Local Deep Kernel Learning for Efficient Non-linear SVM Prediction

Suraj Jain
MSR India

Vinayak Agrawal
IIT Delhi

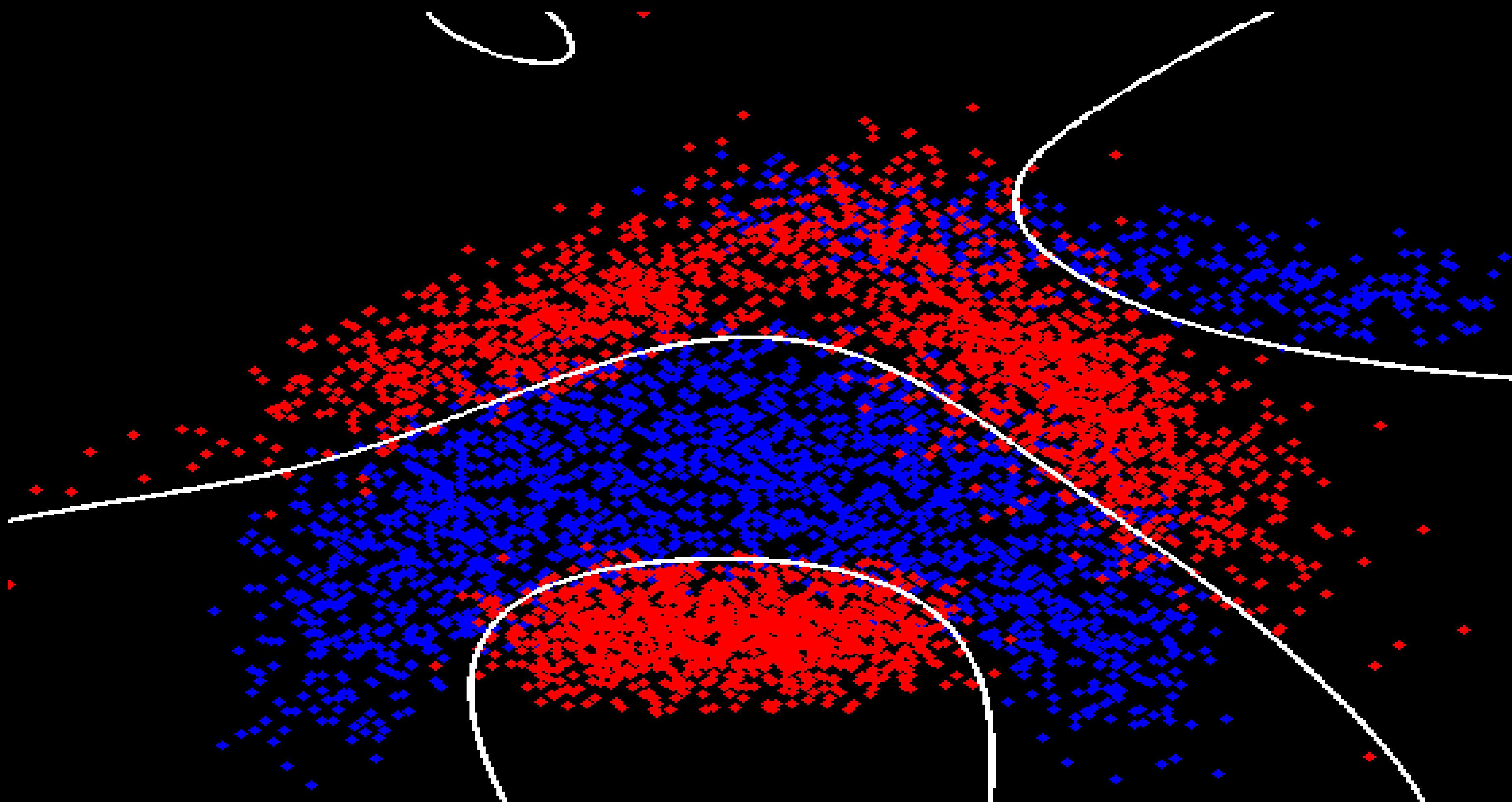
Cijo Jose
EPFL

Prasoon Goyal
IIT Delhi

Manik Varma
MSR India

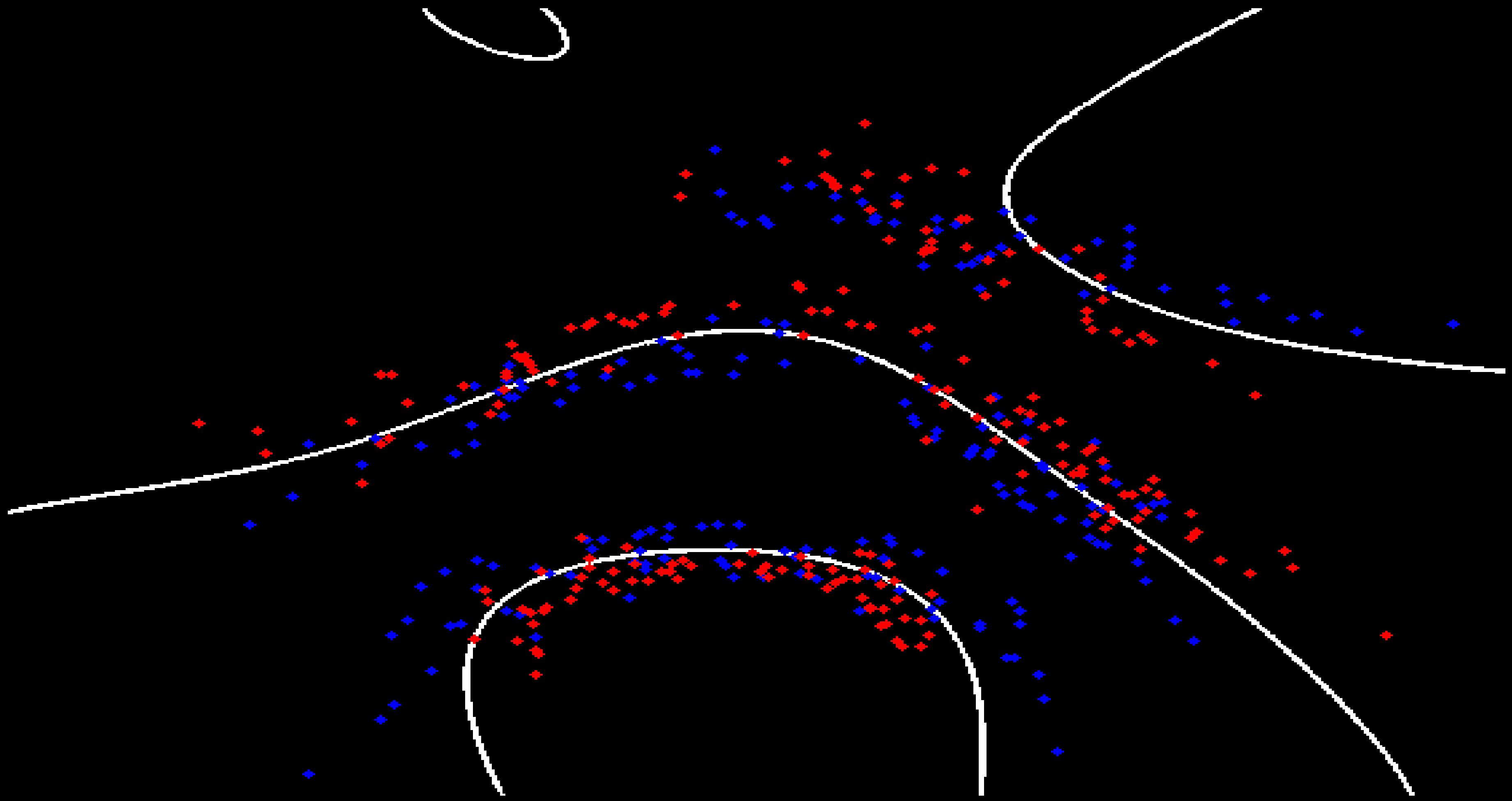
Non-linear SVM Prediction

- Non-linear SVM prediction can be accurate



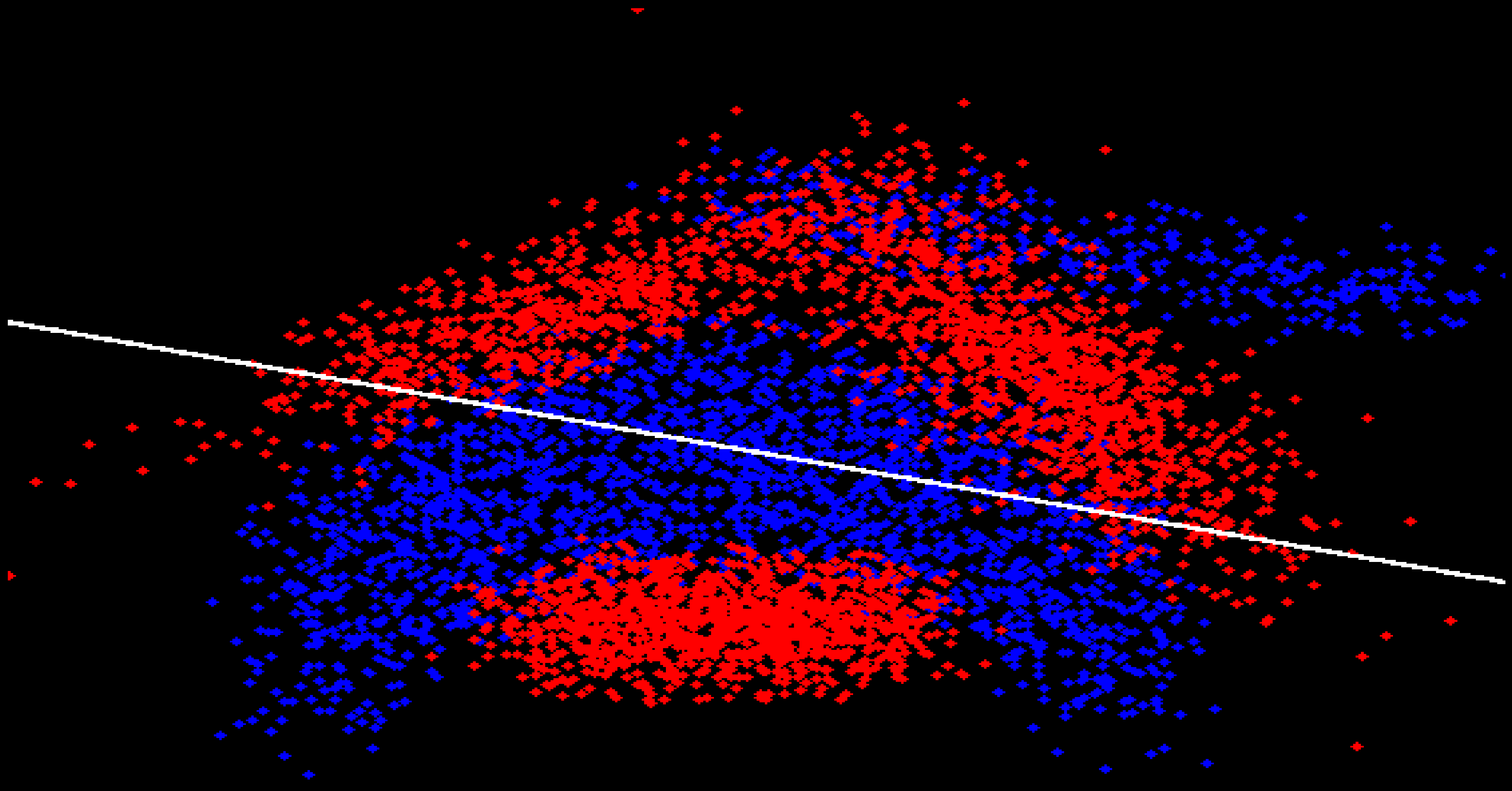
Non-linear SVM Prediction

- Cost of prediction = $O(DN)$



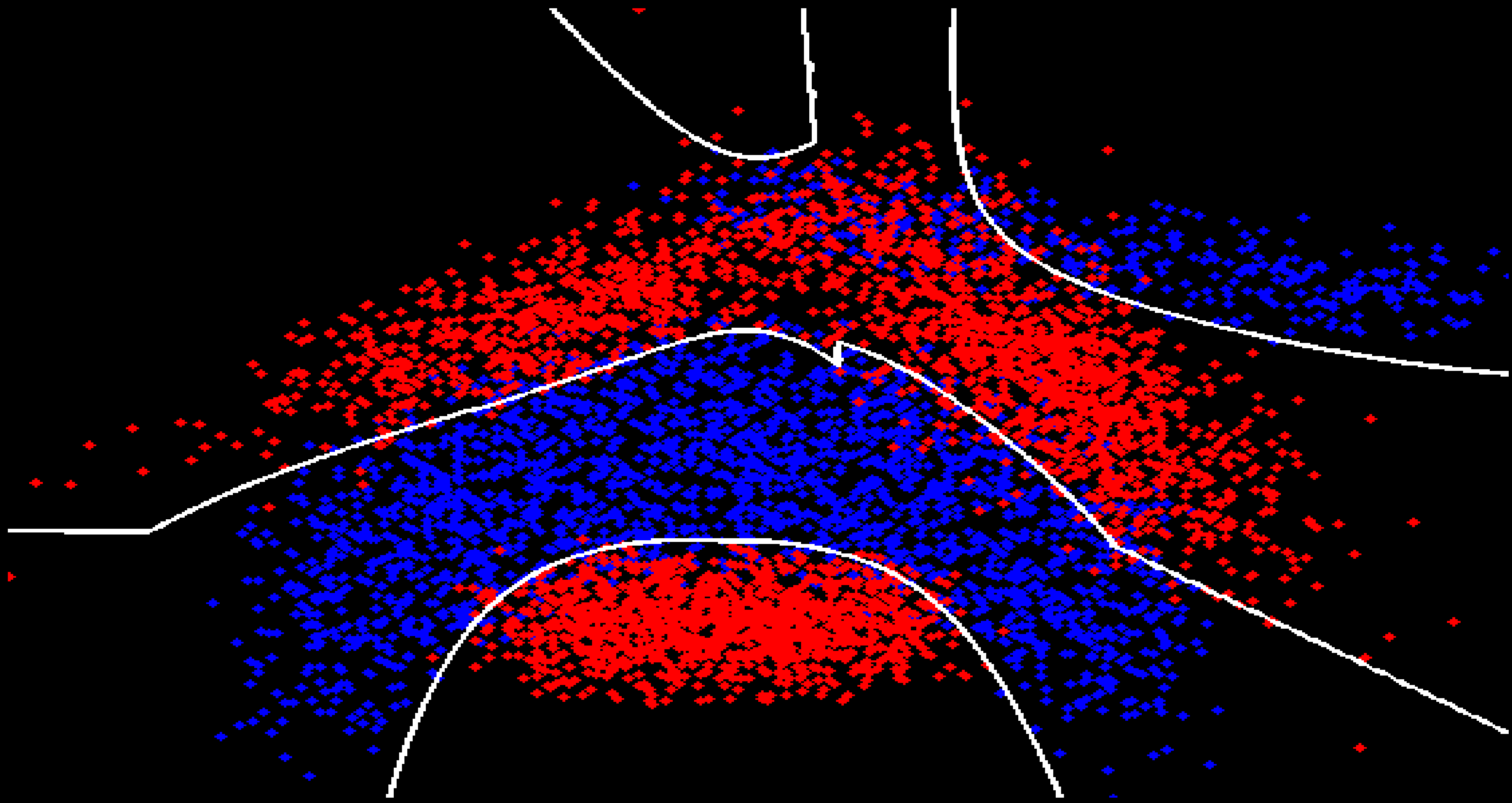
Linear SVM Prediction

- Cost of prediction = $O(D)$



Local Deep Kernel Learning Prediction

- Cost of prediction = $O(D \log N)$



Our Contributions

- Model
  - We learn a locally linear composite kernel
  - We learn a tree structured kernel for logarithmic prediction
  - Each node in the tree generates real-valued feature
- Optimization
  - We learn all the tree parameters jointly
  - We optimize efficiently using primal SGD
  - We scale to problems with millions of data points
- Prediction speedup and accuracy
  - 13,000 times faster than the RBF-SVM in some cases
  - Significantly higher accuracy over the state-of-the-art

