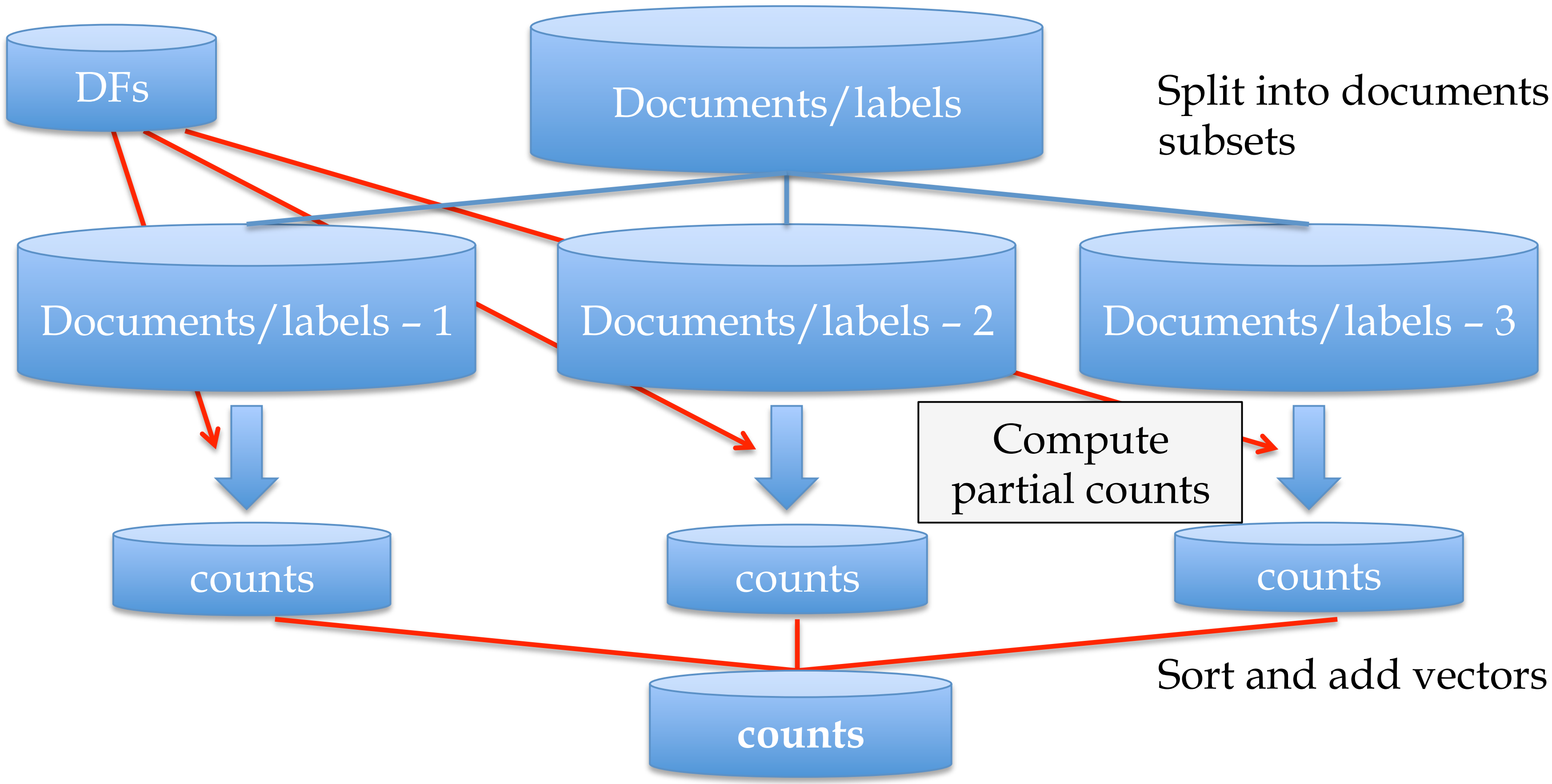

Tales of
MADCAP SCIENCE
BEYOND
NAIVE
BAYES
where NOBODY can
HEAR you STREAM!
There was NO WARNING of their ARRIVAL!
They had no MERCY! They gave NO QUARTER!
CREATED WITH PULP-O-MIZER COVER MAKER

