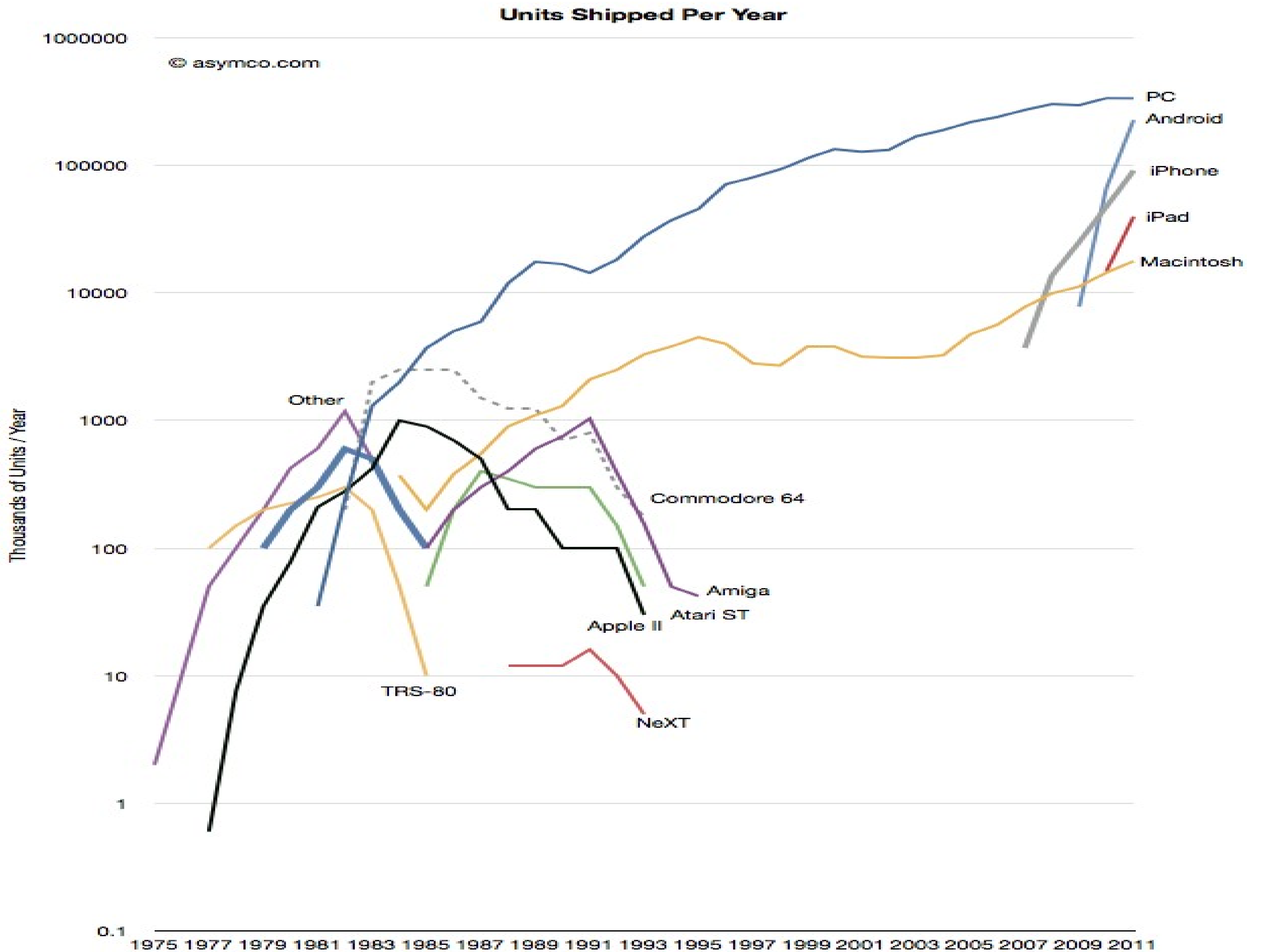
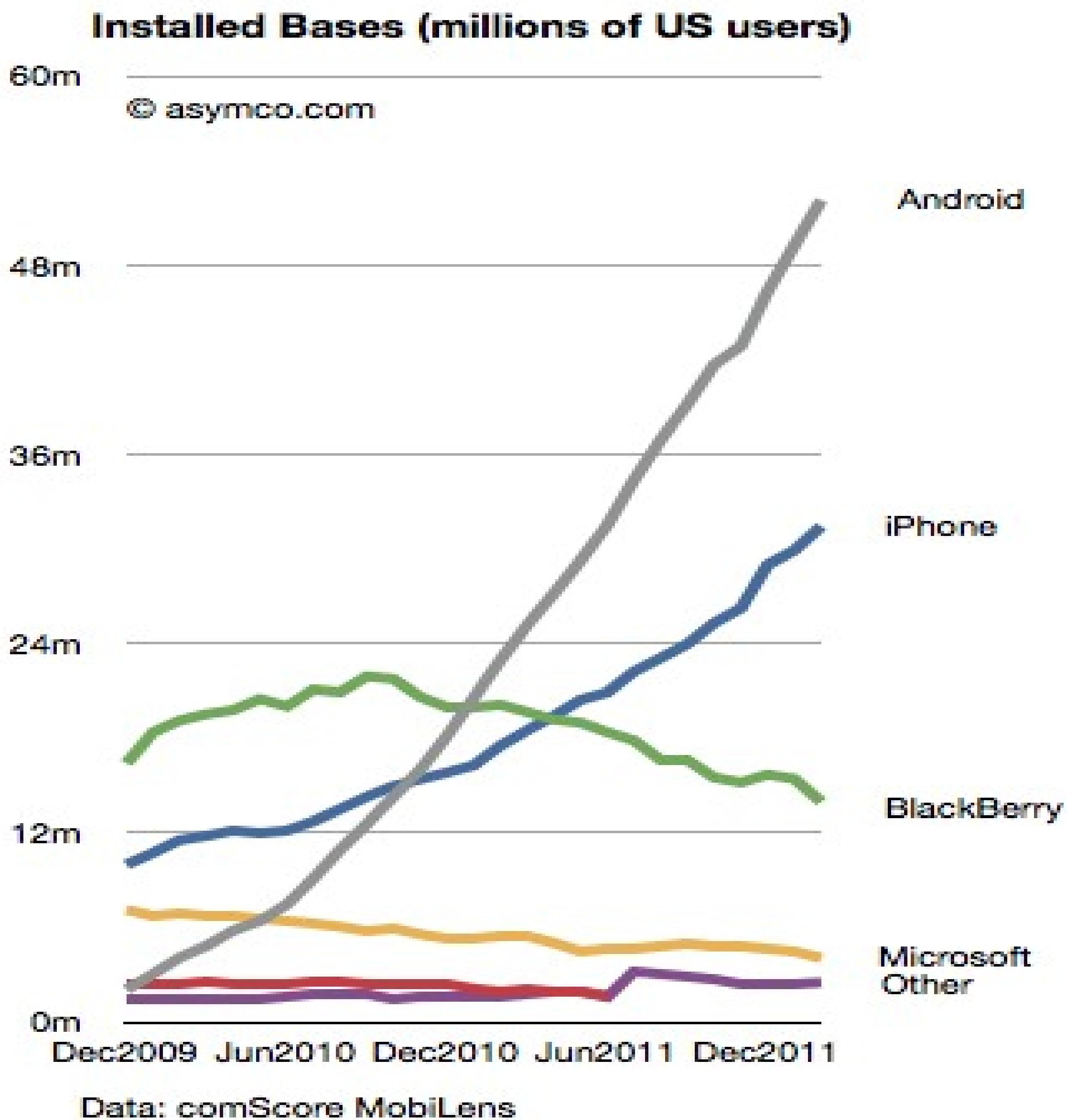
Learning in networks of millions of nodes