Localized Multiple Kernel Learning

- Prediction function

$$y(\mathbf{x}) = \mathrm{sign}\left(\sum_k p(\mathbf{w}_k|\mathbf{x})\mathbf{w}_k^t \boldsymbol{\phi}_k(\mathbf{x}) + b\right)$$

- Kernel function

$$K_p(\mathbf{x}_i, \mathbf{x}_j) = \sum_k p(\mathbf{w}_k|\mathbf{x}_i) K_k(\mathbf{x}_i, \mathbf{x}_j)\, p(\mathbf{w}_k|\mathbf{x}_j)$$

- Dual optimization

$$\begin{aligned} \mathrm{Min}_p \mathrm{Max}_\alpha \quad & \mathbf{1}^t\boldsymbol{\alpha} - \tfrac{1}{2}\boldsymbol{\alpha}^t \mathbf{Y} K_p \mathbf{Y}\boldsymbol{\alpha} \\ \text{s. t.} \quad & \mathbf{1}^t\mathbf{Y}\boldsymbol{\alpha} = 0 \\ & \mathbf{0} \le \boldsymbol{\alpha} \le \mathbf{C} \end{aligned}$$

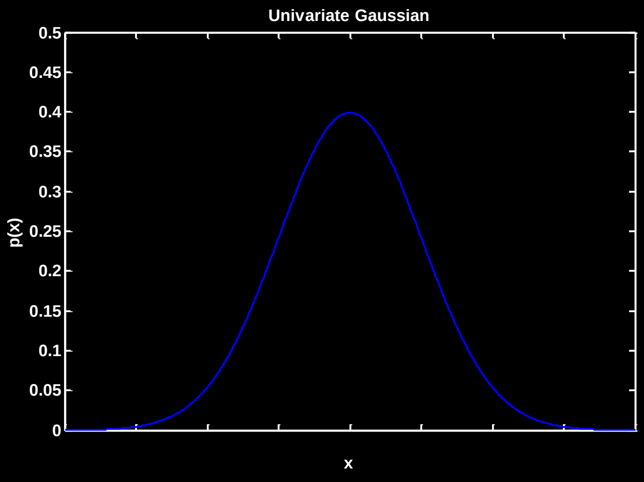
Gonen & Alpaydin
ICML 2008

Non-linear SVM Prediction

- Non-linear SVM prediction

$$y(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i)\right)$$

- RBF kernel: $K(\mathbf{x}, \mathbf{x}_i) \; = e^{-\gamma ||\mathbf{x}-\mathbf{x}_i||_2^2}$



- Cost of prediction = $O(DN)$

Composite Kernel Prediction

- Non-linear SVM prediction

$$y(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i)\right)$$

- LDKL's composite kernel

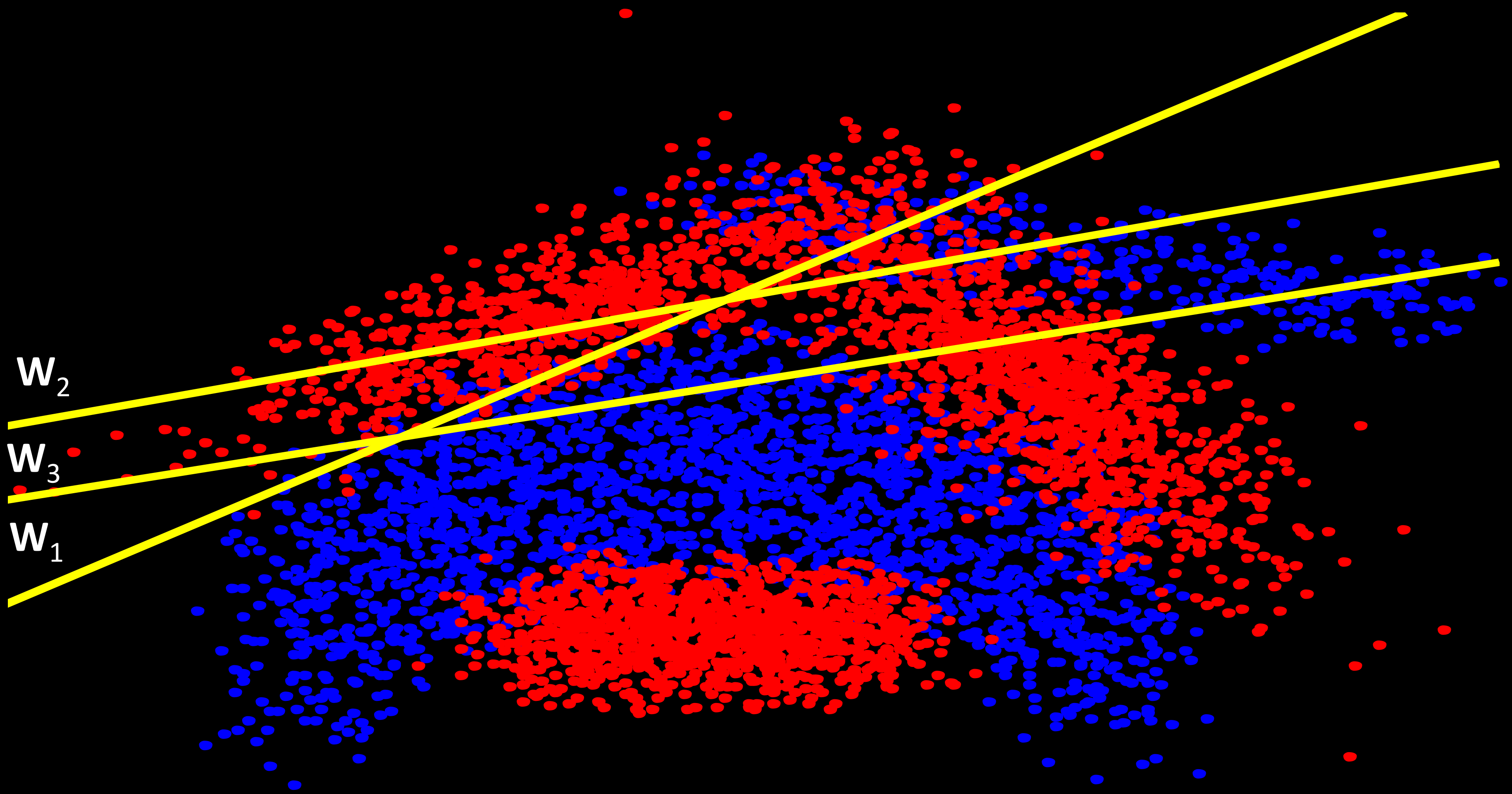
$$\begin{aligned} K(\mathbf{x}, \mathbf{x}_i) &= K_N(\mathbf{x}, \mathbf{x}_i)\, K_L(\mathbf{x}, \mathbf{x}_i) \\ &= \boldsymbol{\phi}^t(\mathbf{x})\boldsymbol{\phi}(\mathbf{x}_i)\mathbf{x}^t\mathbf{x}_i \end{aligned}$$

- LDKL's prediction function

$$y(\mathbf{x}) = \text{sign}\left(\sum_{k=1}^{M} \boldsymbol{\phi}_k(\mathbf{x})\mathbf{w}_k^t\mathbf{x}\right)$$

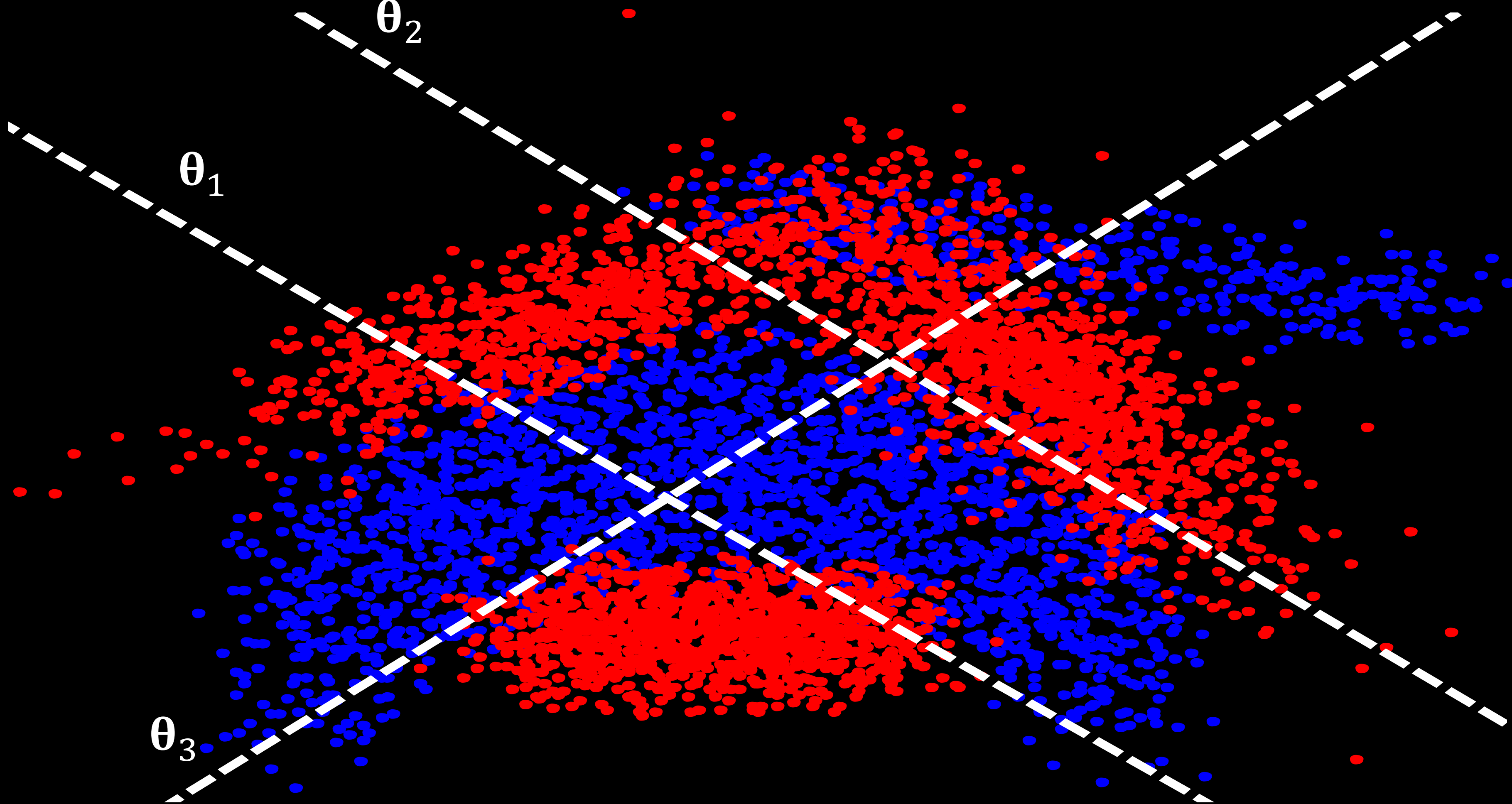
Composite Kernel Prediction

- LDKL's prediction function: $y(\mathbf{x}) = \mathrm{sign}(\sum_{k=1}^{M} \boldsymbol{\phi}_k(\mathbf{x})\mathbf{w}_k^t\mathbf{x})$



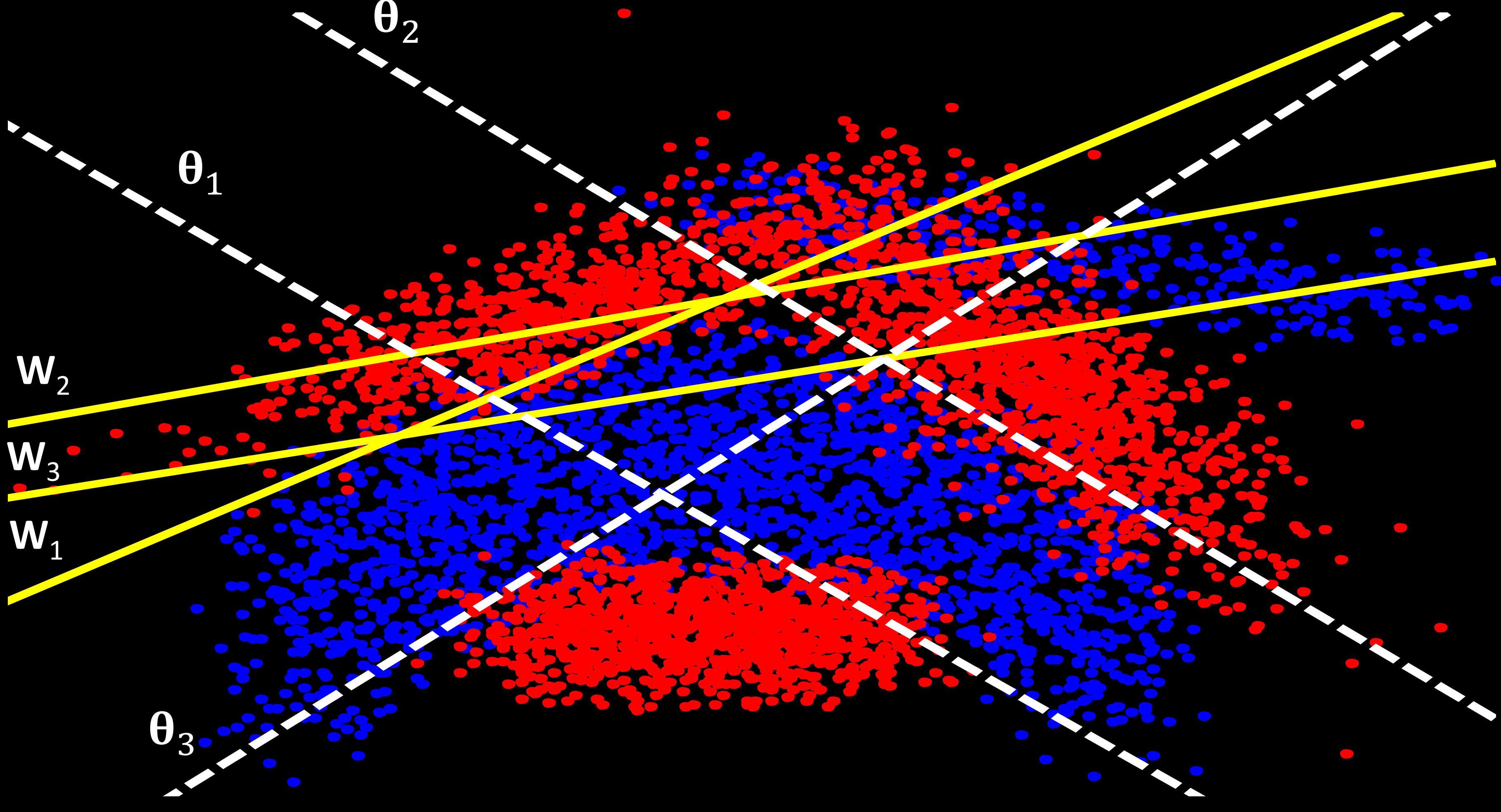
Composite Kernel Prediction

- LDKL's prediction function: $y(\mathbf{x}) = \text{sign}(\sum_{k=1}^{M} \boldsymbol{\phi}_k(\mathbf{x})\mathbf{w}_k^t\mathbf{x})$