RECAP

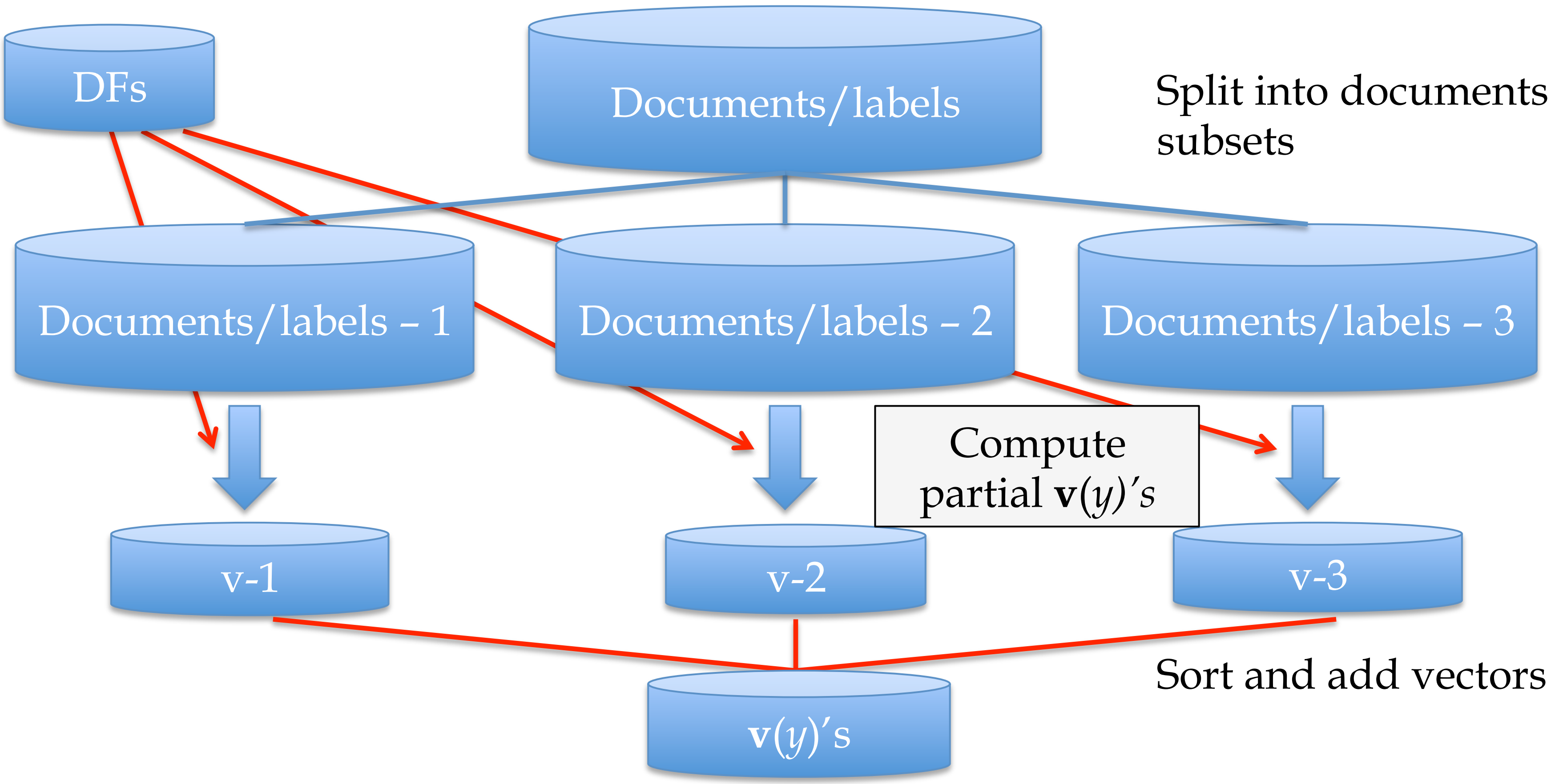
Parallel NB Training



Key Points:

- The "full" event counts are a sum of the "local" counts
- Easy to combine *independently computed* local counts

Parallel Rocchio

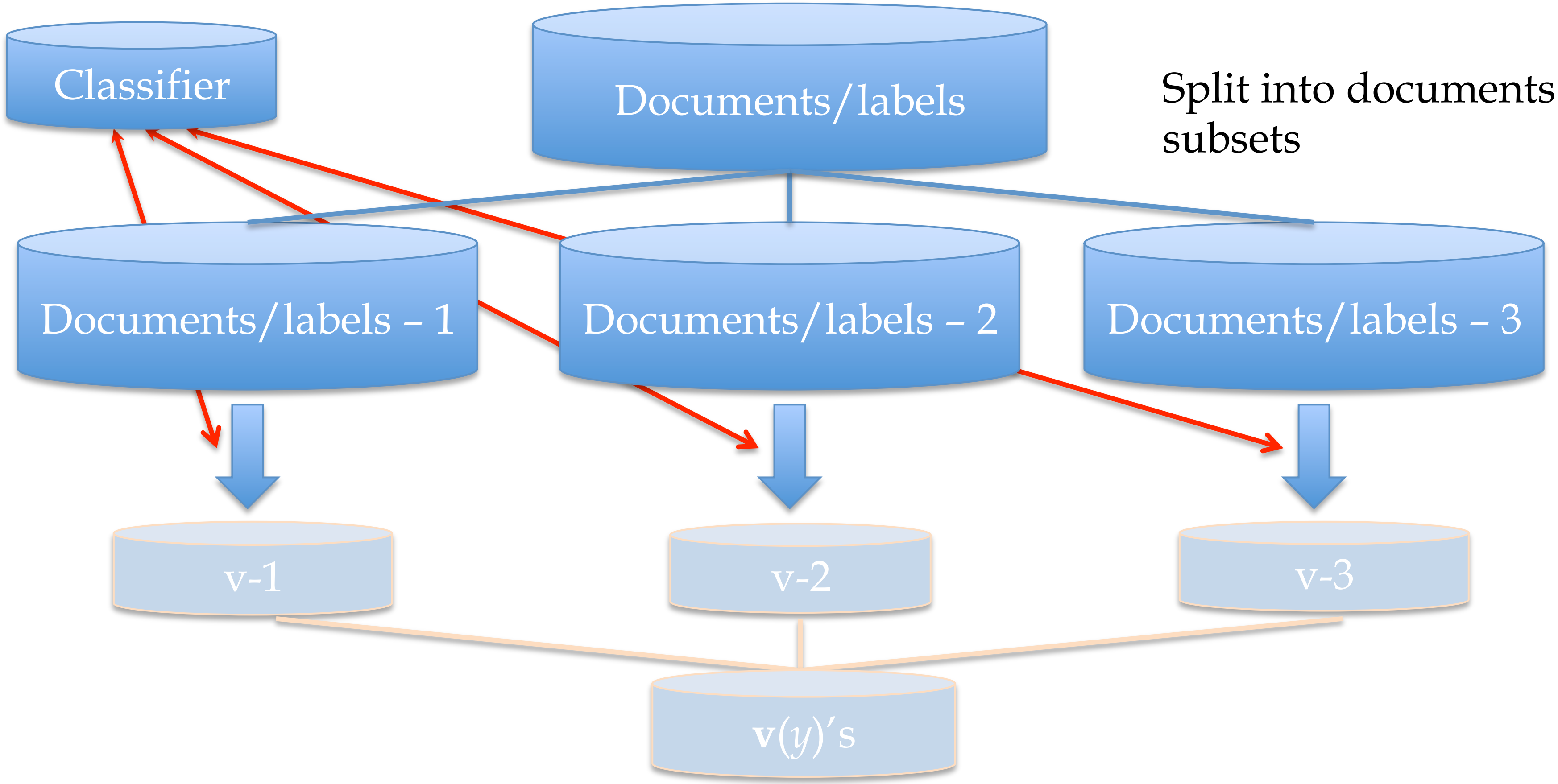


Key Points:

- We need shared *read* access to DF's, but not *write* access.
- The "full classifier" is a *weighted average* of the "local" classifiers – still easy!

Parallel Perceptron Learning?

