CASCADING VECTOR MACHINES

Carlos Riquelme, Lan Nguyen, Sven Schmit

CME 323

OUTLINE

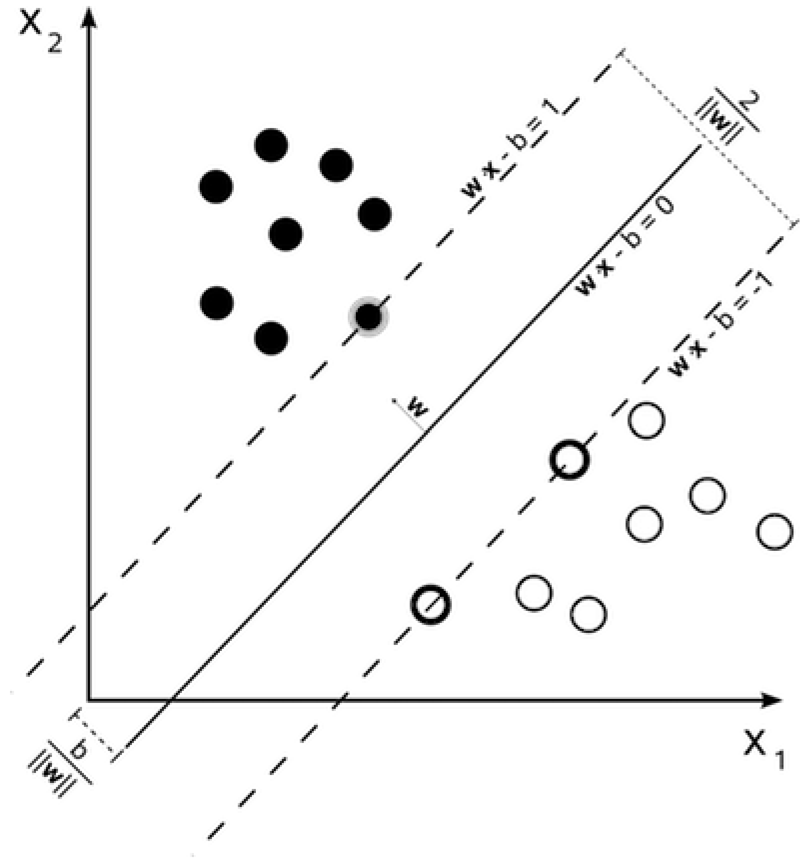
- Support vector machines
- Kernel SVM
- How to parallelize in pySpark
- Experiments
- Take aways