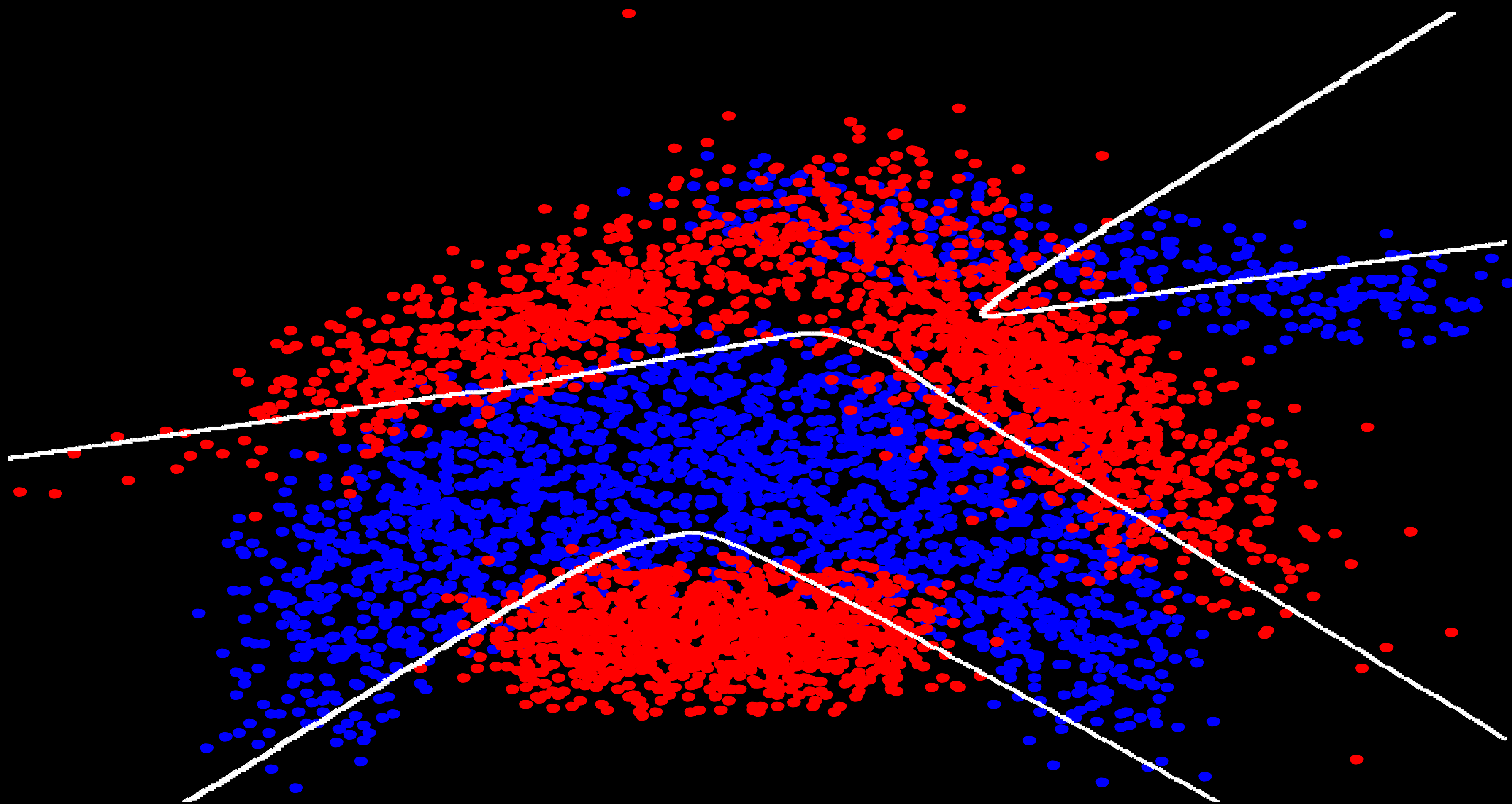
Composite Kernel Prediction

- LDKL's prediction function: $y(\mathbf{x}) = \text{sign}(\sum_{k=1}^{M} \boldsymbol{\phi}_k(\mathbf{x})\mathbf{w}_k^t\mathbf{x})$



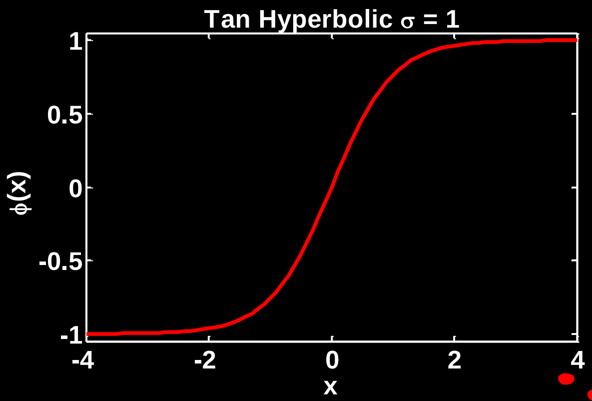
Composite Kernel Prediction

- LDKL's prediction function: $y(\mathbf{x}) = \operatorname{sign}(\sum_{k=1}^{M} \boldsymbol{\phi}_k(\mathbf{x})\mathbf{w}_k^t\mathbf{x})$

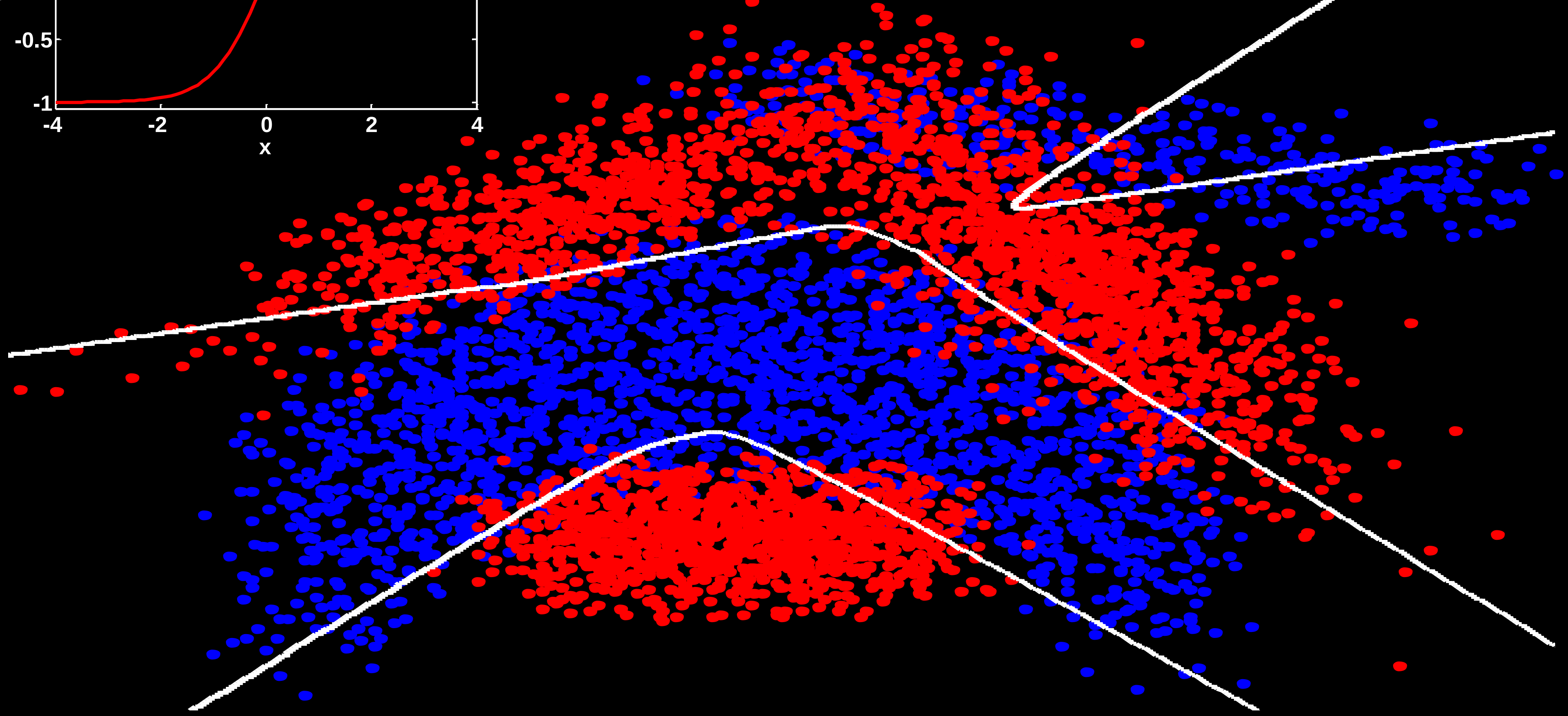


Composite Kernel Prediction

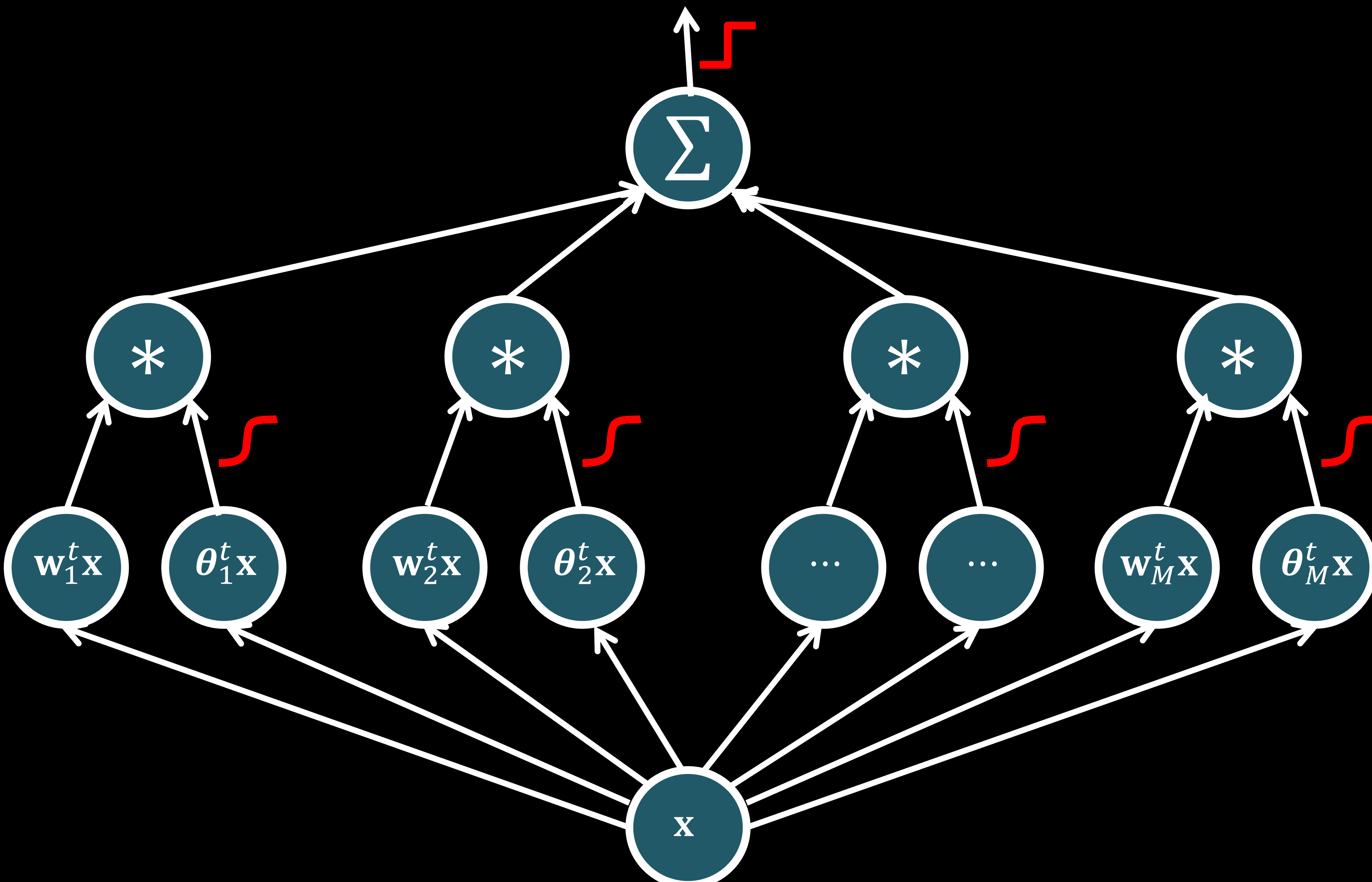
- LDKL's prediction function: $y(\mathbf{x}) = \text{sign}(\sum_{k=1}^{M} \phi_k(\mathbf{x})\mathbf{w}_k^t\mathbf{x})$



$$\boldsymbol{\phi}_k(\mathbf{x}) = \tanh(\sigma\boldsymbol{\theta}_k^t \mathrm{x})$$



