Low-Dimensional Feature Learning with Kernel Construction

Artem Sokolov, LIMSI-CNRS, Orsay, France
Tanguy Urvoy, Orange Labs, Lannion, France
Hai-Son Le, LIMSI-CNRS, Orsay, France

Highlights

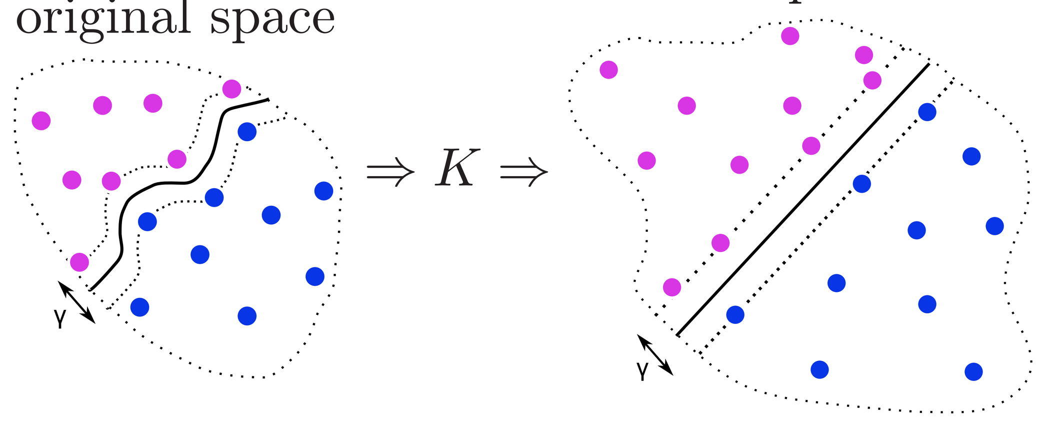
WWW – millions of features

- **too much data**:
  - impossible to keep all on disk
  - necessary to learn on it
  - learning task unknown beforehand
- reduce data to few **informative** features
- **reduced data must permit learning**

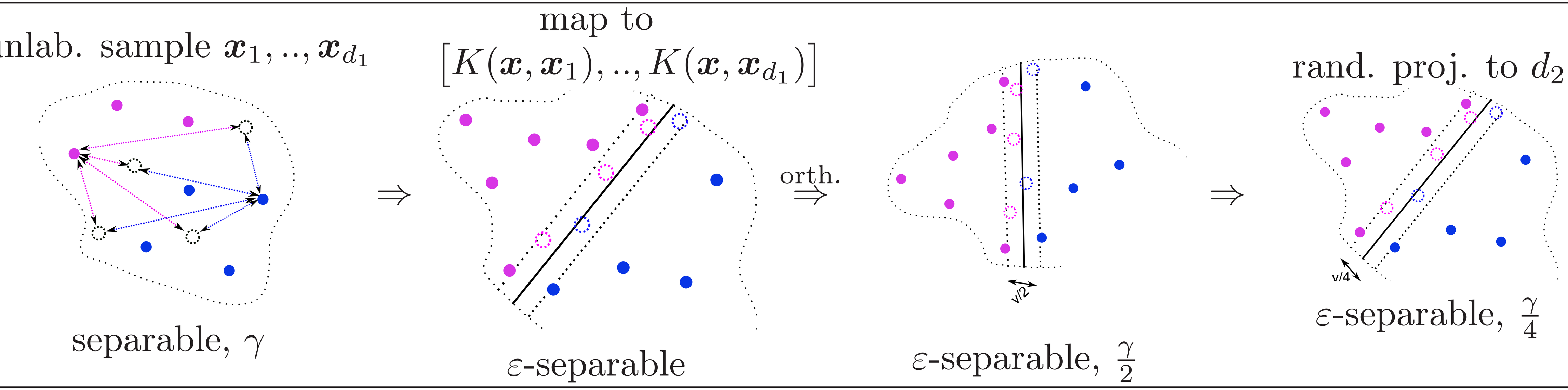
★ **2nd and 3rd place** in Semi-Supervised Feature Learning Challenge (SSFL)

★ **faithful submissions:**
  - using unlabeled data
  - not using test data for self-tuning
  - no "single feature" trick (for 3rd place)

Kernels as features (used here as a "black-box" procedure) [Balcan et al., 2004]