Márk Jelasity



Units Shipped Per Year
© asymco.com
Thousands of Units / Year
1000000
100000
10000
1000
100
10
1
0.1
1975 1977 1979 1981 1983 1985 1987 1989 1991 1993 1995 1997 1999 2001 2003 2005 2007 2009 2011
PC
Android
iPhone
iPad
Macintosh
Other
Commodore 64
Amiga
Atari ST
Apple II
TRS-80
NeXT


Installed Bases (millions of US users)
© asymco.com
60m
48m
36m
24m
12m
0m
Dec2009 Jun2010 Dec2010 Jun2011 Dec2011
Android
iPhone
BlackBerry
Microsoft
Other
Data: comScore MobiLens

Motivation

- Explosive growth of smart phone platforms, and
- Availability of sensor and other contextual data
- Makes collaborative data mining possible
  - Health care: following and predicting epidemics, personal diagnostics
  - Smart city: traffic optimization, accident forecasting
  - (predicting earthquakes, financial applications, etc)
- P2P networks, grid, etc, are also relevant platforms

P2P system model

- Large number (millions or more) computers (nodes)
- Packet switched communication
  - Every node has an address
  - Any node can send a message to any given address
- Messages can be delayed or lost, nodes can crash
- (in parallel computing this is similar to the model of asynchronous chaotic iterations)

Fully distributed data

- Horizontal data distribution
- Every node has very few records, we assume they have only one
- We do not allow for moving data, only local processing (privacy preservation)
- We require that the models are cheaply available for all the nodes