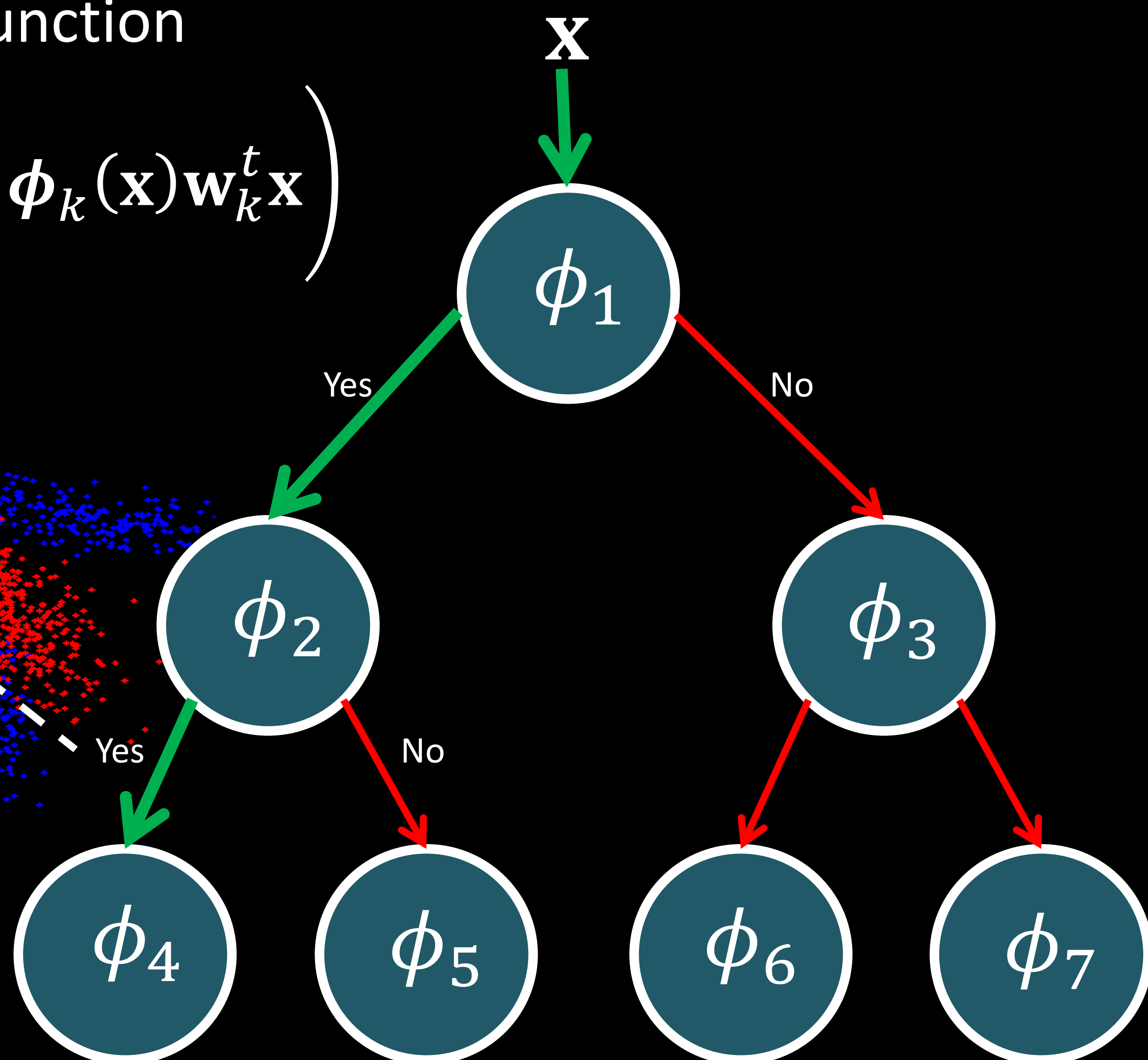
A Shallow Architecture

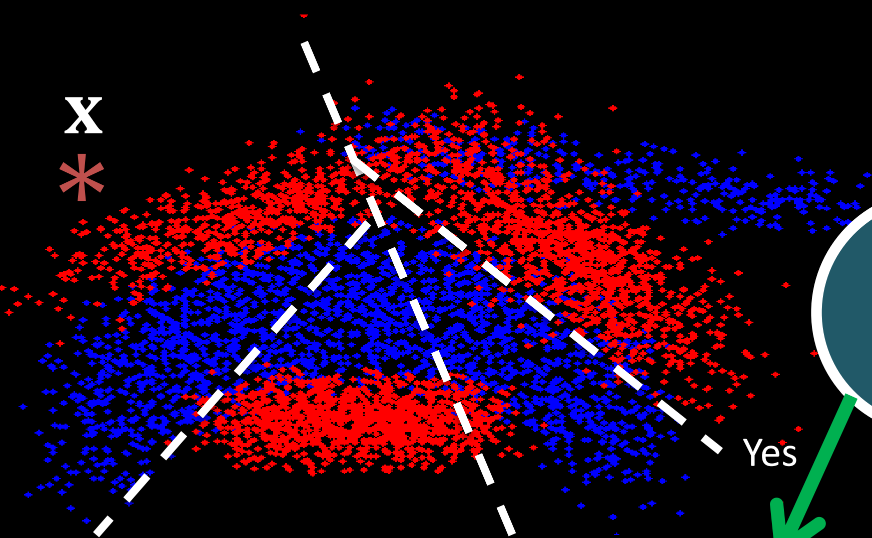


Learning Tree Structured Features

- LDKL's prediction function

$$y(\mathbf{x}) = \text{sign}\left(\sum_{k=1}^{M} \boldsymbol{\phi}_k(\mathbf{x})\mathbf{w}_k^t\mathbf{x}\right)$$

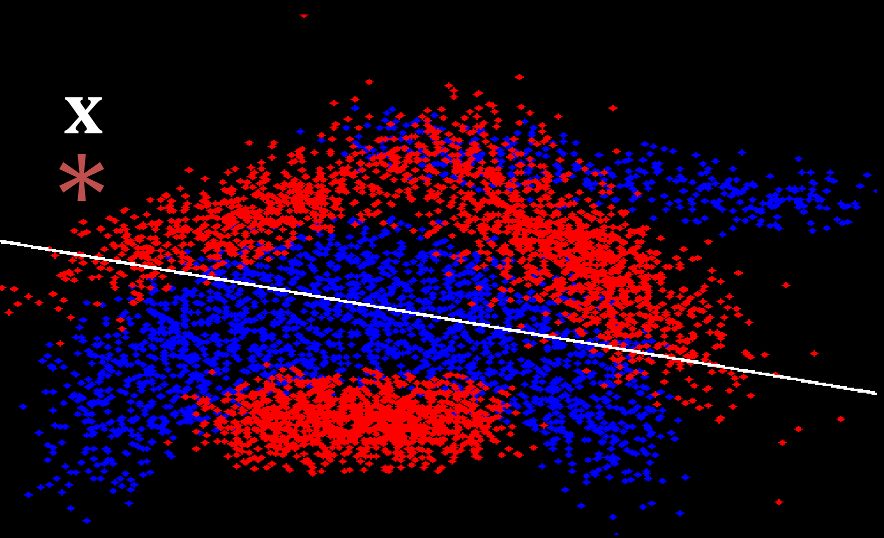


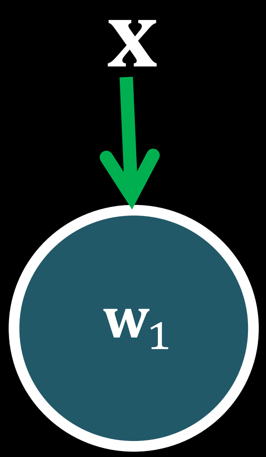


Learning Tree Structured Features

- LDKL's prediction function

$$y(\mathbf{x}) = \text{sign}(\mathbf{w}_1^t \mathbf{x})$$

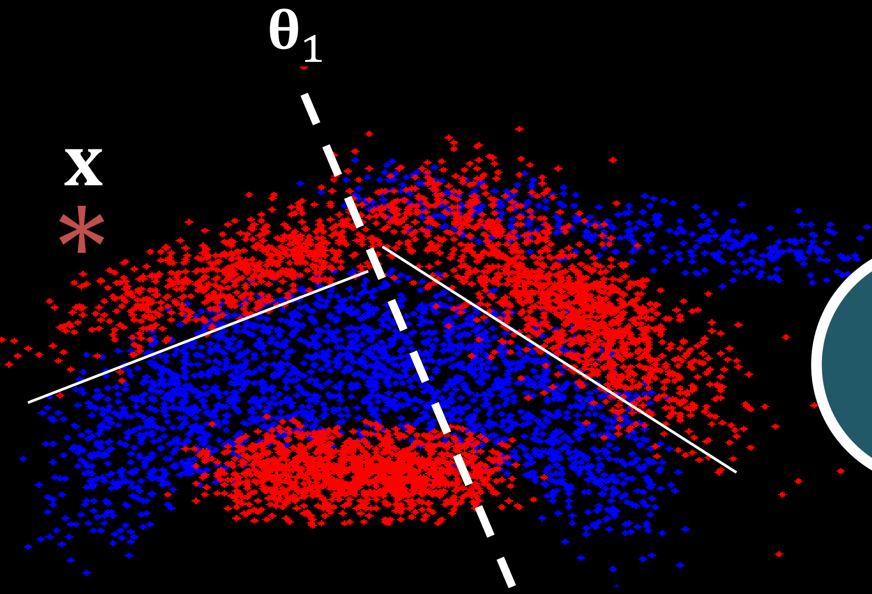


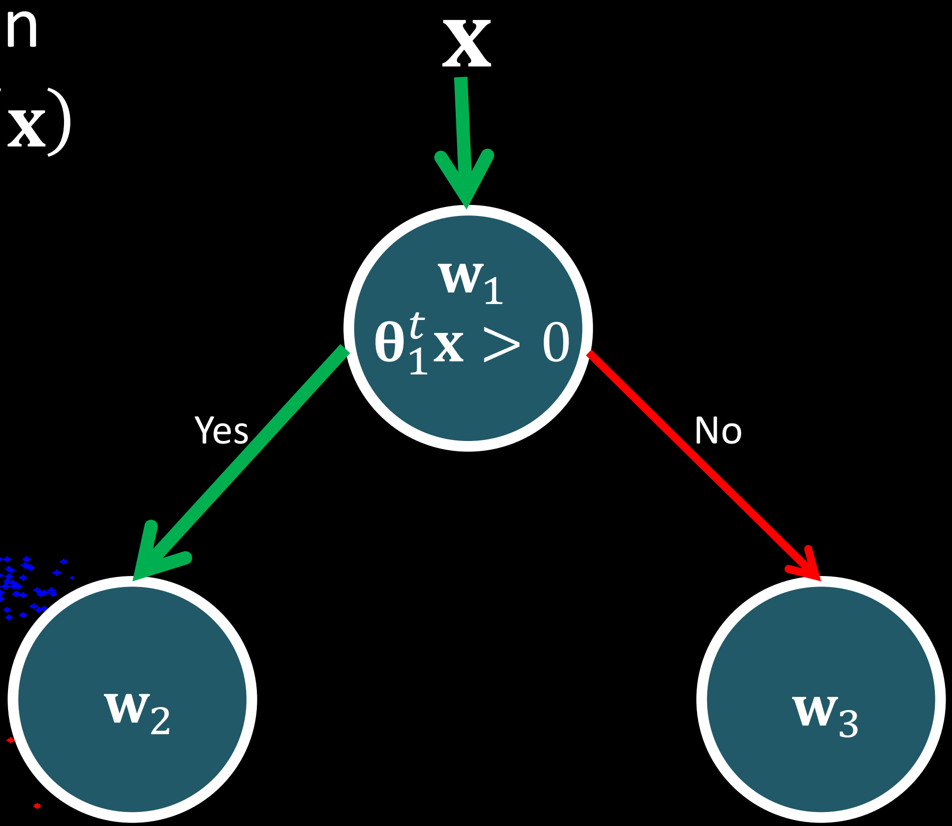


Learning Tree Structured Features

- LDKL's prediction function

$$y(\mathbf{x}) = \text{sign}(\mathbf{w}_1^t\mathbf{x} + \mathbf{w}_2^t\mathbf{x})$$