SUPPORT VECTOR MACHINES

General method for **regression** and **classification**

BINARY CLASSIFICATION

Find **hyperplane** that maximizes **margin**



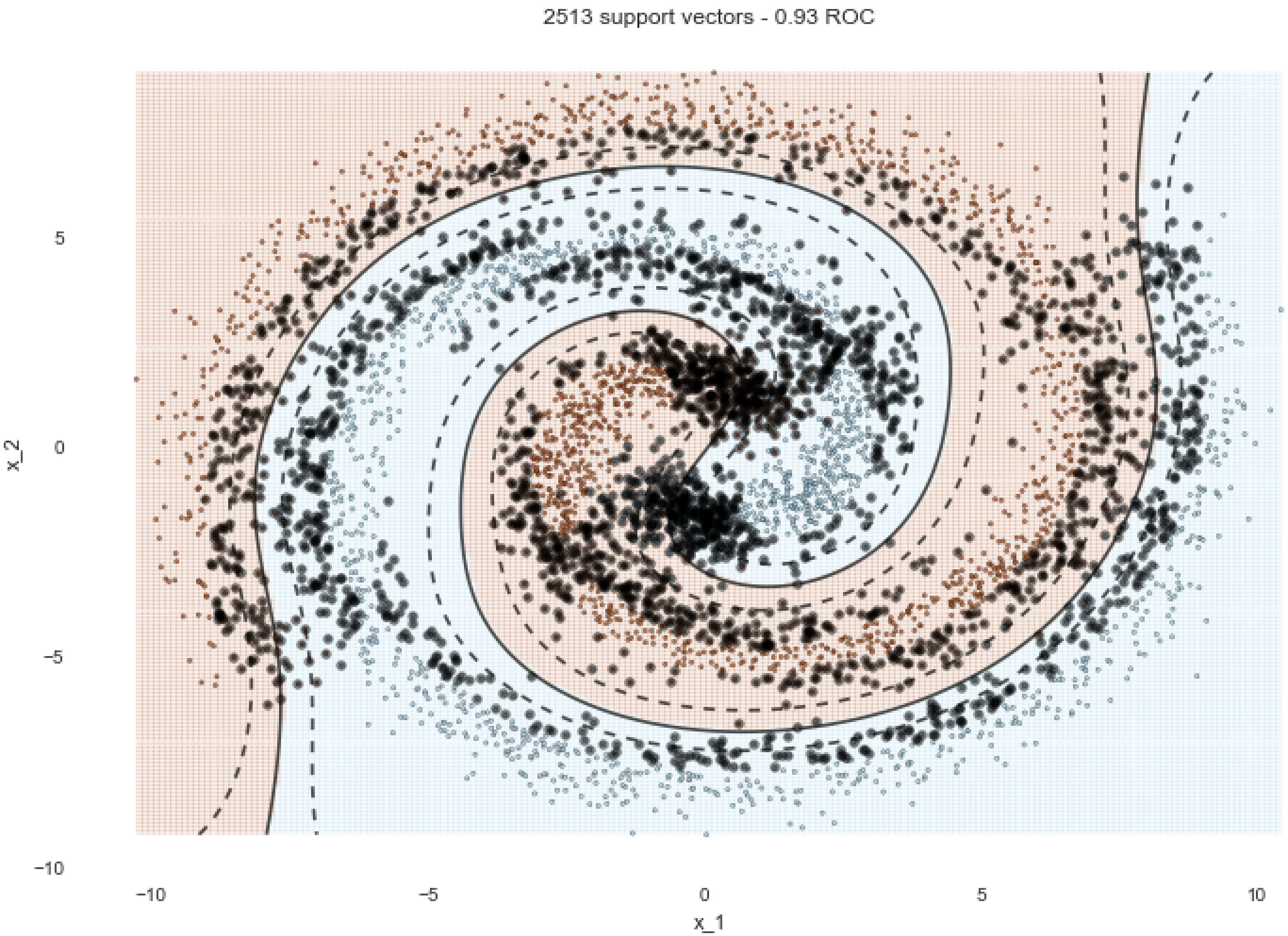
Support vectors

SUPPORT VECTORS $\Leftrightarrow$ SOLUTION

No change if we (re)move other observations

KERNEL SVM

Find (linear) hyperplane in higher/**infinite dimensional** space



5000 data points

HOW DOES KERNEL SVM SCALE?