Illustration: averaging

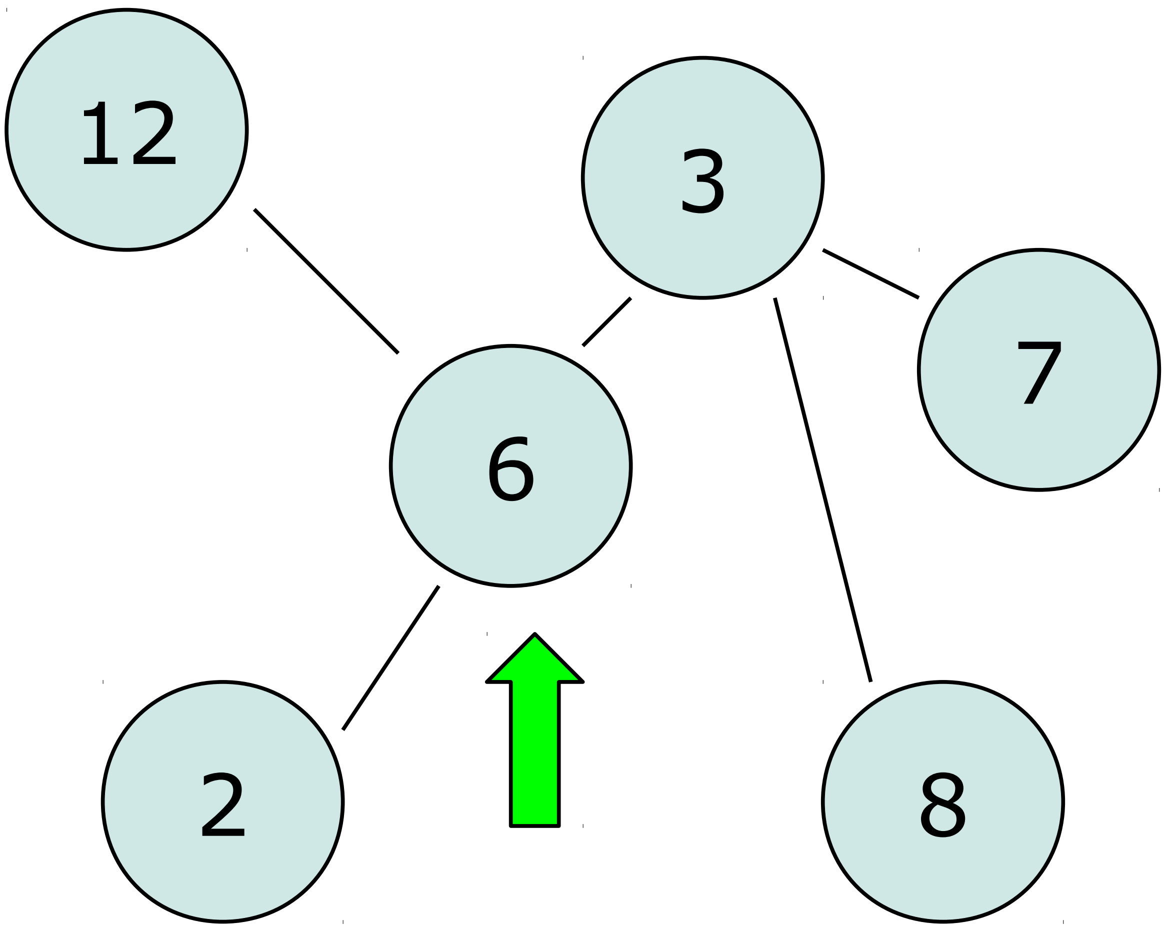


Illustration: averaging