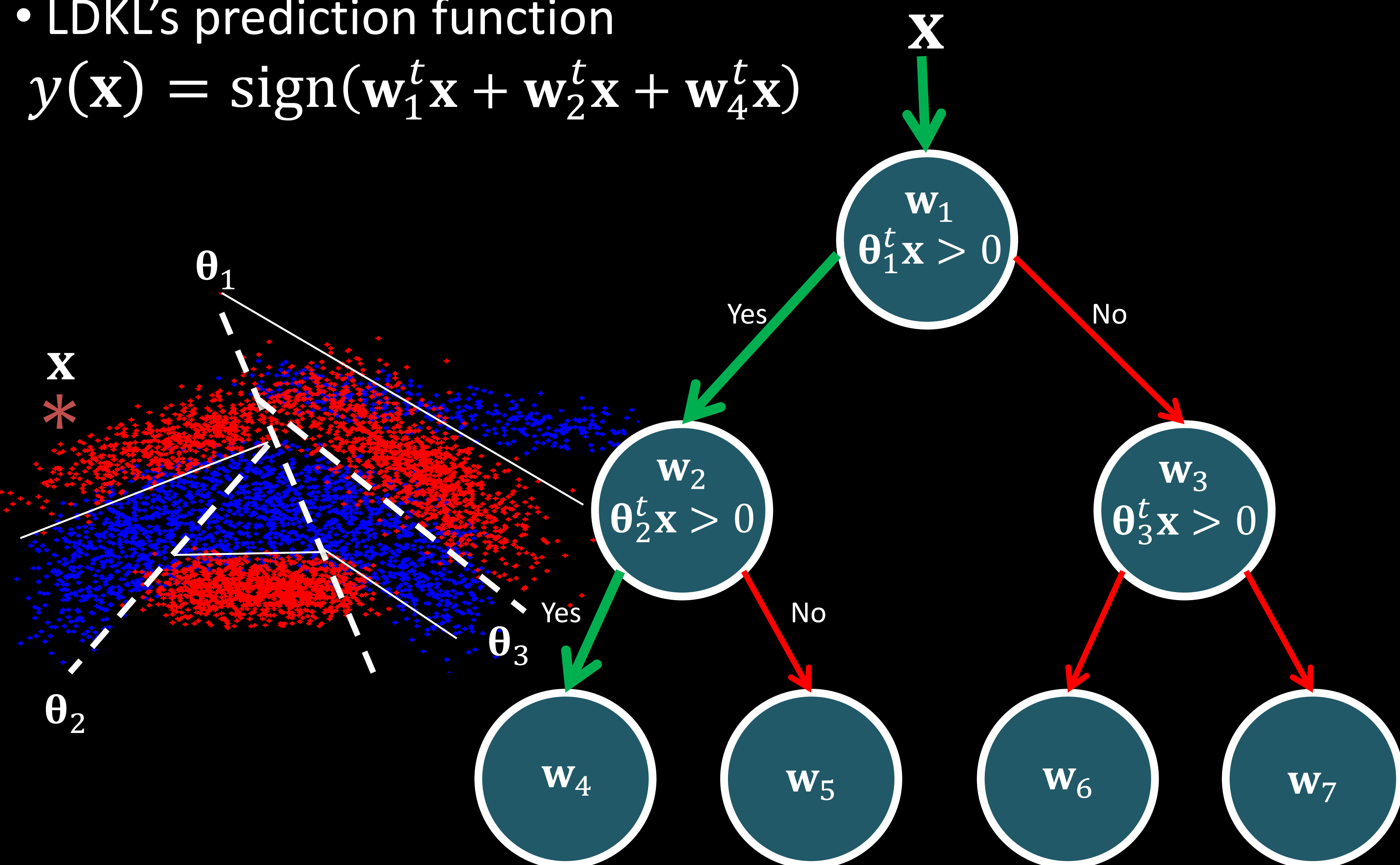




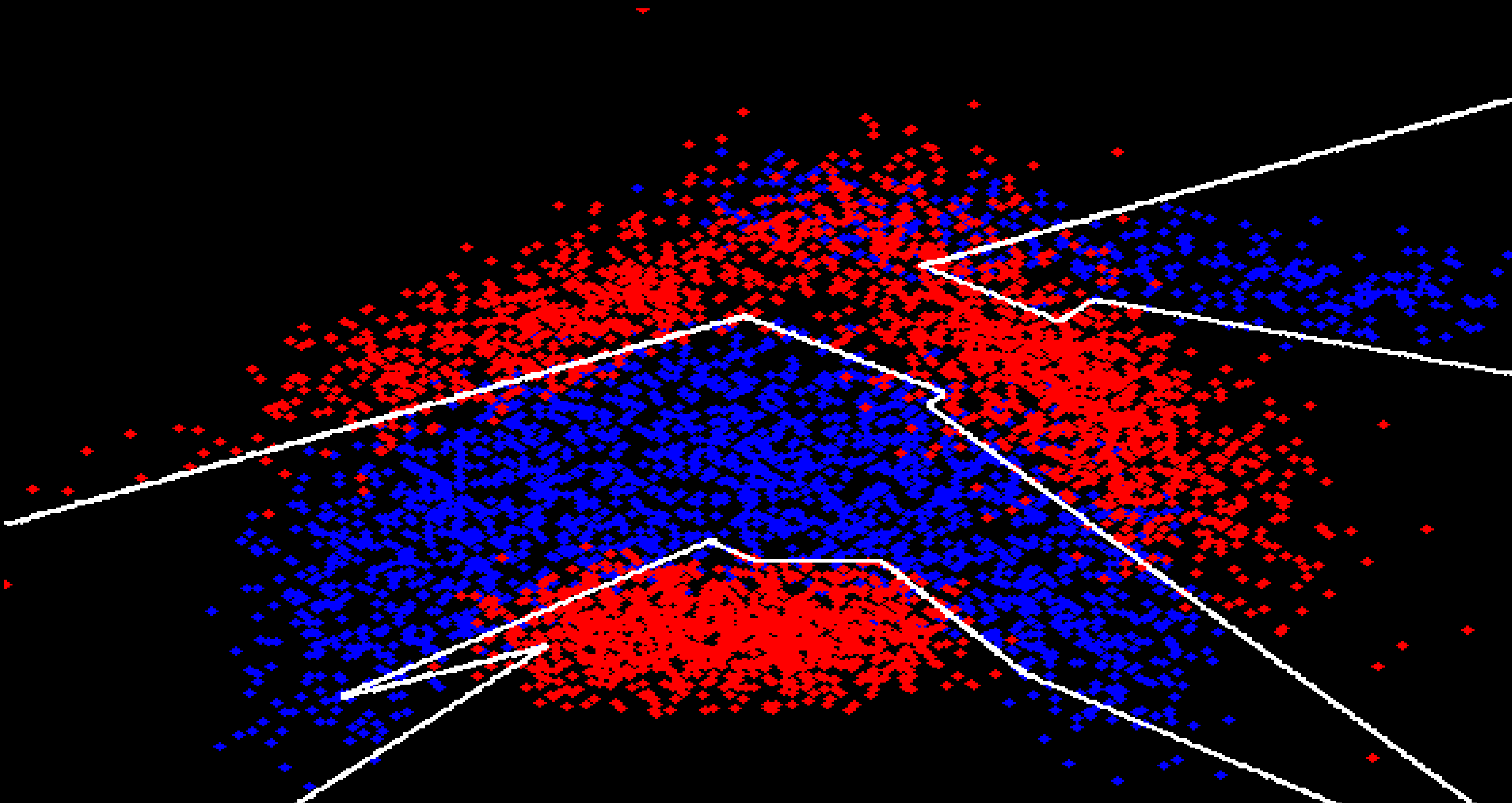
Learning Tree Structured Features

- LDKL's prediction function

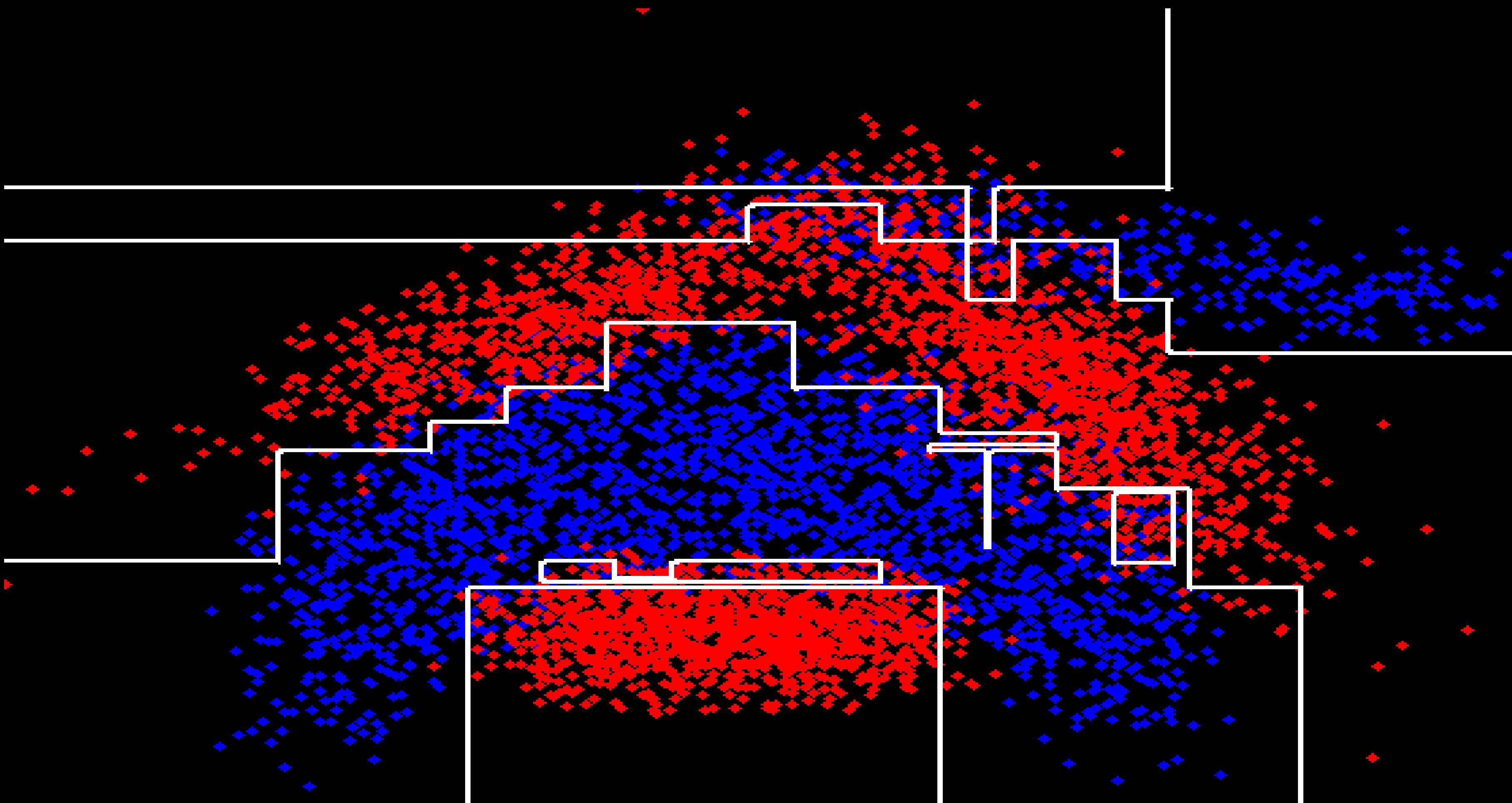
$$y(\mathbf{x}) = \text{sign}(\mathbf{w}_1^t\mathbf{x} + \mathbf{w}_2^t\mathbf{x} + \mathbf{w}_4^t\mathbf{x})$$