requires *kernel matrix* $K \in \mathbb{R}^{n \times n}$

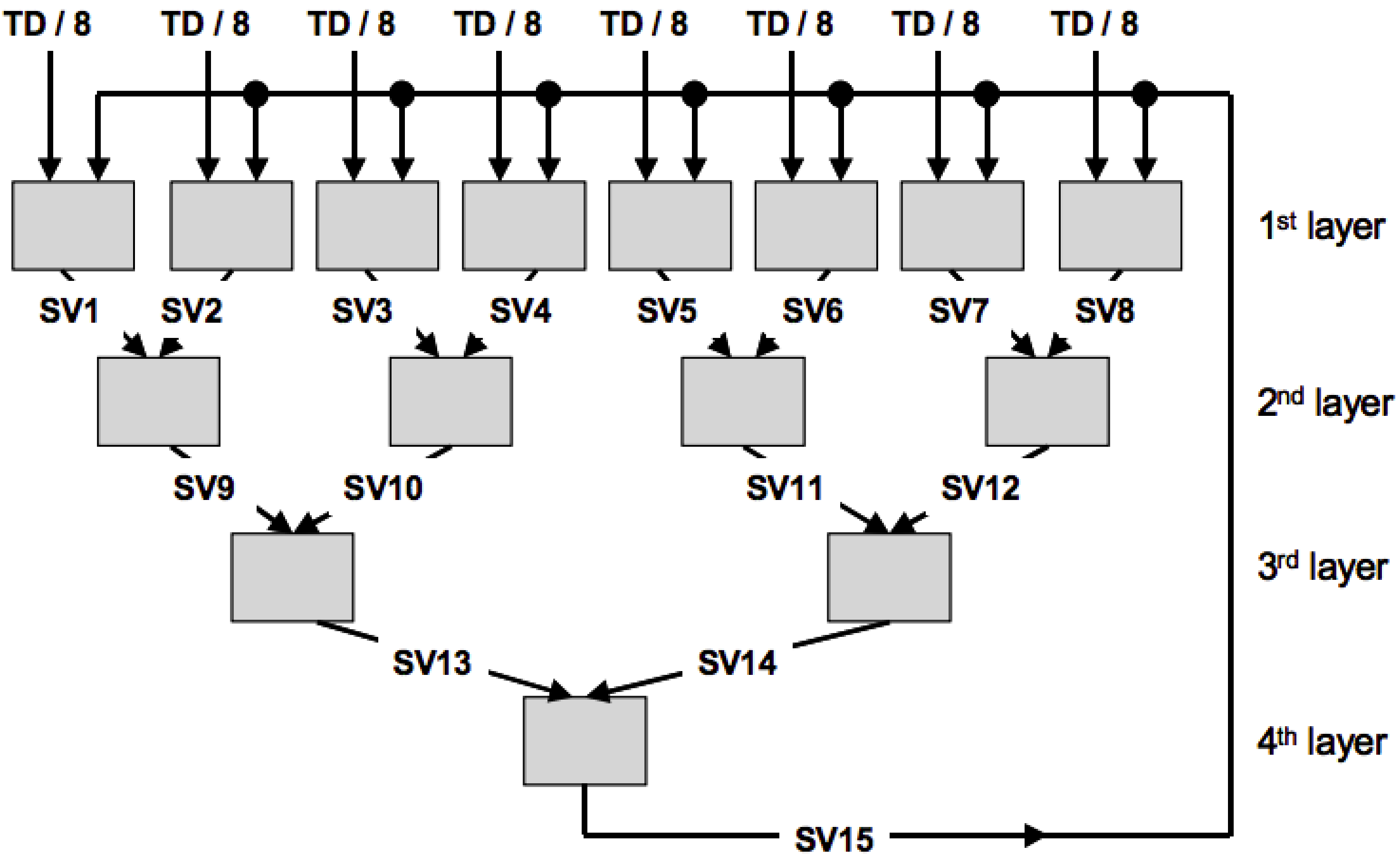
INFEASIBLE FOR LARGE n

libsvm QP solver runs in $\Theta(n^2 p)$

CASCADE SVM

Key: Only points on the margin are relevant [1]

THROW AWAY IRRELEVANT POINTS EARLY


TD / 8
TD / 8
TD / 8
TD / 8
TD / 8
TD / 8
TD / 8
TD / 8
1st layer
SV1
SV2
SV3
SV4
SV5
SV6
SV7
SV8
2nd layer
SV9
SV10
SV11
SV12
3rd layer
SV13
SV14
4th layer
SV15