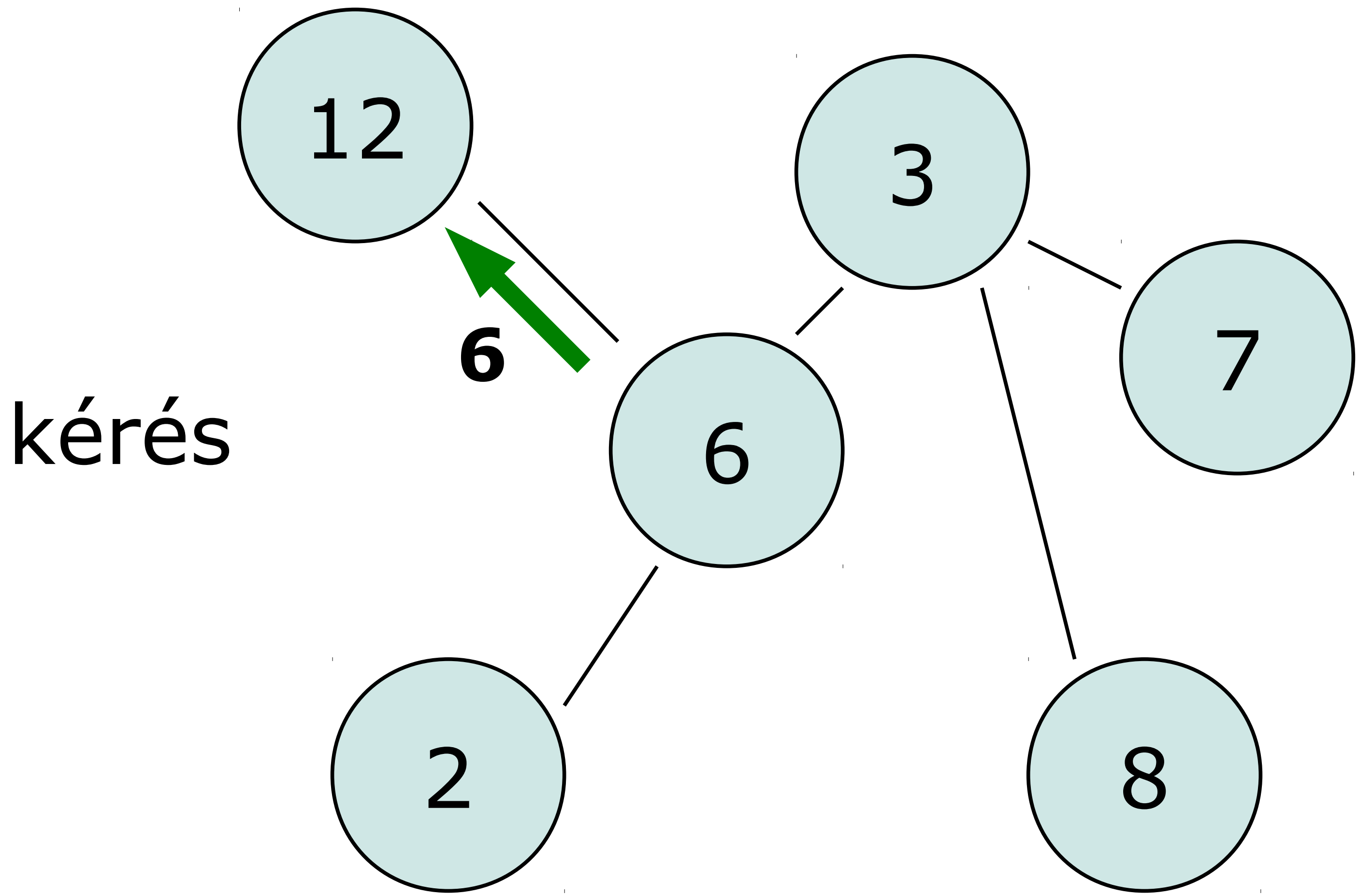
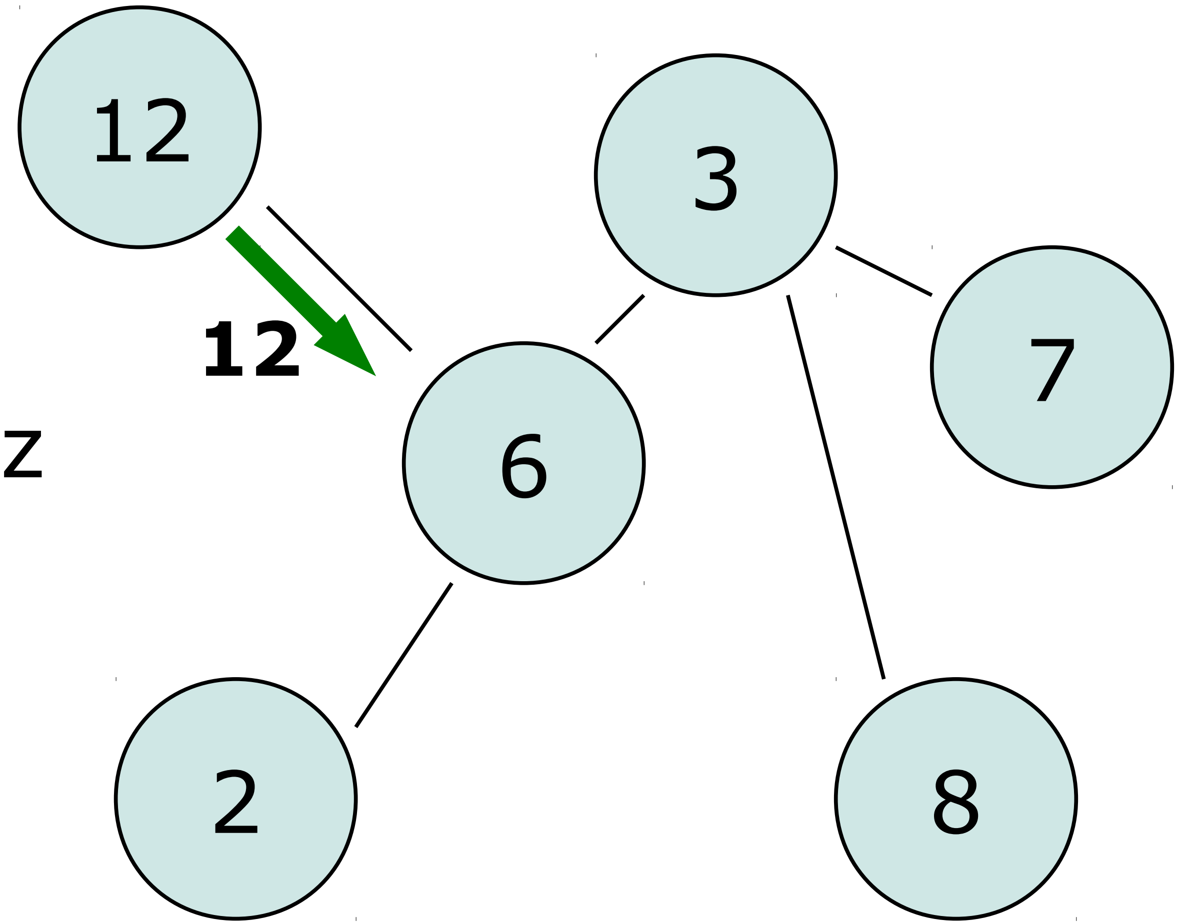
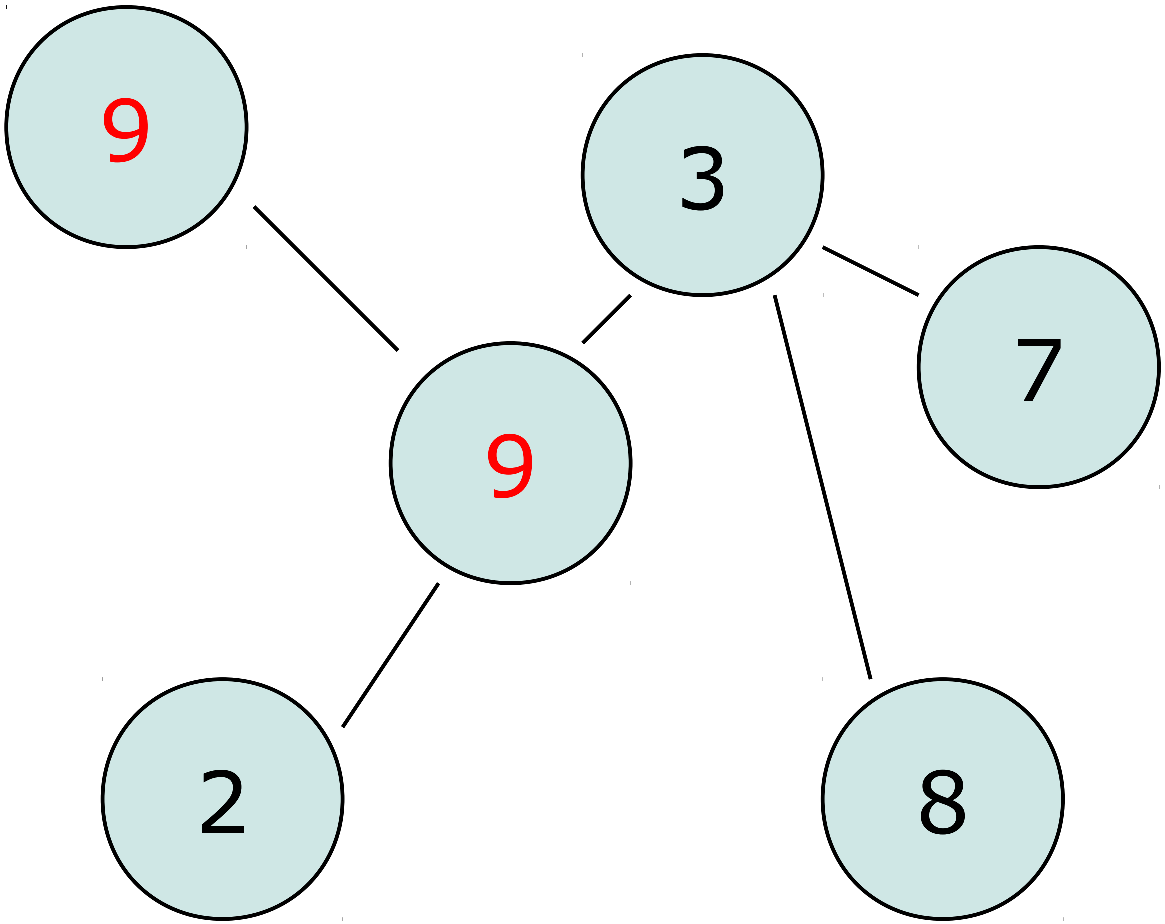