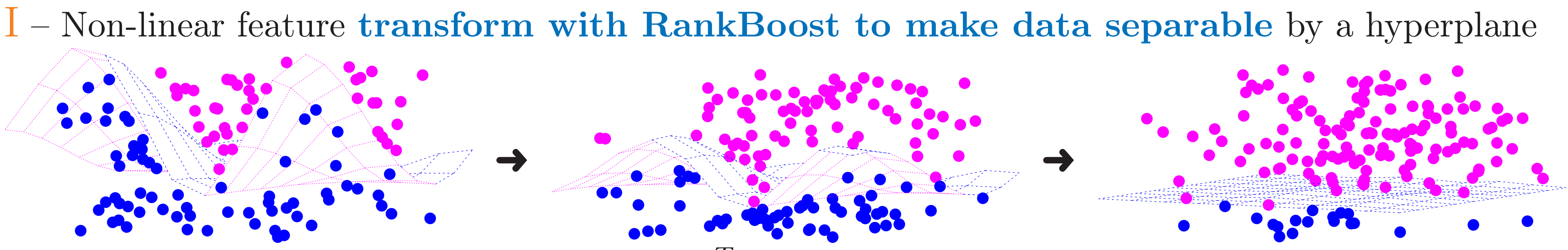
- **Φ is nice** (lin. separable with γ) **but high-dim.**
- **need fewer dimensional** feature space
- rnd. proj. to $d = O(\frac{1}{\gamma^2}\log\frac{1}{\varepsilon\delta})$? $\mathbf{\Phi}$ may be implicit
- **how to exploit unlabeled data?**



- $d_1 = O(\frac{1}{\varepsilon}[\frac{1}{\gamma^2} + \ln\frac{1}{\delta}])$
- $d_2 = O(\frac{1}{\gamma^2}\log\frac{1}{\varepsilon\delta})$
- **good kernel ≃ good features**
- applicable even if K is not a kernel (without guarantees)

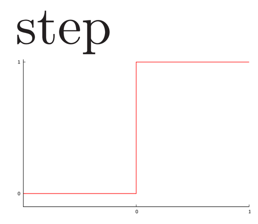
Contrib: Tune kernels with Boosting & Neural Networks

I – Non-linear feature **transform with RankBoost to make data separable** by a hyperplane

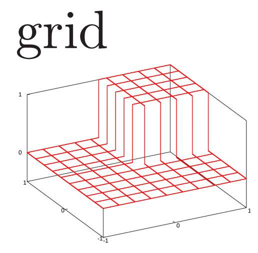


- Seeking scoring function as: $H(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$
- Optimizing pair-wise loss, equivalent to AUC: $L = \sum_{y_i < y_j} [\![H(\boldsymbol{x}_i) > H(\boldsymbol{x}_j)]\!]$
- Weak rankers $h(\boldsymbol{x})$: decision stumps and grids (trees of depth 2)
- Use **weak outputs as new features** $\mathbf{\Phi}(\boldsymbol{x}) = (\alpha_1 h_1(\boldsymbol{x}), \ldots, \alpha_T h_T(\boldsymbol{x}))$
- H can be **viewed as linear discrim. rule** $H(\boldsymbol{x}) = \langle \boldsymbol{w}, \mathbf{\Phi}(\boldsymbol{x})\rangle$ for $\boldsymbol{w} = \bar{1}$
- T may be too big to store training $\mathbf{\Phi}(\boldsymbol{x}_i)$ + **want to exploit unlabeled data**⇒
- **define kernel** as $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \langle \mathbf{\Phi}(\boldsymbol{x}_1), \mathbf{\Phi}(\boldsymbol{x}_2)\rangle$ ⇒ **use "black-box"**