CODE FOR SINGLE PASS

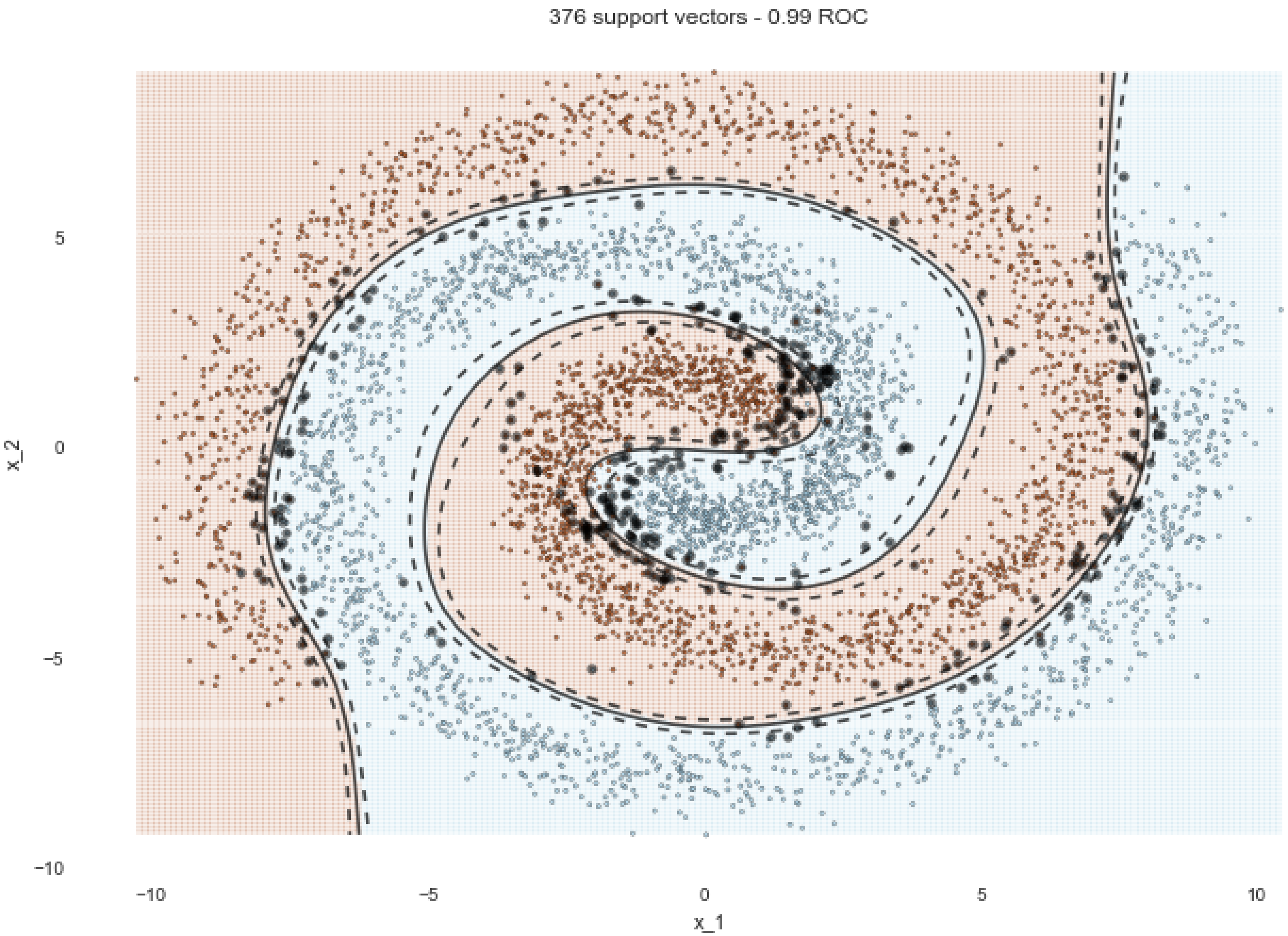
```
def cascade(labeledPointRDD, reducer, nmax):
    n = labeledPointRDD.count()
    numPartitions = int(2**(np.ceil(np.log(n / nmax) / np.log(2.0))))
    leafsRDD = labeledPointRDD.repartition(numPartitions)

    while numPartitions > 1:
        numPartitions = int(numPartitions / 2)

        # need cache against lazy evaluation
        leafsRDD = leafsRDD.mapPartitions(reducer, True) \
                           .coalesce(numPartitions) \
                           .cache()

    return leafsRDD.collect()
```

reducer: fit SVM and keep support vectors



5000 data points

CASCADE X

Can apply same cascade to other procedures

- L1VM
- Kernel Logistic Regression with l_1 penalty
- etc.

ALTERNATIVES

- Subsample data
- Low-rank approximation of K
- Big memory machine

PARALLELIZATION

HOW TO REPRESENT DATA

Every observation is a `LabeledPoint`

Every partition contains a subset of the observations

SCALABILITY

Reduce complexity in n, keep complexity in d

Assumption we can solve SVM of size $\mathcal{O}(\sqrt{n})$, then:

- number of partitions $k \sim \mathcal{O}(\sqrt{n})$
- number of levels $L \sim \mathcal{O}(\log(n))$