Like DFs or event counts, size is O(|V|)

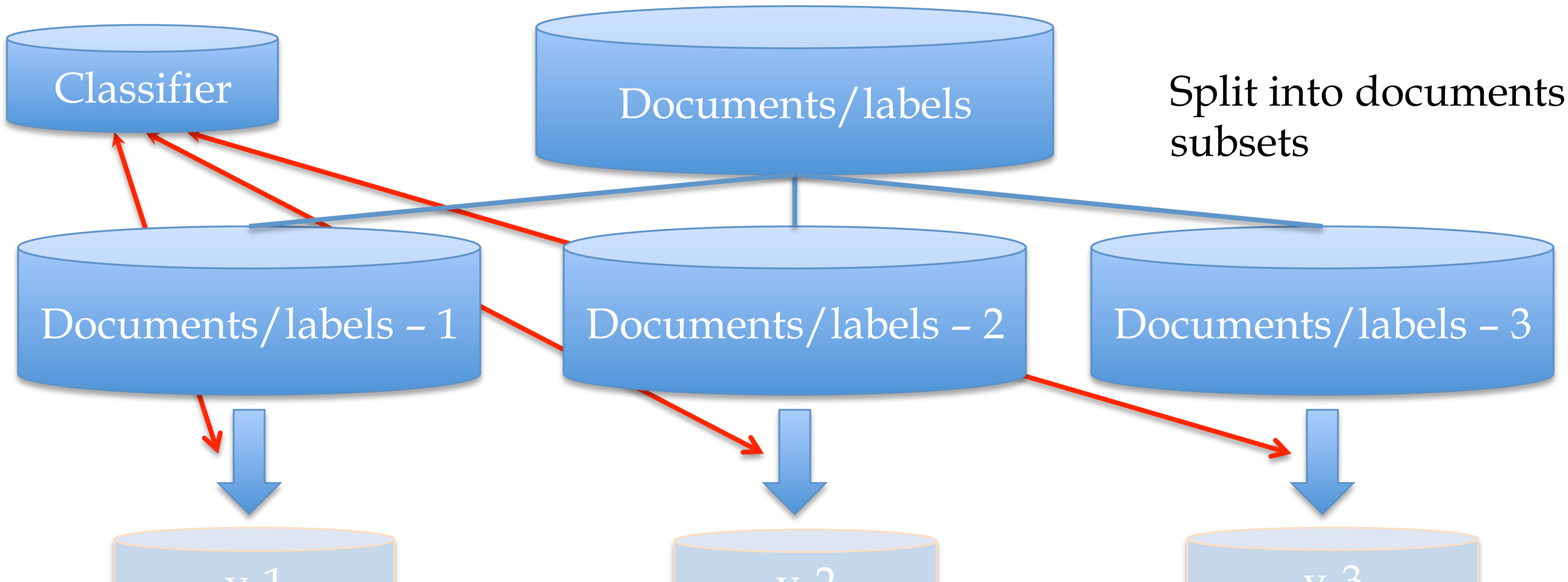


Key Points:

- ~~The "full classifier" is a *weighted average* of the "local" classifiers~~
- Obvious solution requires *read/write* access to a *shared* classifier.

Parallel Streaming Learning

Like DFs or event counts, size is O(|V|)



Key Point: We need shared *write access* to the classifier – not just *read access*. So we only need to not copy the information but **synchronize** it. **Question:** How much extra communication is there?

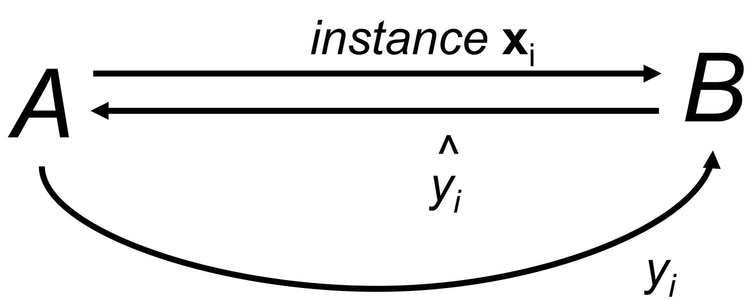
Answer: Depends on how the learner behaves…

…how many weights get updated with each example … (in Naïve Bayes and Rocchio, only weights for features with non-zero weight in **x** are updated when scanning **x**)

…how often it needs to update weight … (how many mistakes it makes)

The perceptron game

$\mathbf{x}$ is a vector
y is -1 or +1



Compute: $\hat{y}_i = \text{sign}(\mathbf{v}_k \cdot \mathbf{x}_i)$

If mistake: $\mathbf{v}_{k+1} = \mathbf{v}_k + y_i \mathbf{x}_i$

mistake bound: $k \leq \left(\frac{R}{\gamma}\right)^2$

Margin γ**.** A must provide examples that can be separated with some vector $\mathbf{u}$ with margin $\gamma > 0$, ie

$$\exists \mathbf{u} : \forall (\mathbf{x}_i, y_i) \text{ given by } A,\ (\mathbf{u} \cdot \mathbf{x}) y_i > \gamma$$

and furthermore, $\|\mathbf{u}\| = 1$.

Radius R**.** A must provide examples "near the origin", ie

$$\forall \mathbf{x}_i \text{ given by } A,\ \|\mathbf{x}_i\|^2 < R^2$$

STRUCTURED PERCEPTRONS…

Distributed Training Strategies for the Structured Perceptron

Ryan McDonald **Keith Hall** **Gideon Mann**
Google, Inc., New York / Zurich
{ryanmcd|kbhall|gmann}@google.com