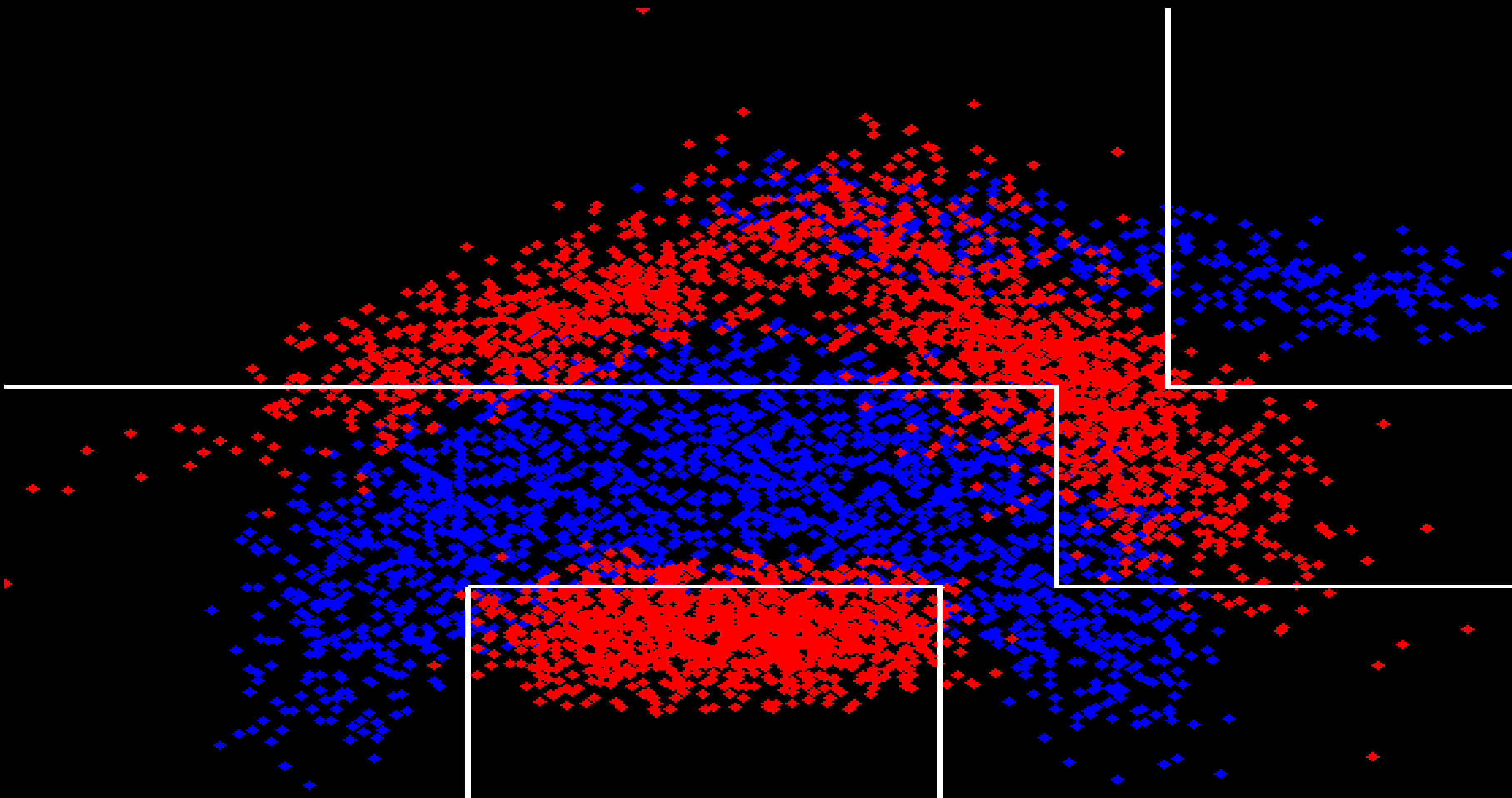
Comparison to a Perceptron Tree



Comparison to a Decision Tree



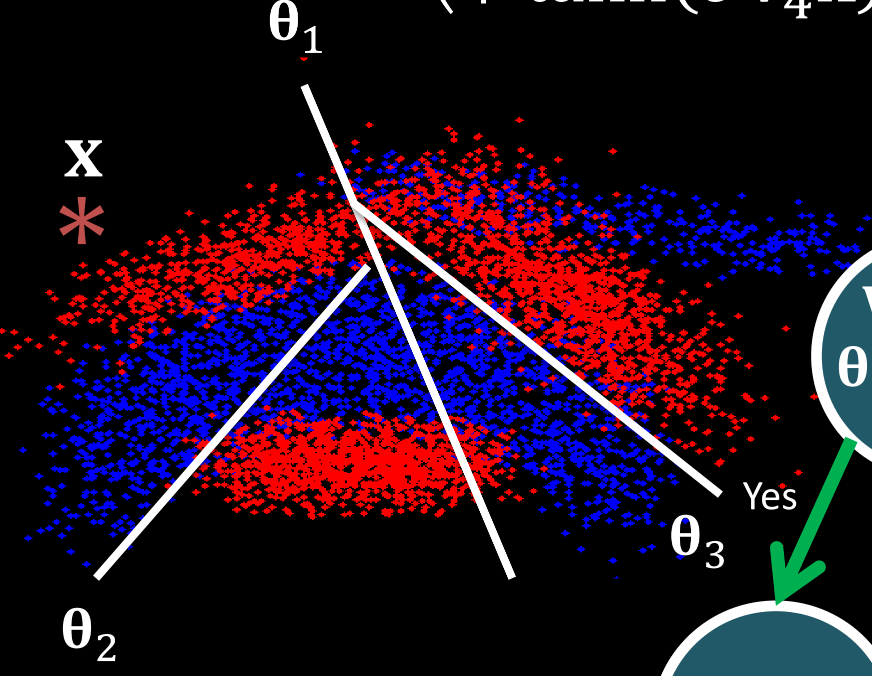
Comparison to a Decision Tree

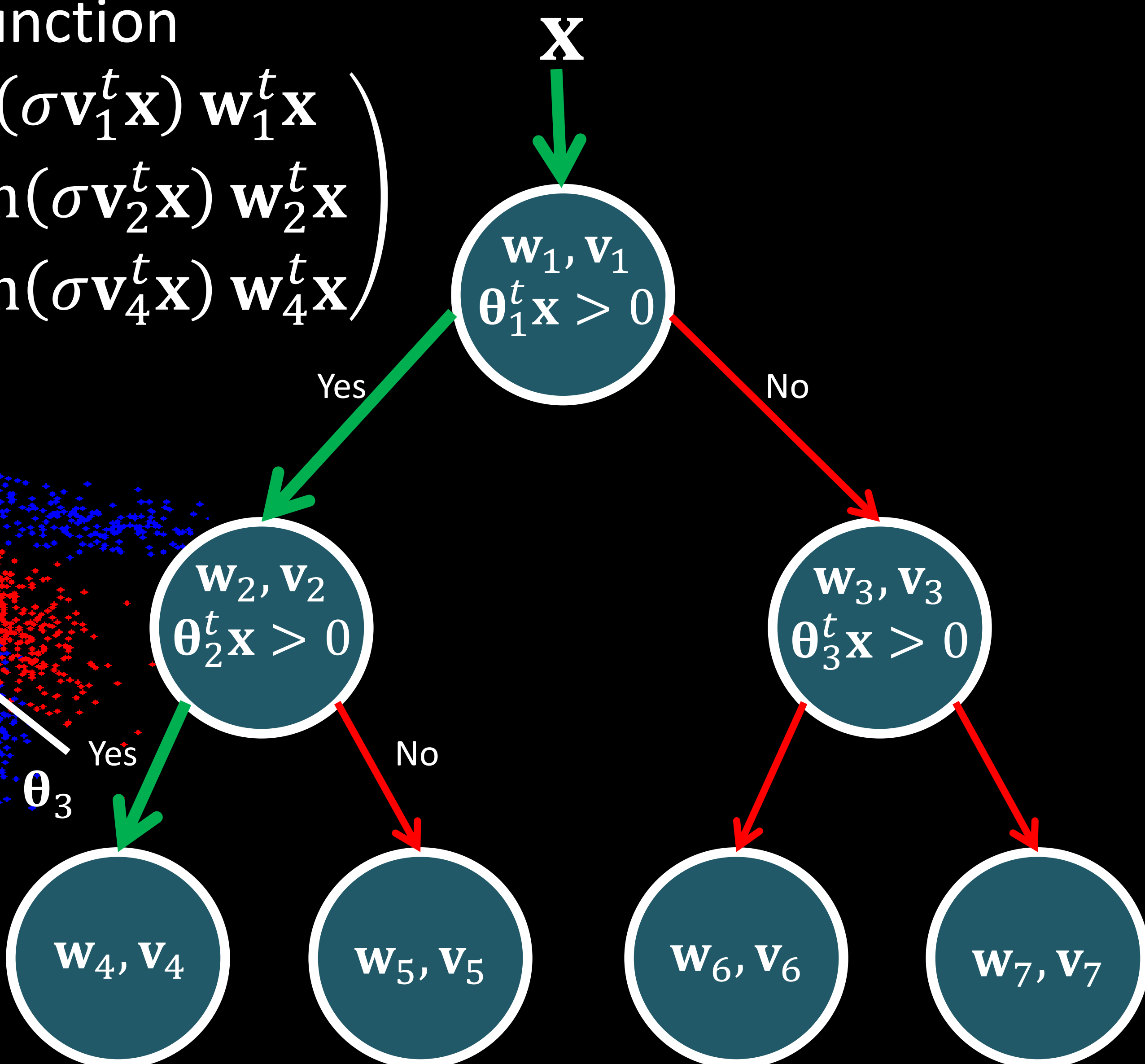


Learning Tree Structured Features

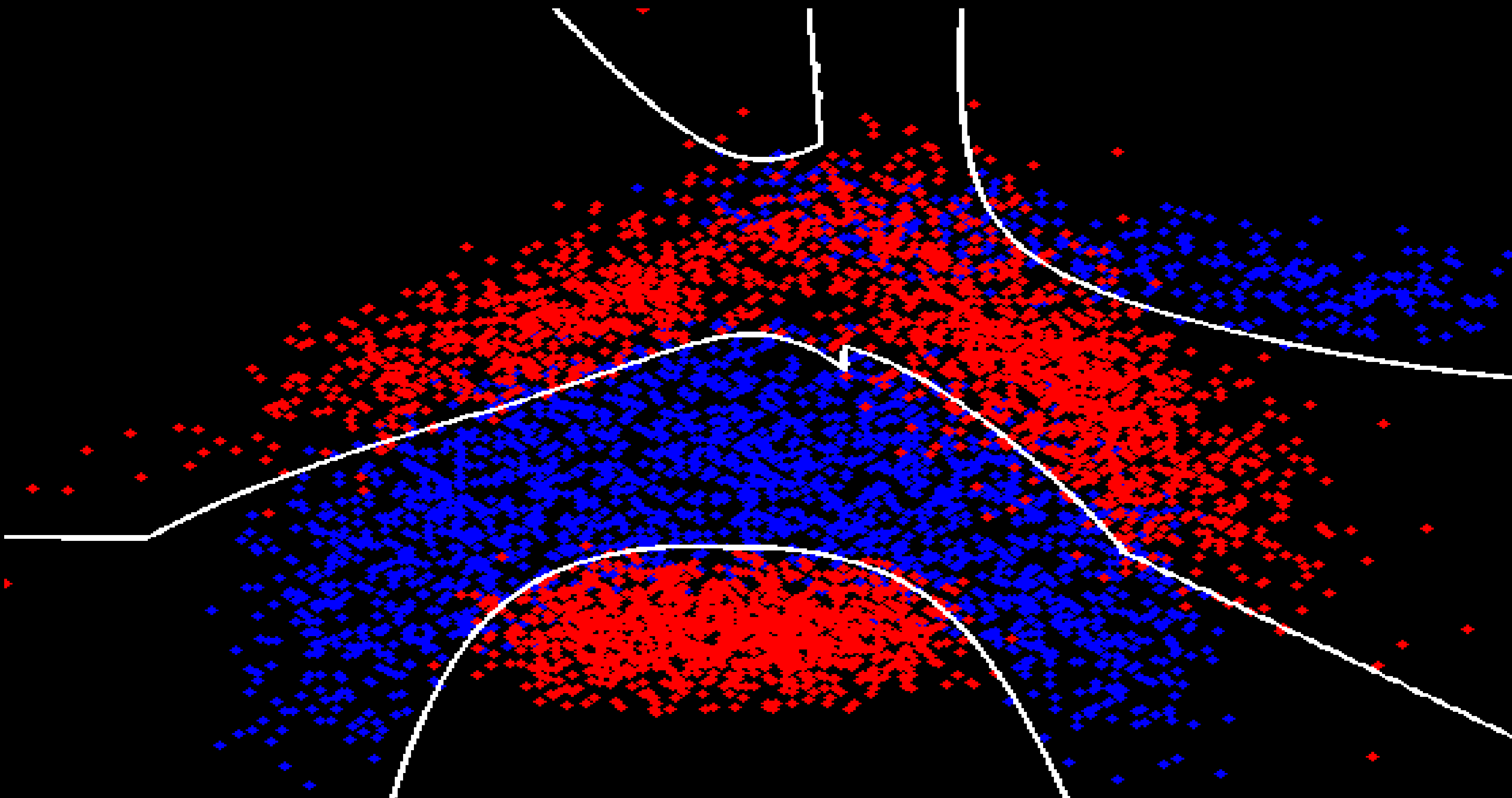
- LDKL's prediction function

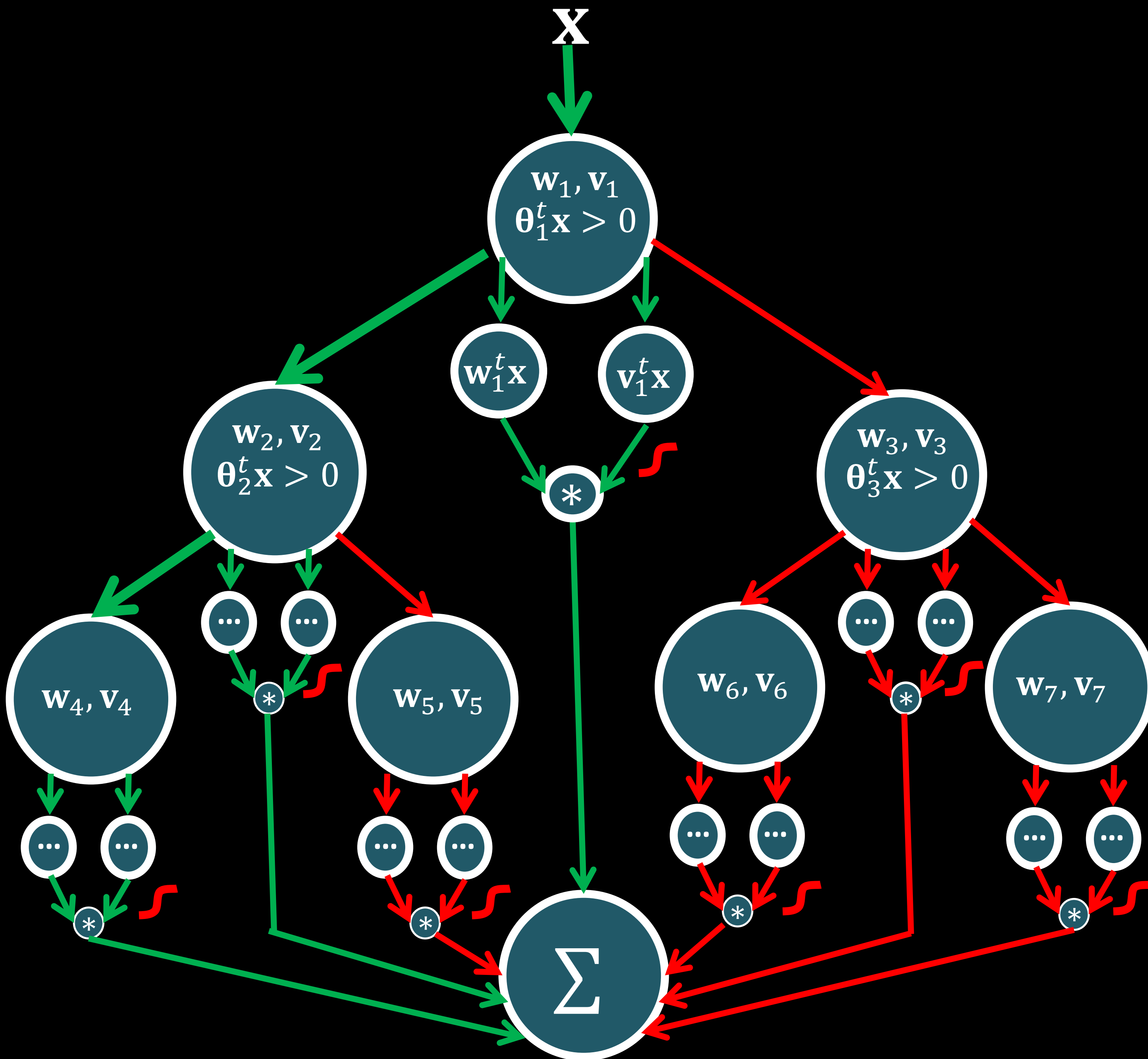
$$y(\mathbf{x}) = \text{sign}\begin{pmatrix} \tanh(\sigma \mathbf{v}_1^t \mathbf{x})\, \mathbf{w}_1^t \mathbf{x} \\ + \tanh(\sigma \mathbf{v}_2^t \mathbf{x})\, \mathbf{w}_2^t \mathbf{x} \\ + \tanh(\sigma \mathbf{v}_4^t \mathbf{x})\, \mathbf{w}_4^t \mathbf{x} \end{pmatrix}$$





LDKL's Decision Boundaries



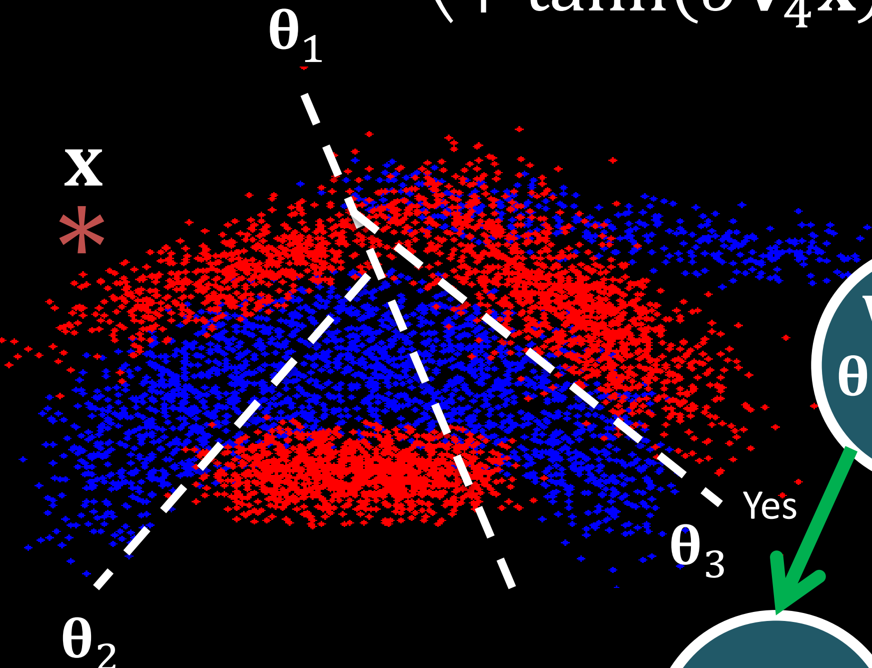


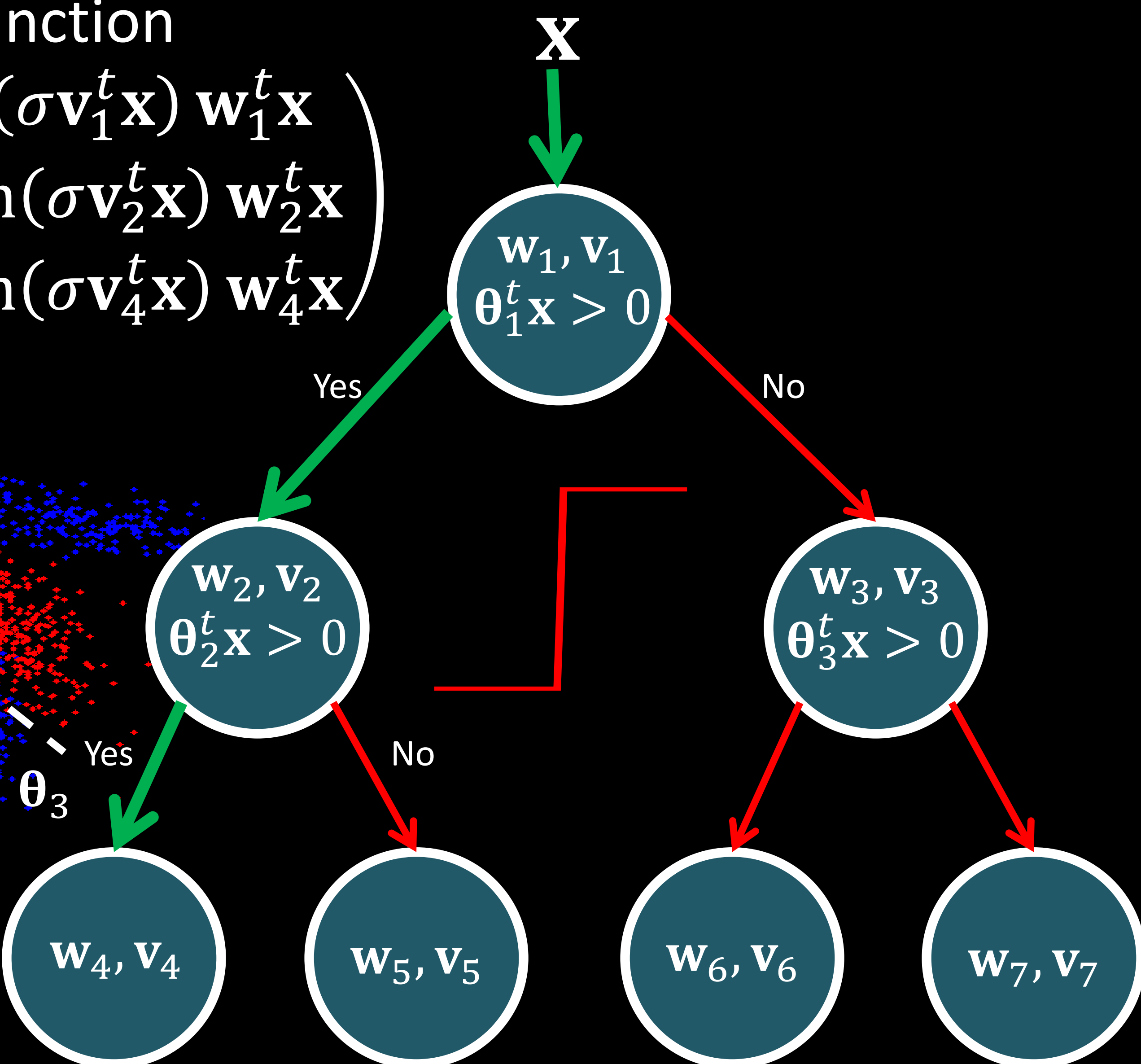
x
w1, v1
θ1t x > 0
w1t x
v1t x
*
w2, v2
θ2t x > 0
w3, v3
θ3t x > 0
w4, v4
w5, v5
w6, v6
w7, v7
Σ