RUN TIME

Solve SVM in $\mathcal{O}(dn^{\alpha})$, for $2 < \alpha < 3$, on single machine

---

CASCADE SVM:

$$\mathcal{O}(dn^{\alpha/2}\, log(n)) < \mathcal{O}(dn^{3/2}\, log(n))$$

Reduction factor of $n^{\alpha/2} / \log(n)$

COMMUNICATION TYPES

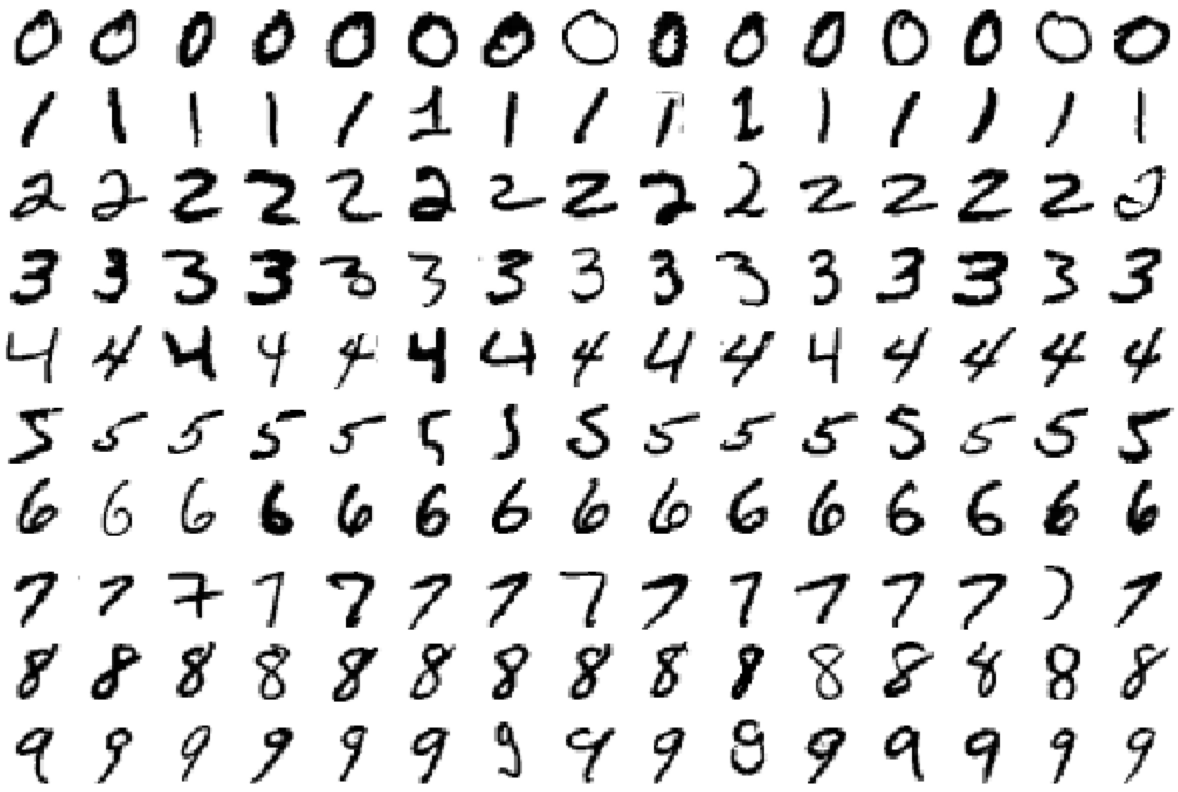
- Repartition: all-to-all
- Coalesce: merge 2 partitions
- Broadcast model: 1-to-all

COMMUNICATION COST

- Repartition data: dn
- Coalesce: $\frac{d(2\sqrt{n}-1)\sqrt{n}}{4} = \mathcal{O}(dn)$
- Distribute model: $d\sqrt{n}$

PERFORMANCE

MNIST



60k training set, 10k test set

BENCHMARKS

- **Lower bound**: subsample data
- **Upper bound**: fit SVM on full dataset

REGULAR SVM

- 2k subsample: 6.5% error
- 10k subsample: 3.5% error
- 60k full sample: 1.7% error

CASCADE SVM

- 2k svms: 4.6% error
- 10k svms: 2.1% error

[1] show optimality with multiple loops

TAKE AWAYS

- Using cascades we can parallelize SVMs
- Good if number of SV $< \sqrt{n}$
- Can extend to similar 'kernel' methods

REFERENCES

[1] Graf, Hans P., et al. "Parallel support vector machines: The cascade svm." *Advances in neural information processing systems*. 2004.

QUESTIONS?