NAACL 2010

Parallel Structured Perceptrons

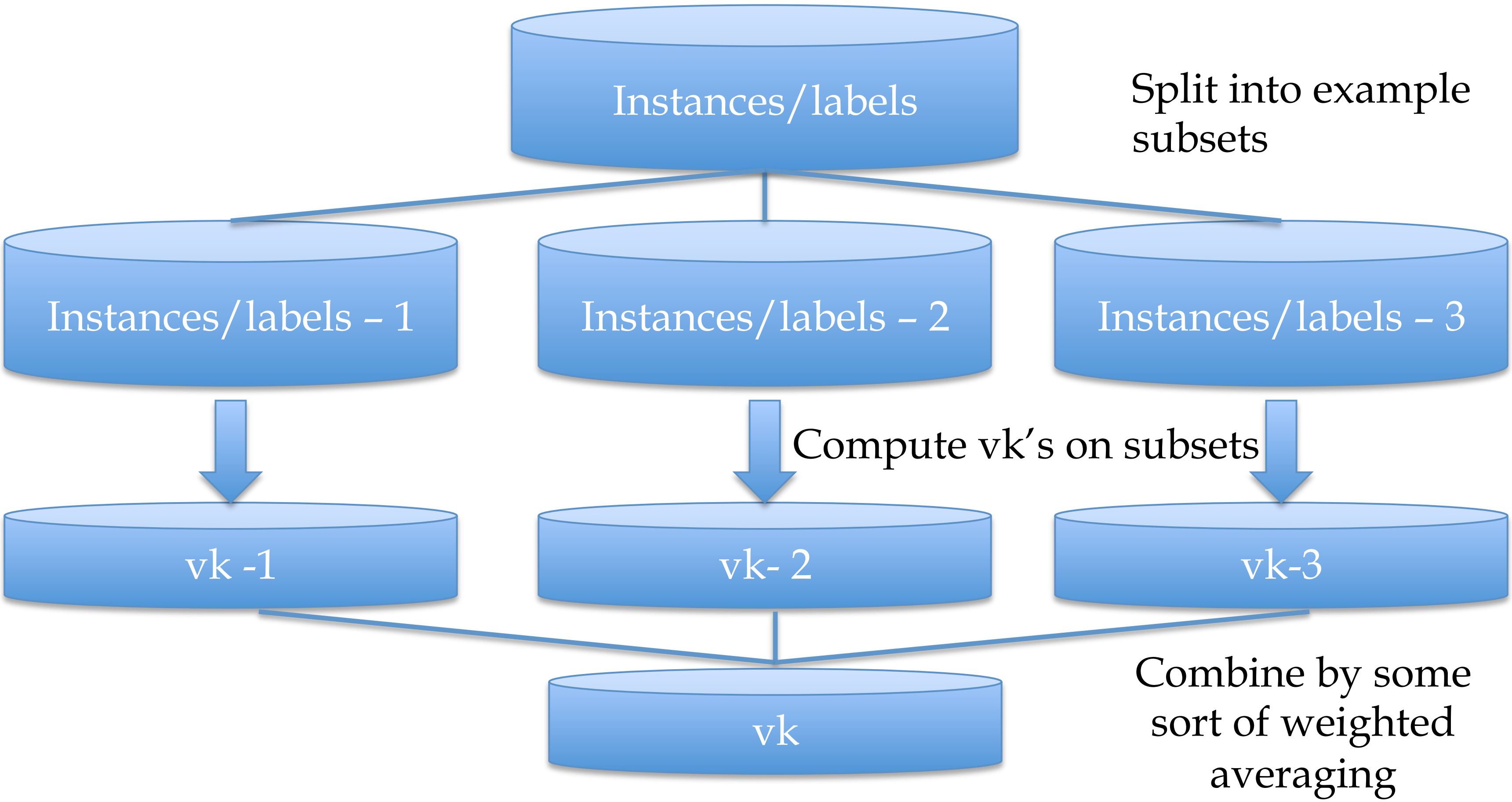
- Simplest idea:
  - Split data into S "shards"
  - Train a perceptron on each shard independently
    - weight vectors are $\mathbf{w}^{(1)}$, $\mathbf{w}^{(2)}$, …
  - Produce some weighted average of the $\mathbf{w}^{(i)}$'s as the final result

PerceptronParamMix($\mathcal{T} = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{|\mathcal{T}|}$)

1. Shard $\mathcal{T}$ into S pieces $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_S\}$
2. $\mathbf{w}^{(i)} = \text{Perceptron}(\mathcal{T}_i)$ †
3. $\mathbf{w} = \sum_i \mu_i \mathbf{w}^{(i)}$ ‡
4. return $\mathbf{w}$

Figure 2: Distributed perceptron using a parameter mixing strategy. † Each $\mathbf{w}^{(i)}$ is computed in parallel. ‡ $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_S\}, \forall \mu_i \in \boldsymbol{\mu} : \mu_i \geq 0$ and $\sum_i \mu_i = 1$.

Parallelizing perceptrons



Parallel Perceptrons

- Simplest idea:
  - Split data into S "shards"
  - Train a perceptron on each shard independently
    - weight vectors are $\mathbf{w}^{(1)}$, $\mathbf{w}^{(2)}$, …
  - Produce some weighted average of the $\mathbf{w}^{(i)}$'s as the final result

- Theorem: this doesn't always work.
- Proof: by constructing an example where you can converge on every shard, and still have the averaged vector *not* separate the full training set – no matter *how* you average the components.

PerceptronParamMix($\mathcal{T} = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{|\mathcal{T}|}$)
1. Shard $\mathcal{T}$ into S pieces $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_S\}$
2. $\mathbf{w}^{(i)} = \text{Perceptron}(\mathcal{T}_i)$ †
3. $\mathbf{w} = \sum_i \mu_i \mathbf{w}^{(i)}$ ‡
4. return $\mathbf{w}$

Figure 2: Distributed perceptron using a parameter mixing strategy. † Each $\mathbf{w}^{(i)}$ is computed in parallel. ‡ $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_S\}$, $\forall \mu_i \in \boldsymbol{\mu} : \mu_i \geq 0$ and $\sum_i \mu_i = 1$.

Parallel Perceptrons – take 2

PerceptronIterParamMix($\mathcal{T} = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{|\mathcal{T}|}$)

Shard $\mathcal{T}$ into S pieces $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_S\}$

$\mathbf{w} = \mathbf{0}$

for n : 1..N

  $\mathbf{w}^{(i,n)} = \text{OneEpochPerceptron}(\mathcal{T}_i, \mathbf{w})$ †

  $\mathbf{w} = \sum_i \mu_{i,n} \mathbf{w}^{(i,n)}$ ‡

return $\mathbf{w}$

OneEpochPerceptron($\mathcal{T}, \mathbf{w}^*$)

$\mathbf{w}^{(0)} = \mathbf{w}^*$; $k = 0$

for t : 1..T

  Let $\mathbf{y}' = \arg\max_{\mathbf{y}'} \mathbf{w}^{(k)} \cdot \mathbf{f}(\mathbf{x}_t, \mathbf{y}')$

  if $\mathbf{y}' \neq \mathbf{y}_t$

    $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{f}(\mathbf{x}_t, \mathbf{y}')$

    $k = k + 1$

return $\mathbf{w}^{(k)}$

Figure 3: Distributed perceptron using an iterative parameter mixing strategy. † Each $\mathbf{w}^{(i,n)}$ is computed in parallel. ‡ $\boldsymbol{\mu}_n = \{\mu_{1,n}, \ldots, \mu_{S,n}\}$, $\forall \mu_{i,n} \in \boldsymbol{\mu}_n$: $\mu_{i,n} \geq 0$ and $\forall n$: $\sum_i \mu_{i,n} = 1$.

Idea: do the simplest possible thing iteratively.