Illustration: averaging



válasz

Illustration: averaging

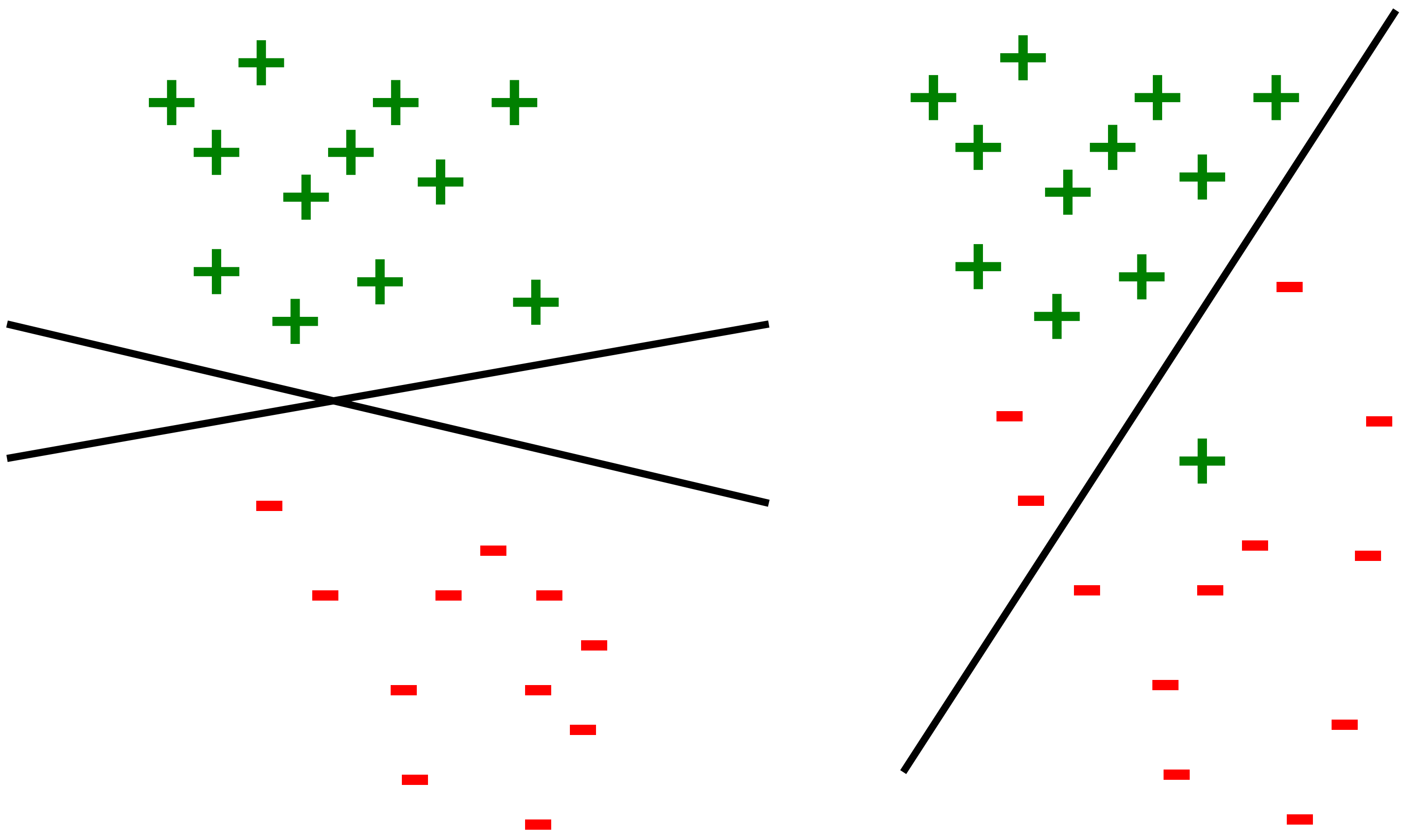


(12+6)/2=9

Classification problem in machine learning

- We are given a set of (x_i, y_i) examples, where y_i is the class of x_i (y_i is eg. -1 or 1)
- We want a model f(), such that for all i, $f(x_i)=y$
- f() is very often a parameterized function $f_w()$, and the classification problem becomes an error minimization problem in w.
  - Neural net weights, linear model parameters, etc
- The error is often defined as a sum of errors over the examples

Illustration of classification with a linear model



Stochastic gradient descent

- Assume the error is defined as

$$Err(w) = \sum_{i=1}^{n} Err(w, x_i)$$

- Then the gradient is

$$\frac{\partial Err(w)}{\partial w} = \sum_{i=1}^{n} \frac{\partial Err(w, x_i)}{\partial w}$$