Training LDKL

- LDKL's prediction function

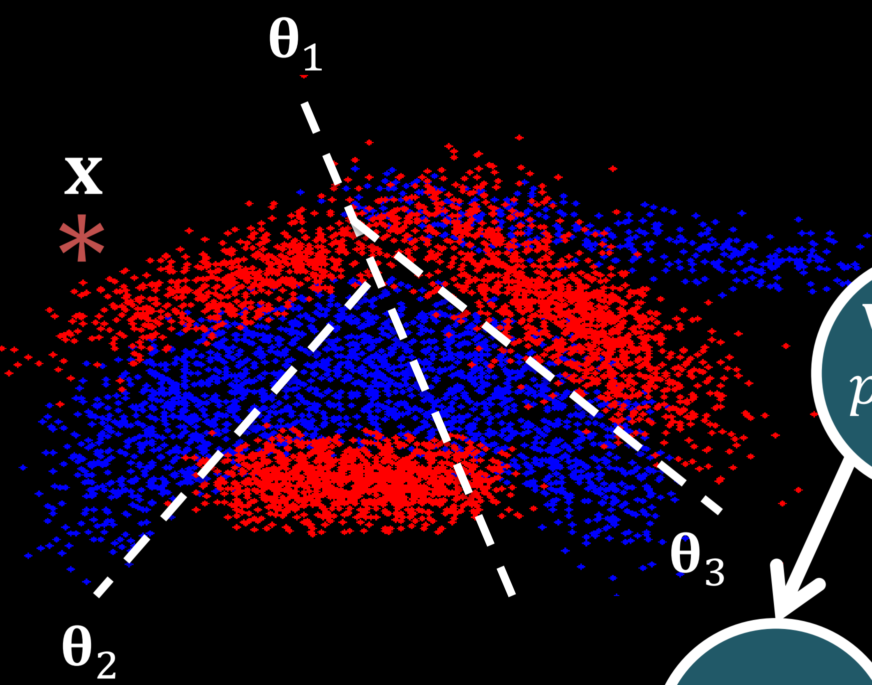
$$y(\mathbf{x}) = \operatorname{sign}\left(\begin{array}{c} \tanh(\sigma\mathbf{v}_1^t\mathbf{x})\,\mathbf{w}_1^t\mathbf{x} \\ +\tanh(\sigma\mathbf{v}_2^t\mathbf{x})\,\mathbf{w}_2^t\mathbf{x} \\ +\tanh(\sigma\mathbf{v}_4^t\mathbf{x})\,\mathbf{w}_4^t\mathbf{x} \end{array}\right)$$

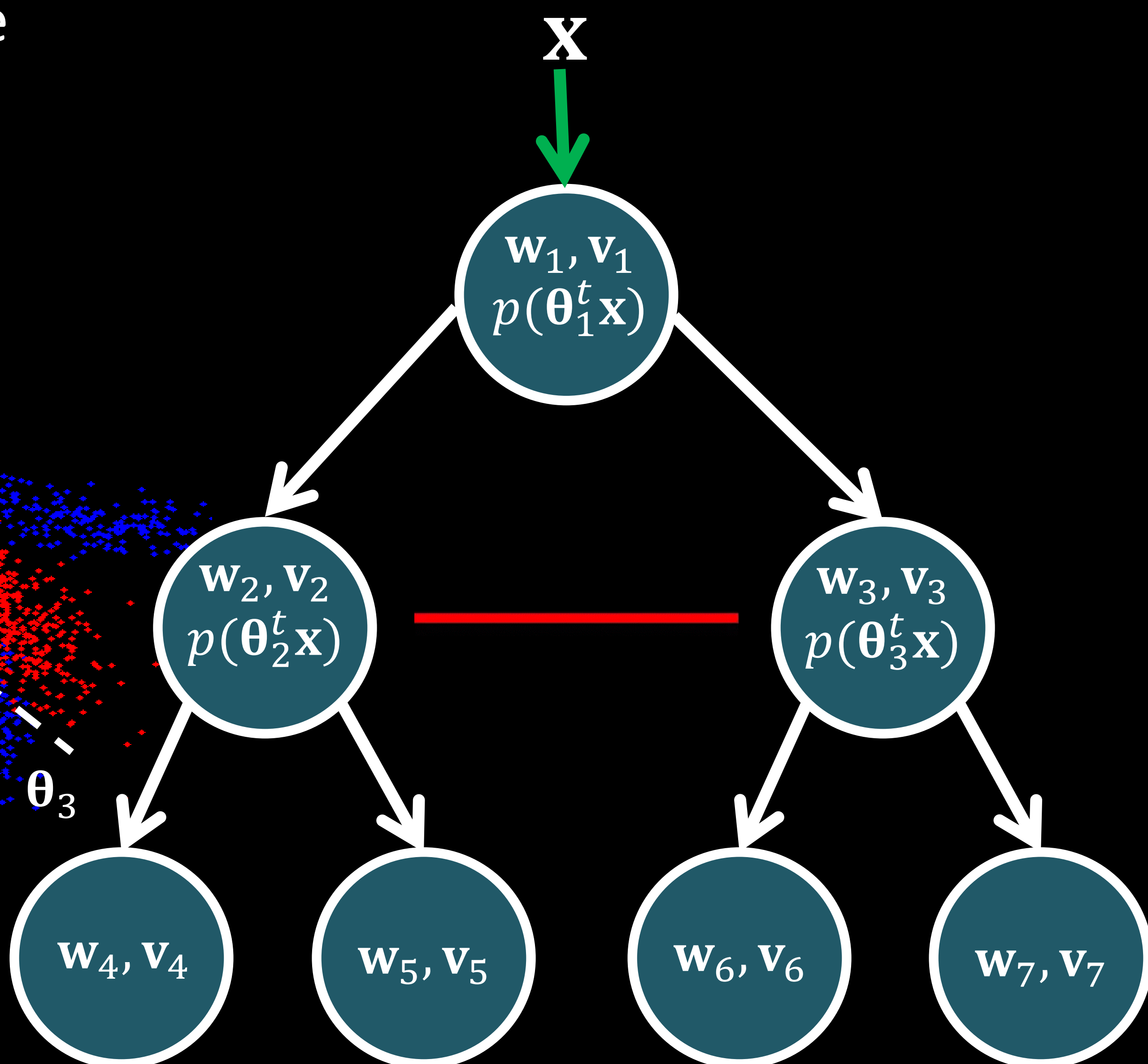




Training LDKL

- Smoothing the tree





Training LDKL

- Primal optimization via stochastic sub-gradient descent

$$\mathrm{Min}_{\mathbf{V},\mathbf{W},\Theta}\ P = \frac{\lambda_{\mathrm{W}}}{2}\mathrm{Tr}(\mathbf{W}^t\mathbf{W}) + \frac{\lambda_{\mathrm{V}}}{2}\mathrm{Tr}(\mathbf{V}^t\mathbf{V}) + \frac{\lambda_{\Theta}}{2}\mathrm{Tr}(\mathbf{\Theta}^t\mathbf{\Theta}) + \sum_i \max(0, 1 - y_i \boldsymbol{\phi}_L^{\boldsymbol{t}}(\mathbf{x}_i)\mathbf{W}^{\boldsymbol{t}}\mathbf{x}_i)$$

- Sub-gradients

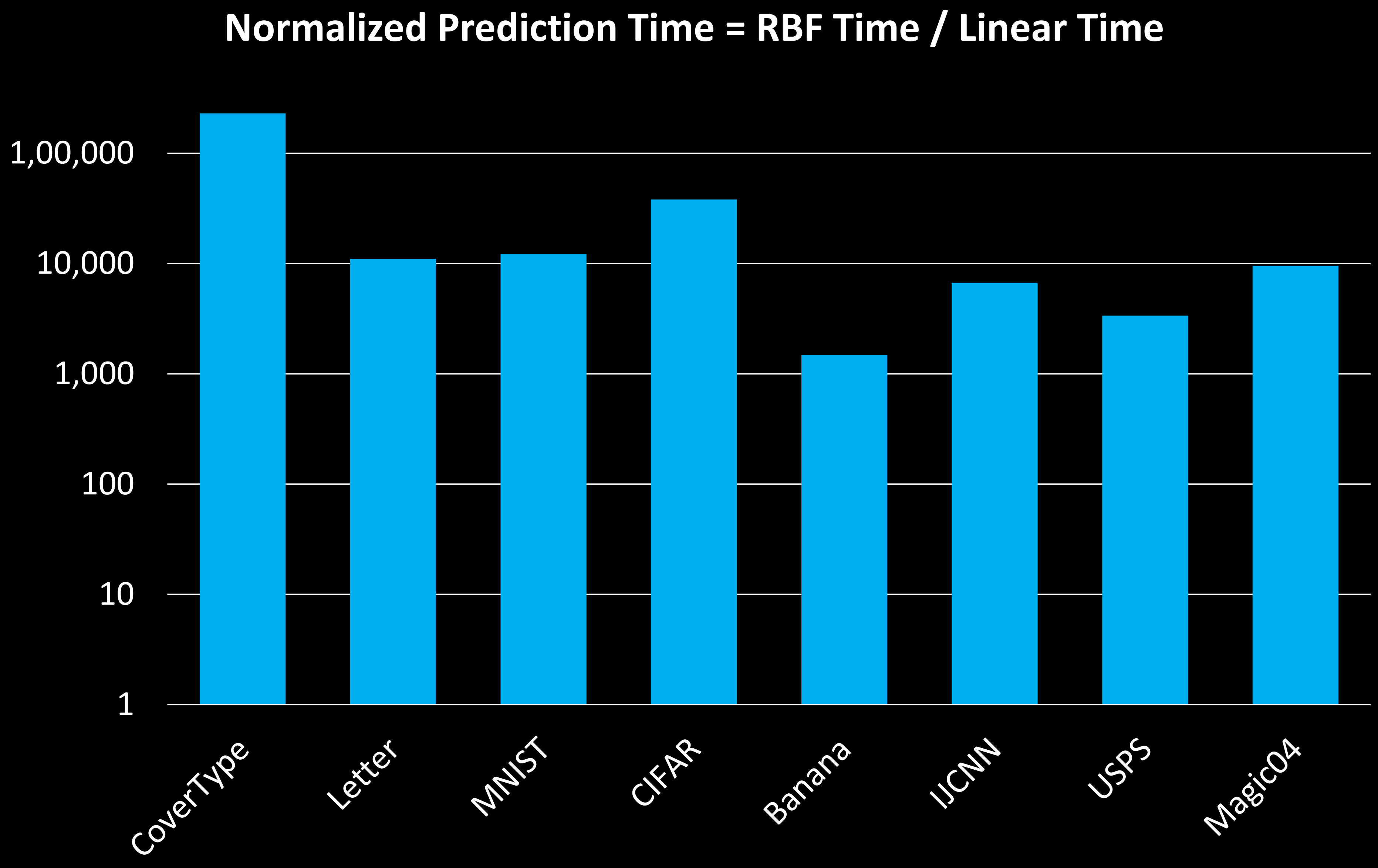
$$\nabla_{\mathbf{w}_k} P(\mathbf{x}_i) = \lambda_{\mathbf{W}}\mathbf{w}_k - \delta_i y_i \phi_{L_k}(\mathbf{x}_i)\mathbf{x}_i$$

$$\nabla_{\boldsymbol{\theta}_k} P(\mathbf{x}_i) = \lambda_{\mathbf{\Theta}}\boldsymbol{\theta}_k - \delta_i y_i \sum_l \tanh(\sigma \mathbf{v}_l^t \mathbf{x}_i)\, \nabla_{\boldsymbol{\theta}_k} I_l(\mathbf{x}_i)\mathbf{w}_l^t \mathbf{x}_i$$

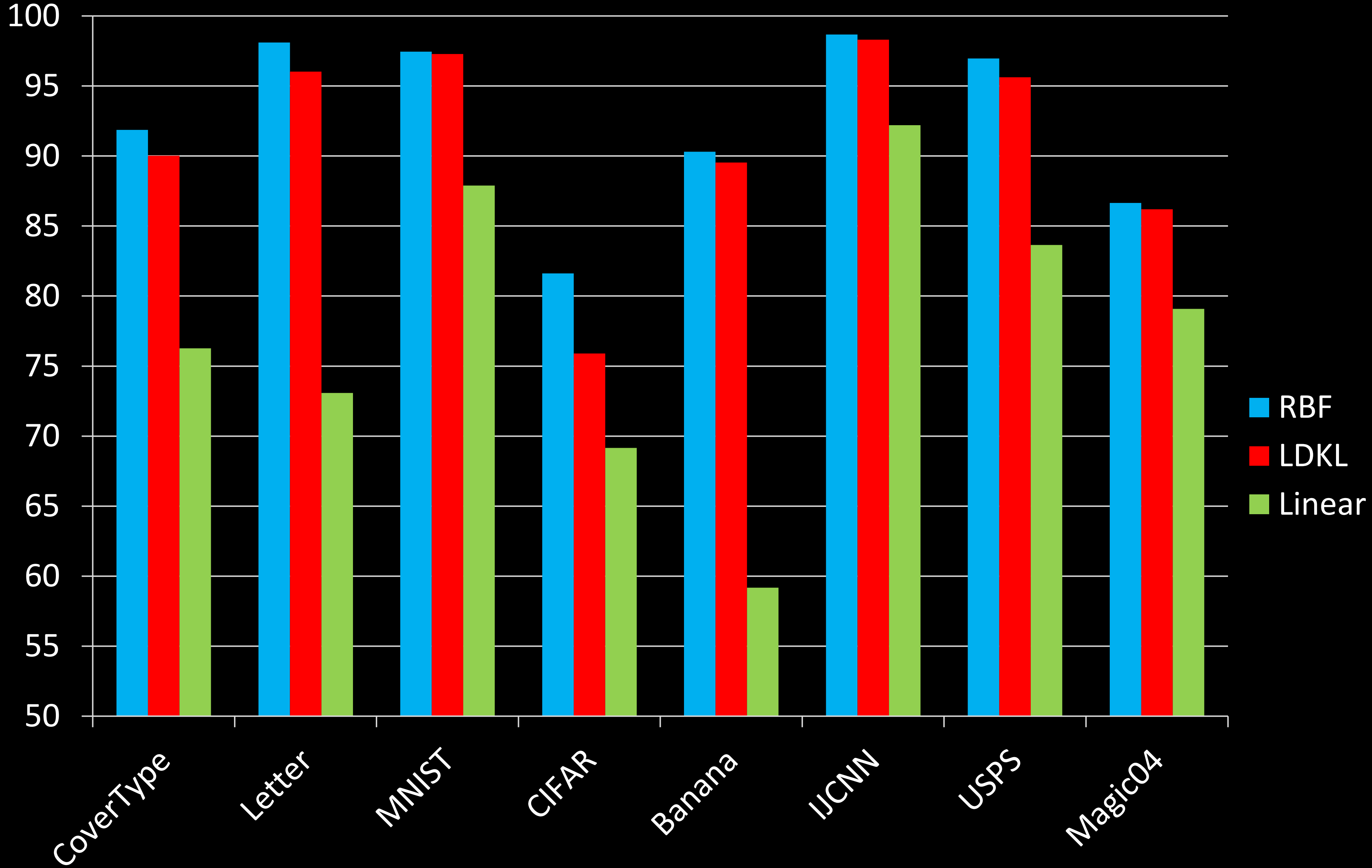
$$\nabla_{\mathbf{v}_k} P(\mathbf{x}_i) = \lambda_{\mathbf{V}}\mathbf{v}_k - \delta_i y_i \sigma(1 - \tanh^2(\sigma \mathbf{v}_k^t \mathbf{x}_i)) I_k(\mathbf{x}_i)\mathbf{w}_k^t \mathbf{x}_i \mathbf{x}_i$$