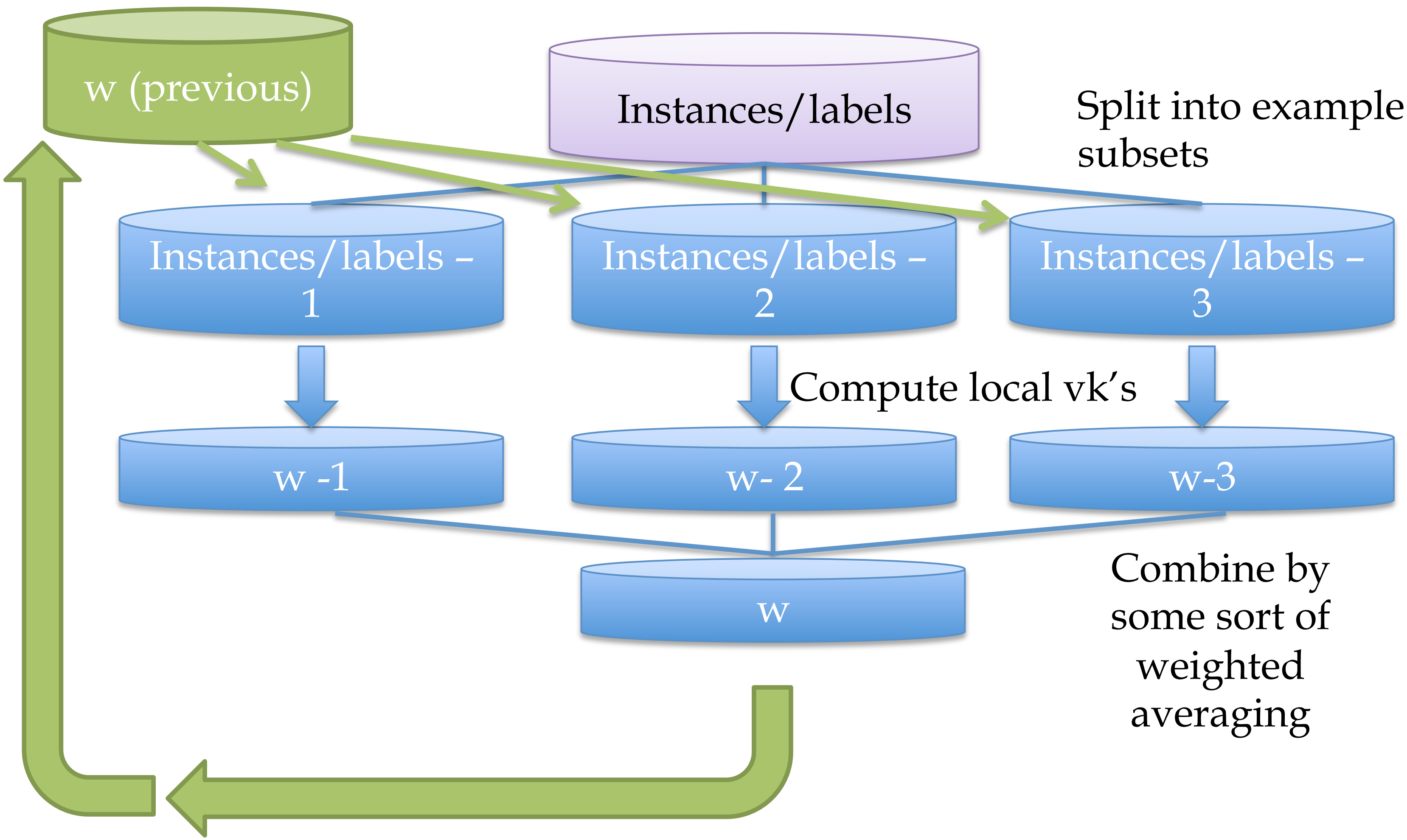
- Split the data into shards
- Let **w = 0**
- For n=1,…
  - Train a perceptron on each shard with one pass *starting with* **w**
  - Average the weight vectors (somehow) and let **w** be that average

All-Reduce

Extra communication cost:

- redistributing the weight vectors
- done less frequently than if fully synchronized, more frequently than if fully parallelized

Parallelizing perceptrons – take 2



A theorem

Theorem 3. *Assume a training set $\mathcal{T}$ is separable by margin γ. Let $k_{i,n}$ be the number of mistakes that occurred on shard i during the nth epoch of training. For any N, when training the perceptron with iterative parameter mixing (Figure 3),*

$$\sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} \leq \frac{R^2}{\gamma^2}$$

Corollary: if we weight the vectors uniformly, then the number of mistakes is still bounded.

I.e., this is "enough communication" to guarantee convergence.

What we know and don't know

Theorem 3. *Assume a training set $\mathcal{T}$ is separable by margin γ. Let $k_{i,n}$ be the number of mistakes that occurred on shard i during the nth epoch of training. For any N, when training the perceptron with iterative parameter mixing (Figure 3),*

uniform mixing… μ =1/S

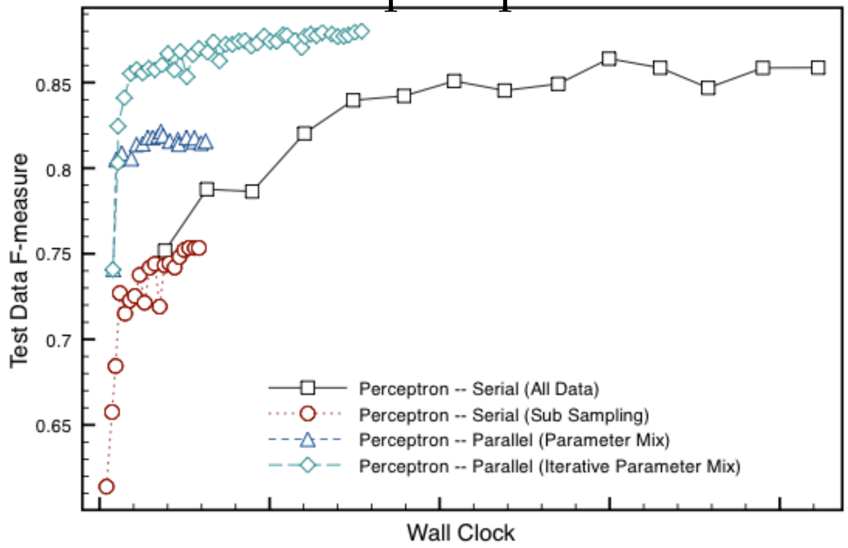
$$\sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} \leq \frac{R^2}{\gamma^2} \qquad \Longrightarrow \qquad \sum_{n=1}^{N} \sum_{i=1}^{S} k_{i,n} \leq S \times \frac{R^2}{\gamma^2}$$

could we lose our speedup-from-parallelizing to slower convergence?