- So the full gradient method looks like

$$w(t+1) = w(t) - \alpha(t) \sum_{i=1}^{n} \frac{\partial Err(w, x_i)}{\partial w}$$

- But one can take only one example at a time iterating in random order over examples

$$w(t+1) = w(t) - \alpha(t) \frac{\partial Err(w, x_i)}{\partial w}$$

Fully distributed classification

- So the problem is to find an optimization method that fits into our system and data model
- Most distributed methods build local models and then combine these through ensemble learning: but we don't have enough local data
- Online algorithms
  - Need only one data record at a time
  - They update the model using this record
- The stochastic gradient method is a popular online learning algorithm (we apply it to the primal form of the SVM error function)

Gossip learning

Algorithm 1 Gossip Learning Scheme

```
initModel()
loop
    wait(Δ)
    p ← selectPeer()
    currentModel ← createModel()
    send currentModel to p
end loop

procedure ONRECEIVEMODEL(m)
    modelQueue.add(m)
end procedure
```

```
procedure CREATEMODELRW
    m ← modelQueue.first()
    update(m)
    return m
end procedure

procedure CREATEMODELMU
    m1 ← modelQueue.first()
    m2 ← modelQueue.second()
    m ← merge(m1, m2)
    update(m)
    return m
end procedure
```

The merge function

- Let z=merge(x,y)=(x+y)/2 (x and y are linear models)
- In the case of the Adaline perceptron
  - Updating z using an example has the same effect as updating x and y with the same example and then averaging these two updated models
  - Making predictions using z is the same as calculating the weighted average of the predictions of x and y
- This means we effectively propagate an exponential number of models, and the voting of these is our prediction
- For the linear SVM algorithm this is only a heuristic argument

Local prediction

- We use only local models
  - The current model
  - Or voting over a number of recent models

```
procedure PREDICT(x)
    w ← currentModel
    return sign(⟨w, x⟩)
end procedure

procedure VOTEDPREDICT(x)
    pRatio ← 0
    for m ∈ modelQueue do
        if sign(⟨m.w, x⟩) ≥ 0 then
            pRatio ← pRatio +1
        end if
    end for
    return sign(pRatio/modelQueue.size()−0.5)
end procedure
```

Experiments

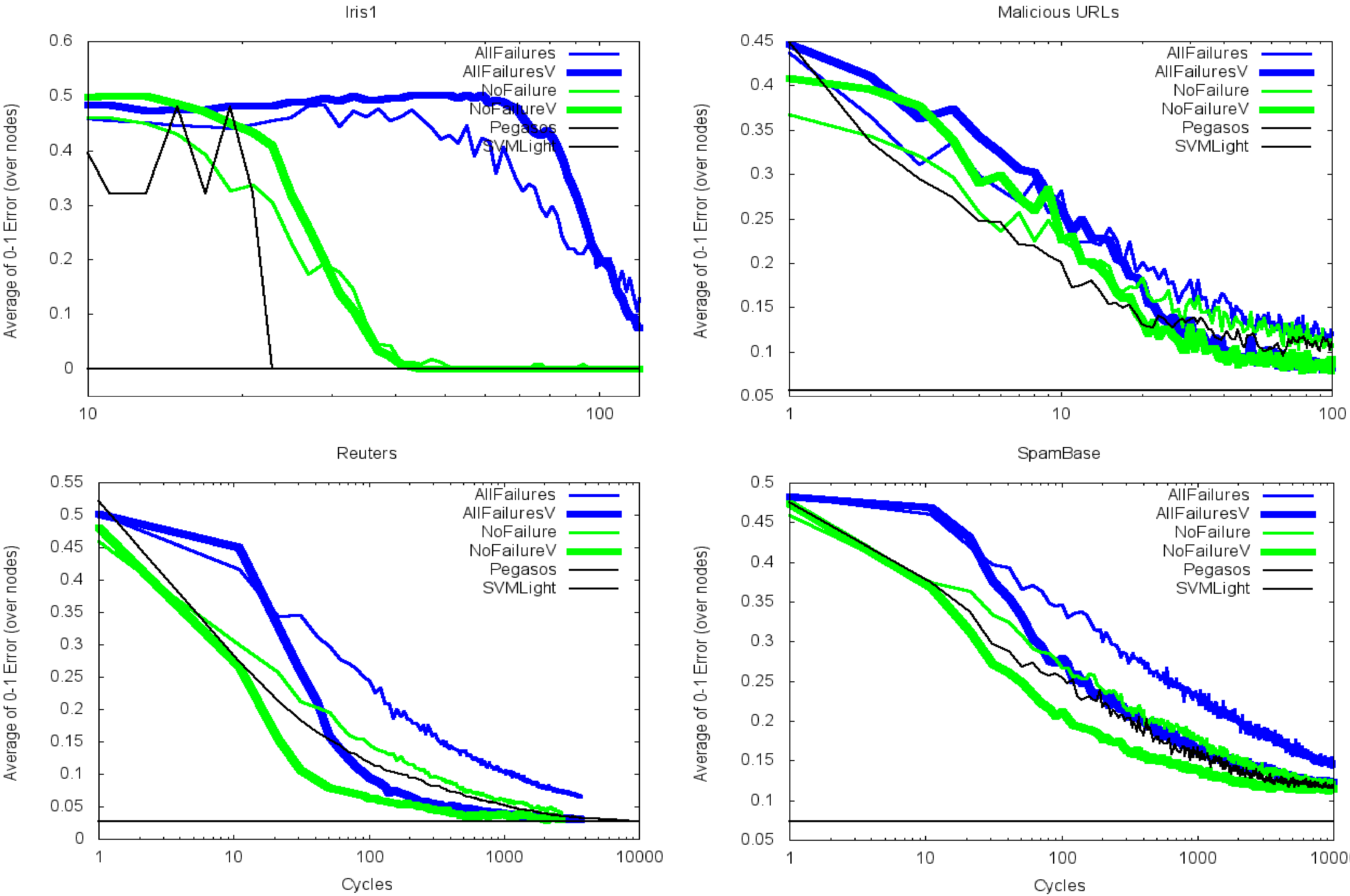
- We implemented a support vector machine with stochastic gradient (Pegasos alg.)
- We used several benchmark data sets for evaluations
  - Data is fully distributed: one data point per node
- We used extreme scenarios
  - 50% message drop rate
  - 1-10 cycles of message delay
  - Churn modeled after the FileList.org trace from Delft