step



grid

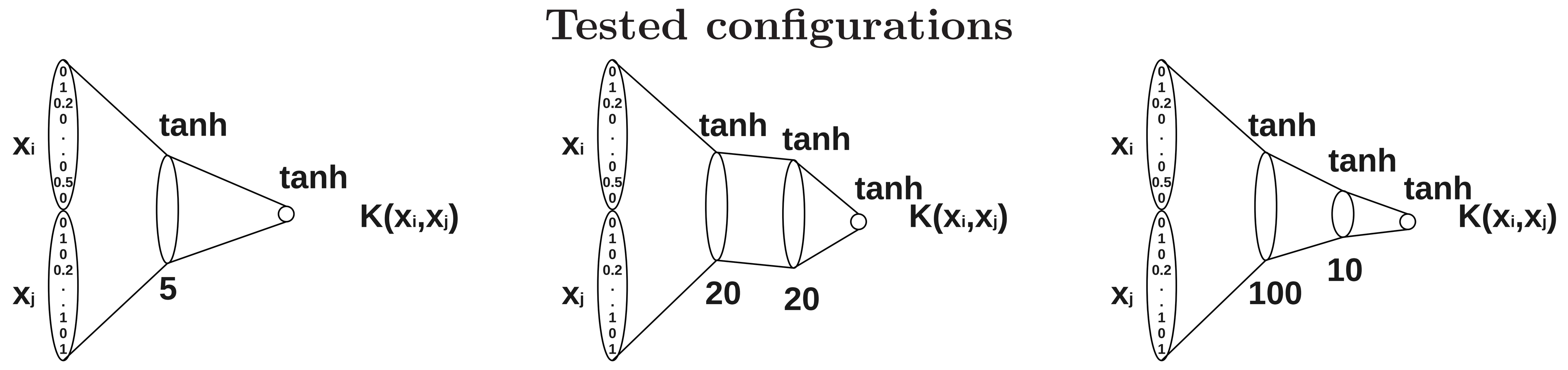


II – Neural network K to optimize **kernel alignment** with perfect kernel $K'(\boldsymbol{x}_i, \boldsymbol{x}_j) = y_i y_j$

$$A(K, K') = K \cdot K' / \sqrt{(K \cdot K)(K' \cdot K')} \qquad K \cdot K' = \sum_{i,j} K(\boldsymbol{x}_i, \boldsymbol{x}_j) y_i y_j$$

- Optimize **quadratic error**: $\sum_{\boldsymbol{x}_i, \boldsymbol{x}_j} (K(\boldsymbol{x}_i, \boldsymbol{x}_j) - y_i y_j)^2$ with stochastic gradient descend
- Normalization factor in A is **ignored**
- $\mathbf{\Phi}$-representation not accessible ⇒ **use "black-box"**

Tested configurations



Stages & Variances

A RankBoost features **B** projection onto unlabeled data **C** random projection **D** classifier learning

kernel	steps	I_{sample}	A→D	A→B→D	A→C→D	A→B→C→D
stump	2270	1000	0.99539	0.99280	0.99517	0.99232
stump	2270	5000	0.99539	0.99296	0.99517	0.99166
grid	2000	1000	0.99076	0.99518	0.99416	0.99378
grid	2000	5000	0.99076	0.99517	0.99416	**0.99546**

kernel	steps	I_{sample}	A→B→D	A→B→C→D
stump	2270	100	0.9916±0.0015	0.9935±0.0017
stump	2270	1000	0.9926±0.0003	0.9925±0.0004
stump	2270	5000	0.9927±0.0001	0.9925±0.0004
grid	2000	100	0.9950±0.0005	0.9951±0.0003
grid	2000	1000	0.9952±0.0004	0.9950±0.0003
grid	2000	5000	0.9950±0.0004	0.9926±0.0007
neural 20-20		1000	0.9956±0.00003	0.9893±0.0007
neural 100-10		1000	0.9955±0.00004	0.9886±0.0021

Less samples ⇒ more variance Unexplained: Stumps: **A→C→D** better than **A→B→D**. Grids: bad **A→D**

SSFL Challenge on web data

- number of features – 10^6
  - 80% of those are binary
  - sparse (∼ 115 active simultaneously)
- required output dimension – **100**
- **reduced data must permit learning**
  - fixed task: **binary linear** classif.
  - performance measure – **AUC**
- full results on poster of D. Sculley

Baseline results

baseline	AUC
100 k-means	0.9831
1000 k-means + random projection	0.9846
1000 k-means + neural dim. red.	0.9868
RankBoost, stumps, 5000 steps	0.9961
RankBoost, stumps, 2270 steps	0.9953
RankBoost, grids, 2000 steps	0.9949
RankBoost, grids, 1150 steps	0.9949
sparse logistic regression	0.9958
sparse log. reg. + 100 k-means	0.9963
sparse log. reg. + 200 k-means	**0.9963**
sparse log. regression + 800 k-means	0.9962
log. regression with neural network	0.9937
log. reg. + graph smoothing	0.9949
log. reg. with NN on relabeled data	0.9847

Method's results

kernel	T	I_{sample}	orth.	AUC
stump	5000	1000	no	0.9803
stump	5000	1000	yes	0.9927
stump	2270	1000	no	0.9923
stump	5000	5000	no	0.9932
stump	5000	5000	yes	0.9920
stump	2270	5000	no	0.9917
grid	2000	1000	no	0.9951
grid	2000	1000	yes	0.9938
grid	**2000**	**5000**	**no**	**0.9955**
grid	2000	5000	yes	0.9951
neural 5		1000	no	0.9895
neural 20-20		1000	no	0.9887
neural 100-10		1000	no	0.9872
neural 5		5000	no	0.9901
neural 20-20		5000	no	0.9928
neural 100-10		5000	no	0.9922