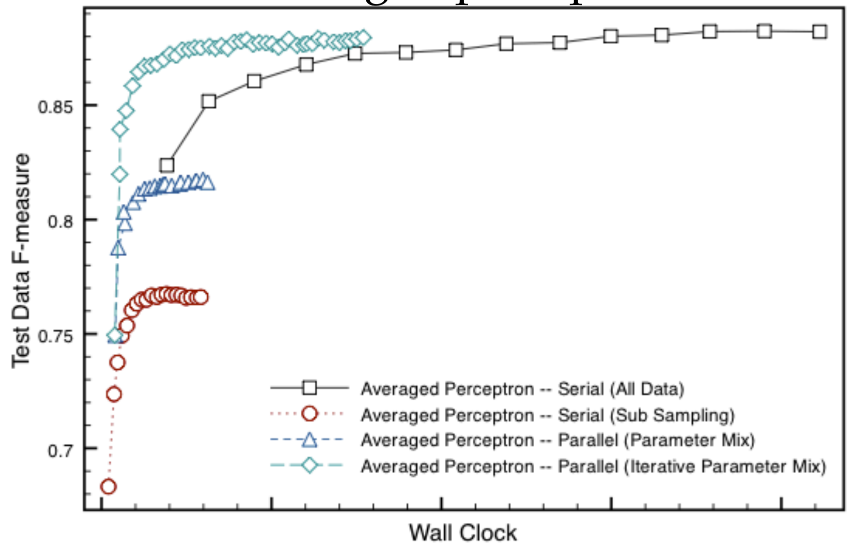
speedup by factor of S is cancelled by slower convergence by factor of S

Results on NER

perceptron



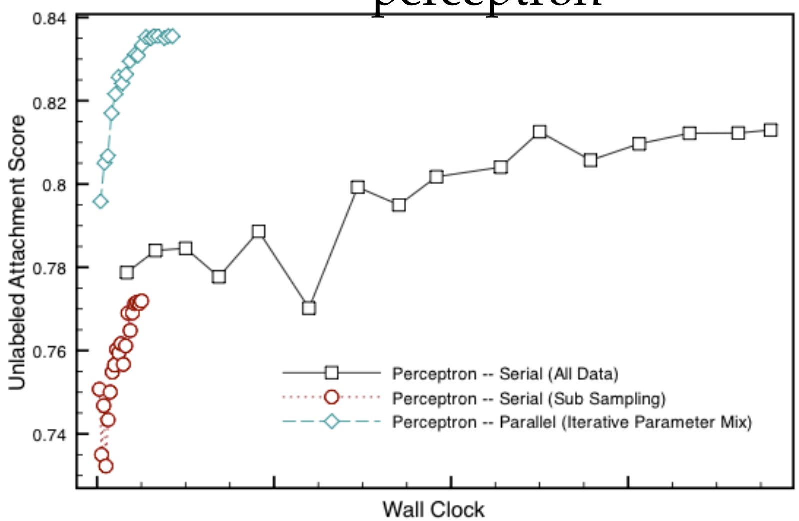
Averaged perceptron



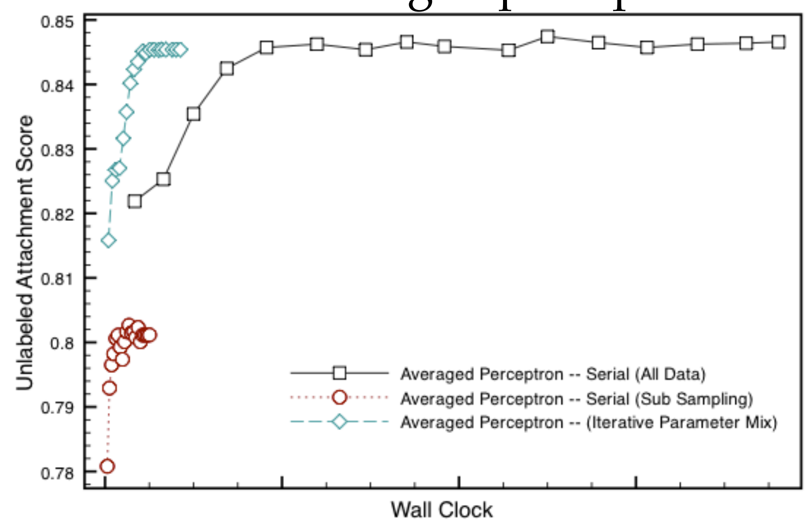
	Reg. Perceptron F-measure	Avg. Perceptron F-measure
Serial (All Data)	85.8	88.2
Serial (Sub Sampling)	75.3	76.6
Parallel (Parameter Mix)	81.5	81.6
Parallel (Iterative Parameter Mix)	87.9	88.1

Results on parsing

perceptron



Averaged perceptron



	Reg. Perceptron Unlabeled Attachment Score	Avg. Perceptron Unlabeled Attachment Score
Serial (All Data)	81.3	84.7
Serial (Sub Sampling)	77.2	80.1
Parallel (Iterative Parameter Mix)	83.5	84.5

The theorem…

Perceptron($\mathcal{T} = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{|\mathcal{T}|}$)

1. $\mathbf{w}^{(0)} = 0;\ k = 0$
2. for $n : 1..N$
3. for $t : 1..T$
4. Let $\mathbf{y}' = \arg\max_{\mathbf{y}'} \mathbf{w}^{(k)} \cdot \mathbf{f}(\mathbf{x}_t, \mathbf{y}')$
5. if $\mathbf{y}' \neq \mathbf{y}_t$
6. $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{f}(\mathbf{x}_t, \mathbf{y}')$
7. $k = k + 1$
8. return $\mathbf{w}^{(k)}$

Theorem 3. *Assume a training set $\mathcal{T}$ is separable by margin γ. Let $k_{i,n}$ be the number of mistakes that occurred on shard i during the nth epoch of training. For any N, when training the perceptron with iterative parameter mixing (Figure 3),*

$$\sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} \leq \frac{R^2}{\gamma^2}$$

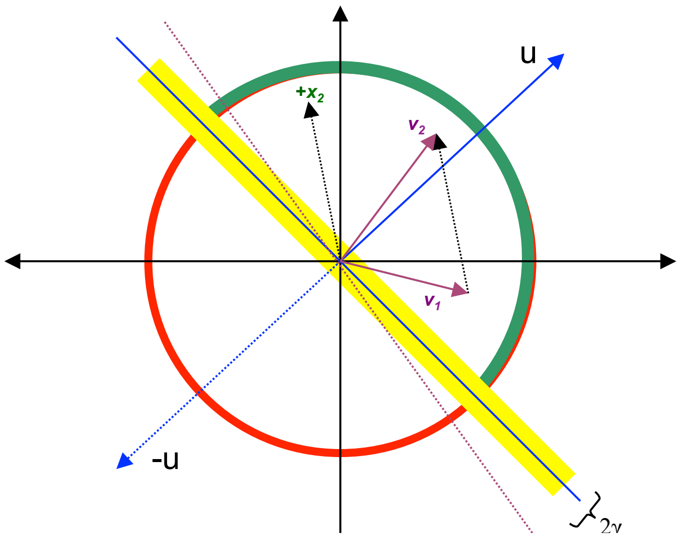
$$\mathbf{w}^{(\mathrm{avg},n)} = \sum_{i=1}^{S} \mu_{i,n} \mathbf{w}^{(i,n)}$$