Data sets

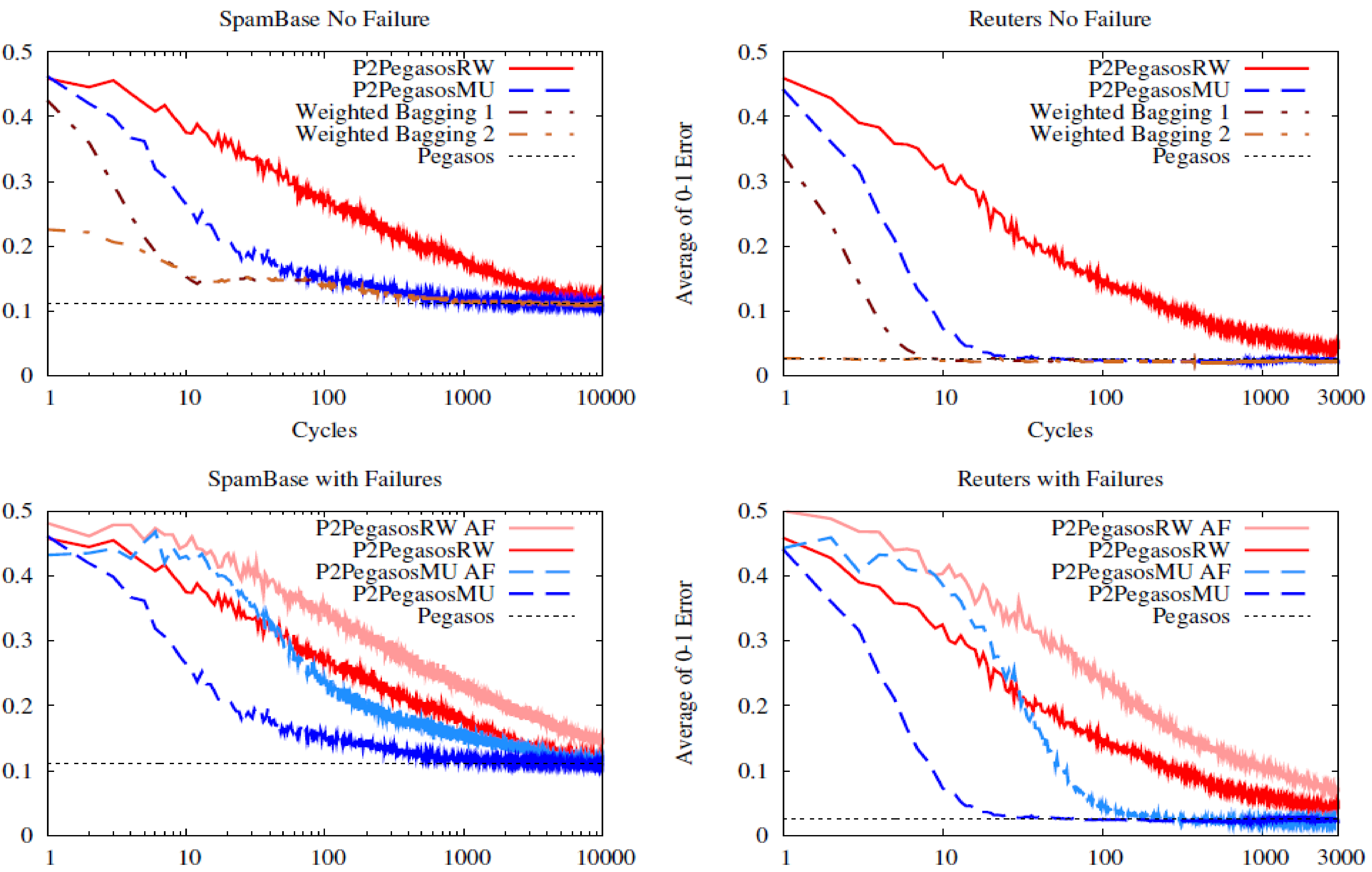
	Iris1	Iris2	Irirs3	Reuters	SpamBase	Malicious10
Training set size	90	90	90	2000	4140	2155622
Test set size	10	10	10	600	461	240508
Number of features	4	4	4	9947	57	10
Classlabel ratio	50/50	50/50	50/50	1300/1300	1813/2788	792145/1603985
Pegasos 20000 iter.	0	0	0	0.025	0.111	0.080 (0.081)
Pegasos 1000 iter.	0	0	0.4	0.057	0.137	0.095 (0.060)
SVMLight	0	0	0.1	0.027	0.074	0.056 (–)