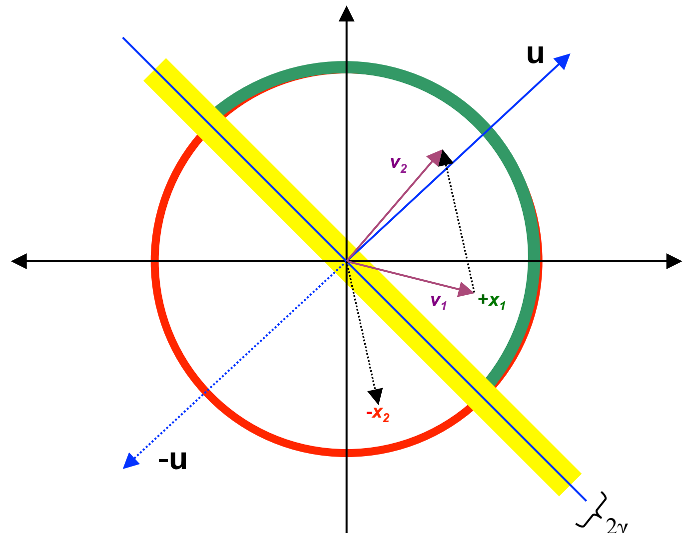
$$\begin{aligned}
\|\mathbf{w}^{(i,n)}\|^2 &= \|\mathbf{w}^{([i,n]-1)}\|^2 \\
&\quad + \|\mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{f}(\mathbf{x}_t, \mathbf{y}')\|^2 \\
&\quad + 2\mathbf{w}^{([i,n]-1)}(\mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{f}(\mathbf{x}_t, \mathbf{y}')) \\
&\leq \|\mathbf{w}^{([i,n]-1)}\|^2 + R^2 \\
&\leq \|\mathbf{w}^{([i,n]-2)}\|^2 + 2R^2 \\
\ldots &\leq \|\mathbf{w}^{(\mathrm{avg},n-1)}\|^2 + k_{i,n} R^2 \quad \text{(A2)}
\end{aligned}$$

This is not new….

(3a) The guess $\mathbf{v_2}$ after the two positive examples: $\mathbf{v}_2=\mathbf{v}_1+\mathbf{x}_2$

(3b) The guess $\mathbf{v_2}$ after the one positive and one negative example: $\mathbf{v}_2=\mathbf{v}_1-\mathbf{x}_2$





Lemma 2 $\forall k,\ \|\mathbf{v}_k\|^2 \leq kR$. *In other words, the norm of* $\mathbf{v}_k$ *grows "slowly", at a rate depending on* R.

Proof:

$$\mathbf{v}_{k+1} \cdot \mathbf{v}_{k+1} = (\mathbf{v}_k + y_i \mathbf{x}_i) \cdot (\mathbf{v}_k + y_i \mathbf{x}_i)$$

$$\Rightarrow \quad \|\mathbf{v}_{k+1}\|^2 = \|\mathbf{v}_{k+1}\|^2 + 2 y_i \mathbf{x}_i \cdot \mathbf{v}_k + y_i^2 \|\mathbf{x}\|^2$$

If mistake: $y_i\, \mathbf{x}_i\, \mathbf{v}_k < 0$

$$\Rightarrow \quad \|\mathbf{v}_{k+1}\|^2 = \|\mathbf{v}_{k+1}\|^2 + [\text{something negative}] + 1\|\mathbf{x}\|^2$$

$$\Rightarrow \quad \|\mathbf{v}_{k+1}\|^2 \leq \|\mathbf{v}_{k+1}\|^2 + \|\mathbf{x}\|^2$$

$$\Rightarrow \quad \|\mathbf{v}_{k+1}\|^2 \leq \|\mathbf{v}_{k+1}\|^2 + R^2 \qquad \forall \mathbf{x}_i \text{ given by } A,\ \|\mathbf{x}\|^2 < R^2$$

$$\Rightarrow \quad \|\mathbf{v}_k\|^2 \leq kR^2$$

Using A1/A2 we prove two inductive hypotheses:

$$\mathbf{u} \cdot \mathbf{w}^{(\mathrm{avg},N)} \geq \sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} \gamma \qquad \text{(IH1)}$$

$$\|\mathbf{w}^{(\mathrm{avg},N)}\|^2 \leq \sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} R^2 \qquad \text{(IH2)}$$

The base case is $\mathbf{w}^{(\mathrm{avg},1)}$, where we can observe:

$$\mathbf{u} \cdot \mathbf{w}^{\mathrm{avg},1} = \sum_{i=1}^{S} \mu_{i,1} \mathbf{u} \cdot \mathbf{w}^{(i,1)} \geq \sum_{i=1}^{S} \mu_{i,1} k_{i,1} \gamma$$

Follows from: $u.w^{(i,1)} >= k_{1,i}\ \gamma$

This is new …. We've never considered averaging operations before

IH1 inductive case:

$$\mathbf{w}^{(\mathrm{avg},n)} = \sum_{i=1}^{S} \mu_{i,n} \mathbf{w}^{(i,n)}$$

IH1

$$\begin{aligned}
\mathbf{u} \cdot \mathbf{w}^{(\mathrm{avg},N)} &= \sum_{i=1}^{S} \mu_{i,N} (\mathbf{u} \cdot \mathbf{w}^{(i,N)}) \\
&\geq \sum_{i=1}^{S} \mu_{i,N} (\mathbf{u} \cdot \mathbf{w}^{(\mathrm{avg},N-1)} + k_{i,N}\gamma) \\
&\overset{\cancel{\sum_i \mu_{i,N}}}{=} \mathbf{u} \cdot \mathbf{w}^{(\mathrm{avg},N-1)} + \sum_{i=1}^{S} \mu_{i,N} k_{i,N} \gamma \\
&\geq \left[\sum_{n=1}^{N-1} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} \gamma \right] + \sum_{i=1}^{S} \mu_{i,N} k_{i,N} \gamma \\
&= \sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} \gamma
\end{aligned}$$

From A1

Distribute

μ's are *distribution*

$$\boxed{\mathbf{u} \cdot \mathbf{w}^{(\mathrm{avg},N)} \geq \sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} \gamma \qquad \text{(IH1)}}$$

The first inequality uses A1, the second step $\sum_i \mu_{i,N} = 1$ and the second inequality the inductive hypothesis IH1.

Using A1/A2 we prove two inductive hypotheses:

$$\mathbf{u} \cdot \mathbf{w}^{(\mathrm{avg},N)} \geq \sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} \gamma \qquad \text{(IH1)}$$

$$\|\mathbf{w}^{(\mathrm{avg},N)}\|^2 \leq \sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} R^2 \qquad \text{(IH2)}$$

IH2 proof is similar

IH1 implies $\|\mathbf{w}^{(\mathrm{avg},N)}\| \geq \sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} \gamma$
since $\mathbf{u} \cdot \mathbf{w} \leq \|\mathbf{u}\| \|\mathbf{w}\|$ and $\|\mathbf{u}\| = 1$.