- Statistics of data sets
- The performance of some known algorithms

Without merge



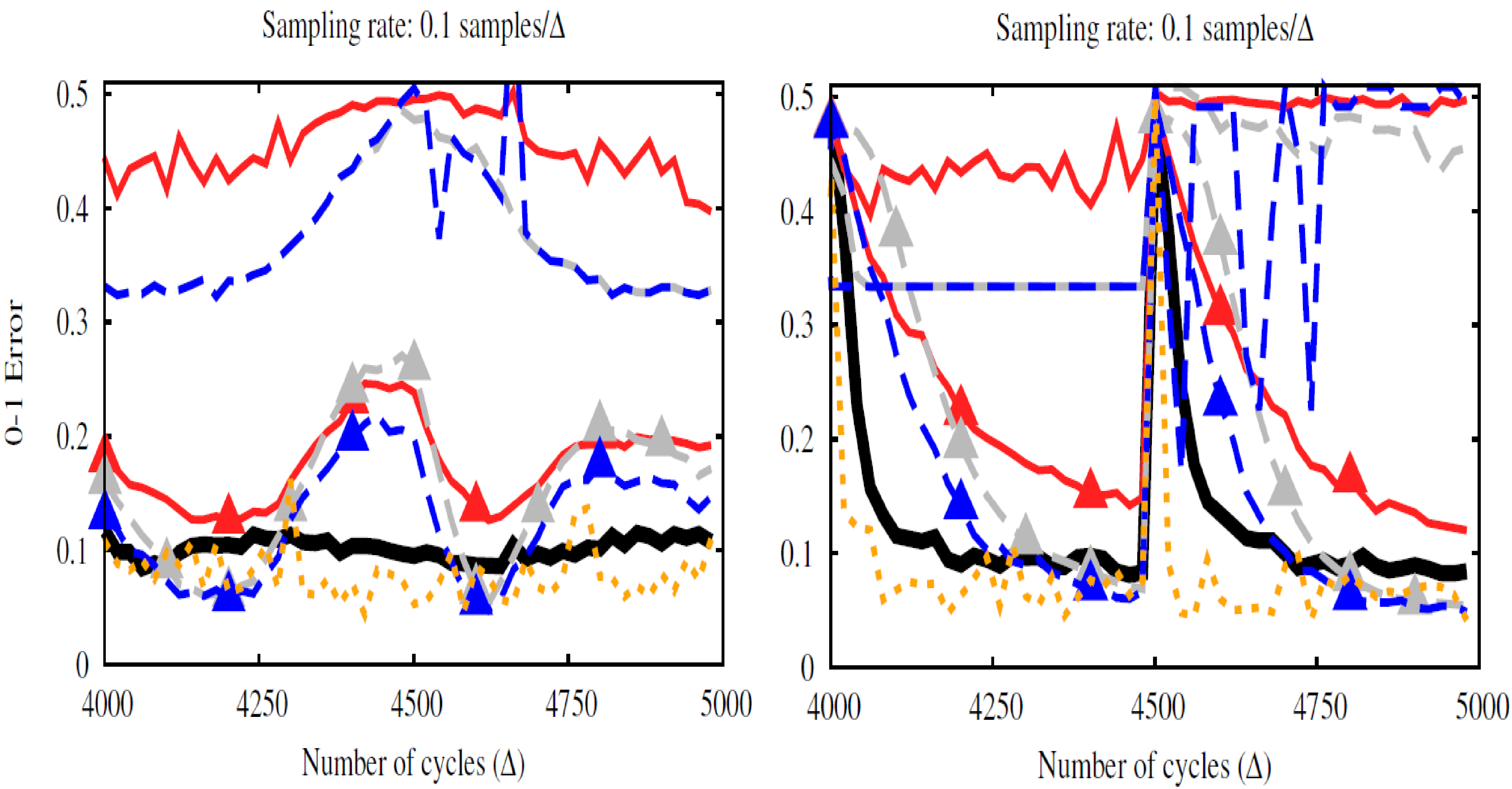
With merge


SpamBase No Failure
P2PegasosRW
P2PegasosMU
Weighted Bagging 1
Weighted Bagging 2
Pegasos
0.5
0.4
0.3
0.2
0.1
0
1
10
100
1000
10000
Cycles
Reuters No Failure
Average of 0-1 Error
P2PegasosRW
P2PegasosMU
Weighted Bagging 1
Weighted Bagging 2
Pegasos
1
10
100
1000
3000
Cycles
SpamBase with Failures
P2PegasosRW AF
P2PegasosRW
P2PegasosMU AF
P2PegasosMU
Pegasos
Reuters with Failures
Average of 0-1 Error
P2PegasosRW AF
P2PegasosRW
P2PegasosMU AF
P2PegasosMU
Pegasos

Additional results

- We implemented multiclass boosting in the gossip framework
- We developed techniques for dealing with concept drift
  - The algorithms is running continuously
  - We keep the age distribution of models fixed
  - At any point in time we have good models

Adaptivity



Sampling rate: 0.1 samples/Δ
0-1 Error
Number of cycles (Δ)
Sampling rate: 0.1 samples/Δ
Number of cycles (Δ)

Publications

- Róbert Ormándi, István Hegedűs, and Márk Jelasity. Asynchronous peer-to-peer data mining with stochastic gradient descent. In Emmanuel Jeannot, Raymond Namyst, and Jean Roman, editors, Euro-Par 2011, volume 6852 of Lecture Notes in Computer Science, pages 528–540. Springer-Verlag, 2011.
- Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip learning with linear models on fully distributed data. Concurrency and Computation: Practice and Experience, 2012. to appear.
- István Hegedűs, Róbert Busa-Fekete, Róbert Ormándi, Márk Jelasity, and Balázs Kégl. Peer-to-peer multi-class boosting. In Euro-Par 2012, Lecture Notes in Computer Science. Springer-Verlag, 2012. to appear.

Remarks regarding the chaotic model

- If uniformity of random walk is guaranteed, then all the models converge to the true model eventually, irrespective of all failures
- If no uniformity can be guaranteed, but the local data is statistically independent of the visiting probability, then again convergence to the true model is guaranteed in general for all models
- If no uniformity and no independence could be guaranteed, convergence to a good model is still ensured provided that the data is separable, and all misclassified examples are visited "often enough"