IH1, IH2 together imply the bound (as in the usual perceptron case)

Theorem 3. *Assume a training set $\mathcal{T}$ is separable by margin γ. Let $k_{i,n}$ be the number of mistakes that occurred on shard i during the nth epoch of training. For any N, when training the perceptron with iterative parameter mixing (Figure 3),*

$$\sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} \leq \frac{R^2}{\gamma^2}$$

Review/outline

- Streaming learning algorithms … and beyond
  - Naïve Bayes
  - Rocchio's algorithm
- Similarities & differences
  - Probabilistic vs vector space models
  - Computationally similar
  - Parallelizing Naïve Bayes and Rocchio
- Alternative:
  - Adding up contributions for every example vs conservatively updating a linear classifier
  - On-line learning model: mistake-bounds
    - some theory
    - a mistake bound for perceptron
  - Parallelizing the perceptron

What we know and don't know

Theorem 3. *Assume a training set $\mathcal{T}$ is separable by margin γ. Let $k_{i,n}$ be the number of mistakes that occurred on shard i during the nth epoch of training. For any N, when training the perceptron with iterative parameter mixing (Figure 3),*

uniform mixing…

$$\sum_{n=1}^{N}\sum_{i=1}^{S}\mu_{i,n}k_{i,n} \leq \frac{R^2}{\gamma^2} \qquad \Longrightarrow \qquad \sum_{n=1}^{N}\sum_{i=1}^{S}k_{i,n} \leq S \times \frac{R^2}{\gamma^2}$$

could we lose our speedup-from-parallelizing to slower convergence?

What we know and don't know

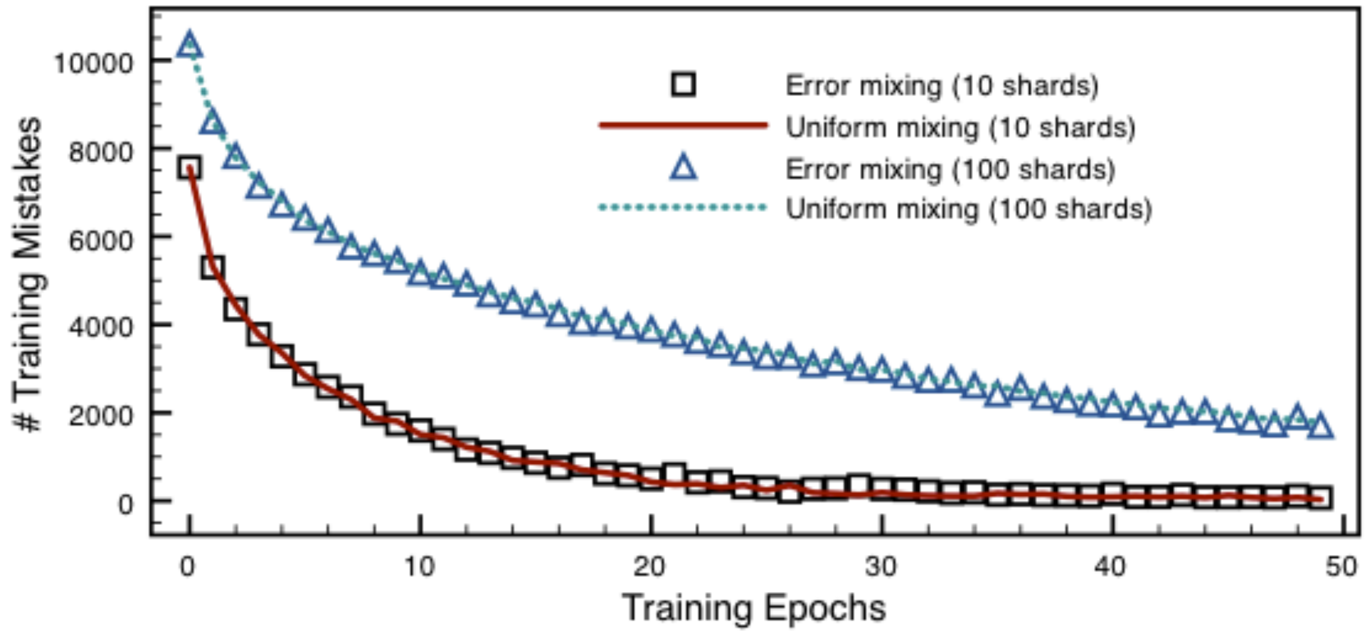


Figure 6: Training errors per epoch for different shard size and parameter mixing strategies.

What we know and don't know

Thus, for cases where training errors are uniformly distributed across shards, it is possible that, in the worst-case, convergence may slow proportional the the number of shards. This implies a trade-off between slower convergence and quicker epochs when selecting a large number of shards. In fact, we observed a tipping point for our experiments in which increasing the number of shards began to have an adverse effect on training times, which for the named-entity experiments occurred around 25-50 shards. This is both due to reasons described in this section as well as the added overhead of maintaining and summing multiple high-dimensional weight vectors after each distributed epoch.

What we know and don't know

In this paper we have investigated distributing the structured perceptron via simple parameter mixing strategies. Our analysis shows that an iterative parameter mixing strategy is both guaranteed to separate the data (if possible) and significantly reduces the time required to train high accuracy classifiers. However, there is a trade-off between increasing training times through distributed computation and slower convergence relative to the number of shards.

Review/outline

- Streaming learning algorithms … and beyond
  - Naïve Bayes
  - Rocchio's algorithm
- Similarities & differences
  - Probabilistic vs vector space models
  - Computationally similar
  - Parallelizing Naïve Bayes and Rocchio
- Alternative:
  - Adding up contributions for every example vs conservatively updating a linear classifier
  - On-line learning model: mistake-bounds
    - some theory
    - a mistake bound for perceptron
  - Parallelizing the perceptron

Where we are…

- Summary of course so far:
  - Math tools: complexity, probability, on-line learning
  - Algorithms: Naïve Bayes, Rocchio, Perceptron, Phrase-finding as BLRT/pointwise KL comparisons, …
  - Design patterns: stream and sort, messages
    - How to write scanning algorithms that scale linearly on large data (memory does not depend on input size)
  - Beyond scanning: parallel algorithms for ML
  - Formal issues involved in parallelizing
    - Naïve Bayes, Rocchio, … easy?
    - Conservative on-line methods (e.g., perceptron) … hard?
- Next: practical issues in parallelizing
  - details on Hadoop