Accuracy Comparison: RBF vs Linear

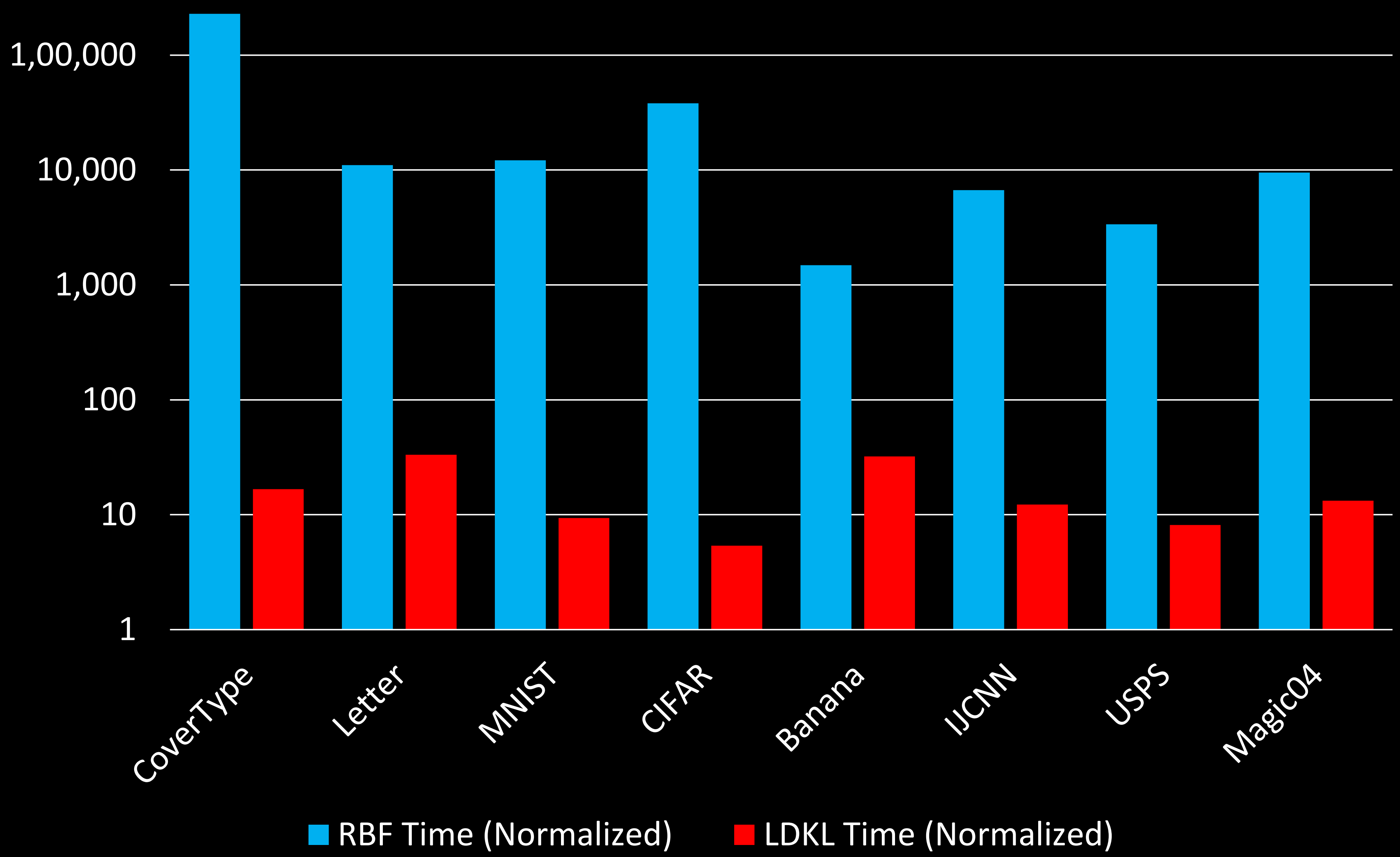
RBF-SVM: Cost of Prediction



LDKL's Prediction Accuracy

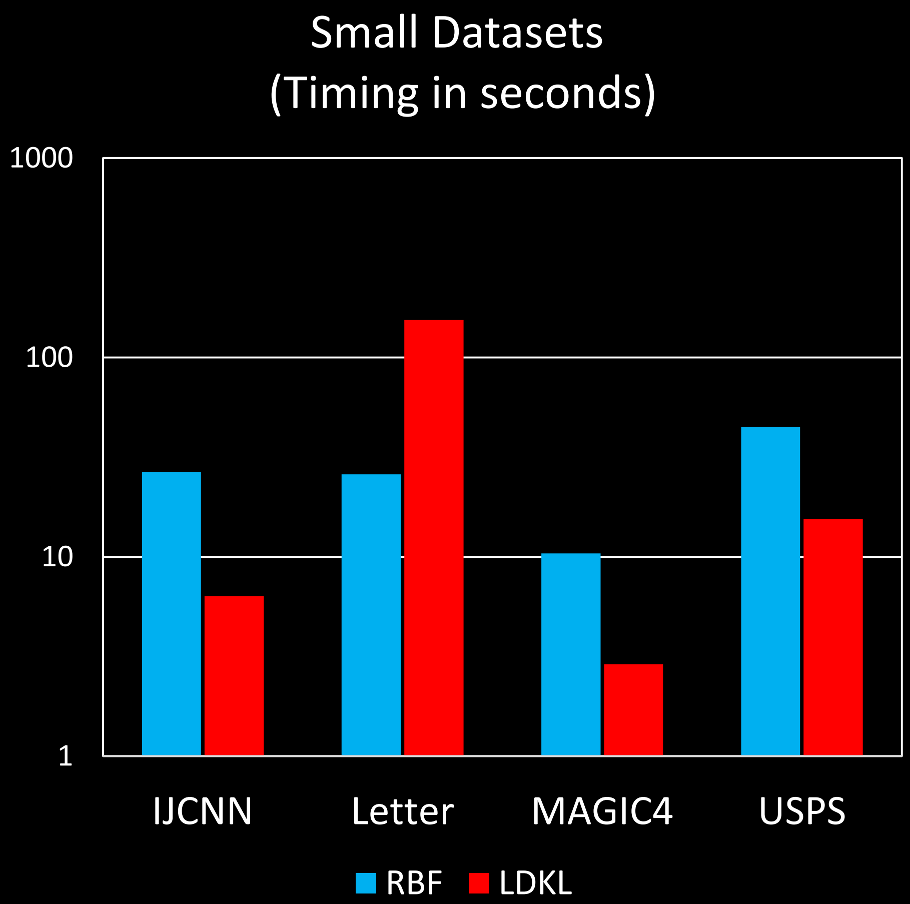


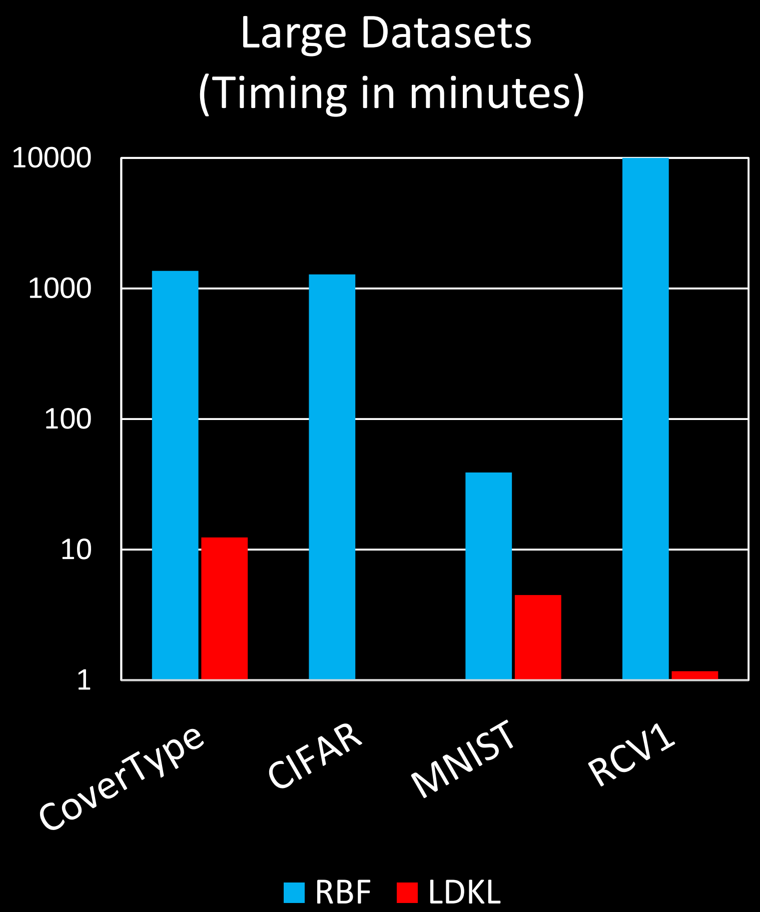
LDKL's Prediction Cost



Training Time Comparison: RBF vs LDKL

- Training time on a single core of a 2.68 Ghz Xeon processor with 8 GB RAM.



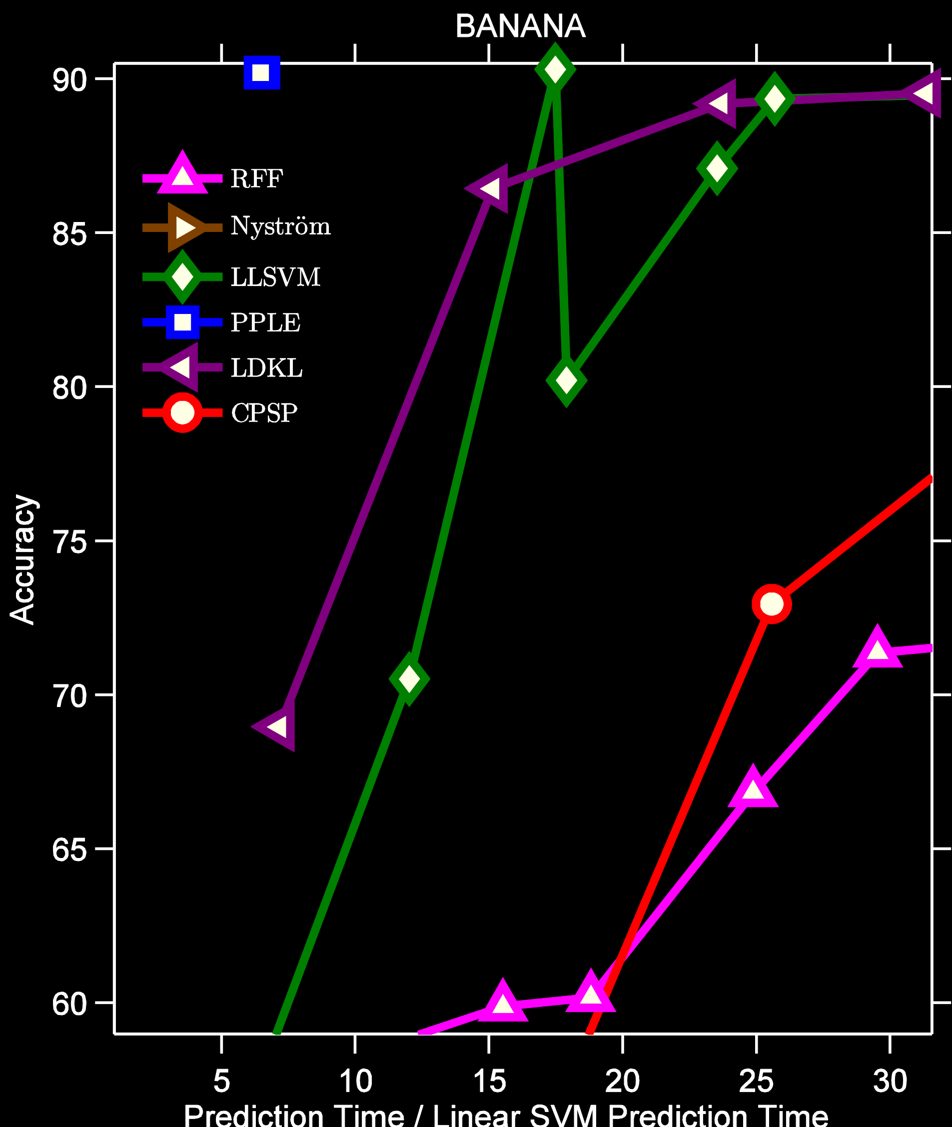


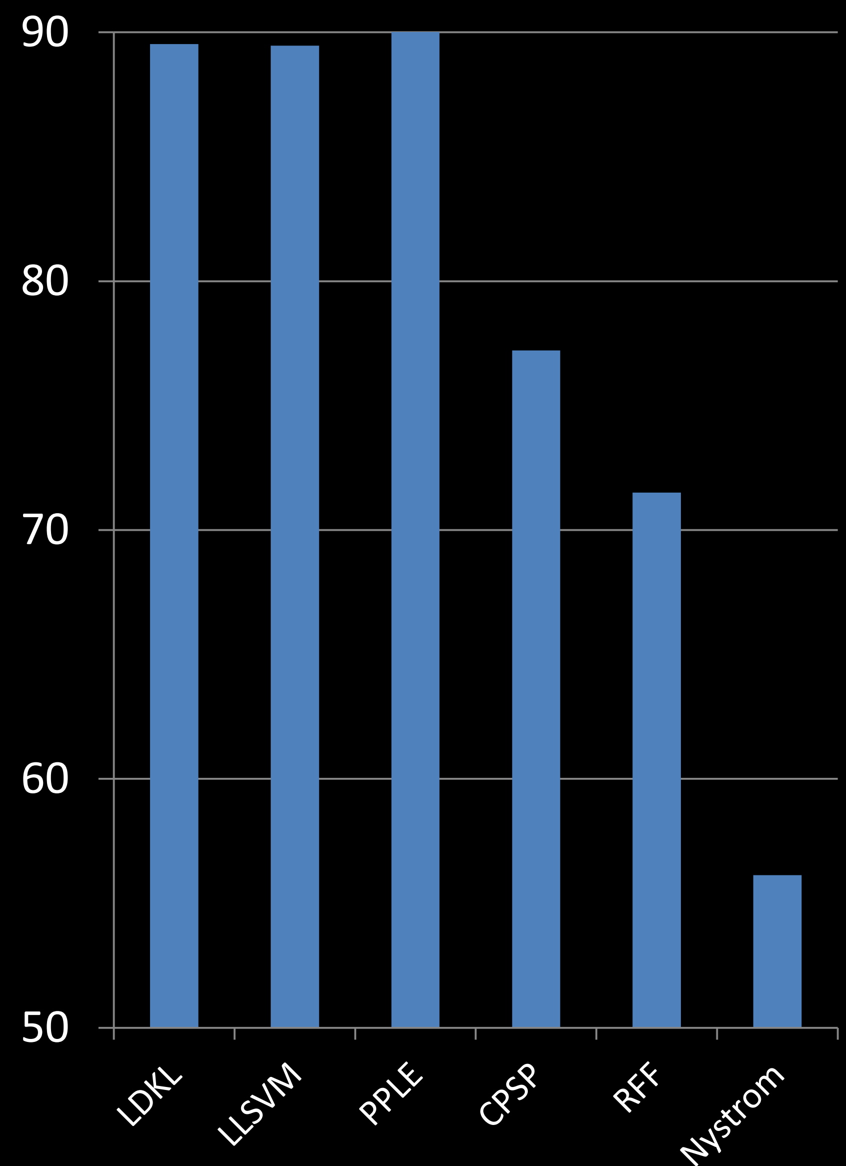
Training Time Comparison: RBF vs LDKL

- Training time in minutes on a single core of a 2.68 Ghz Xeon processor with 8 GB RAM.

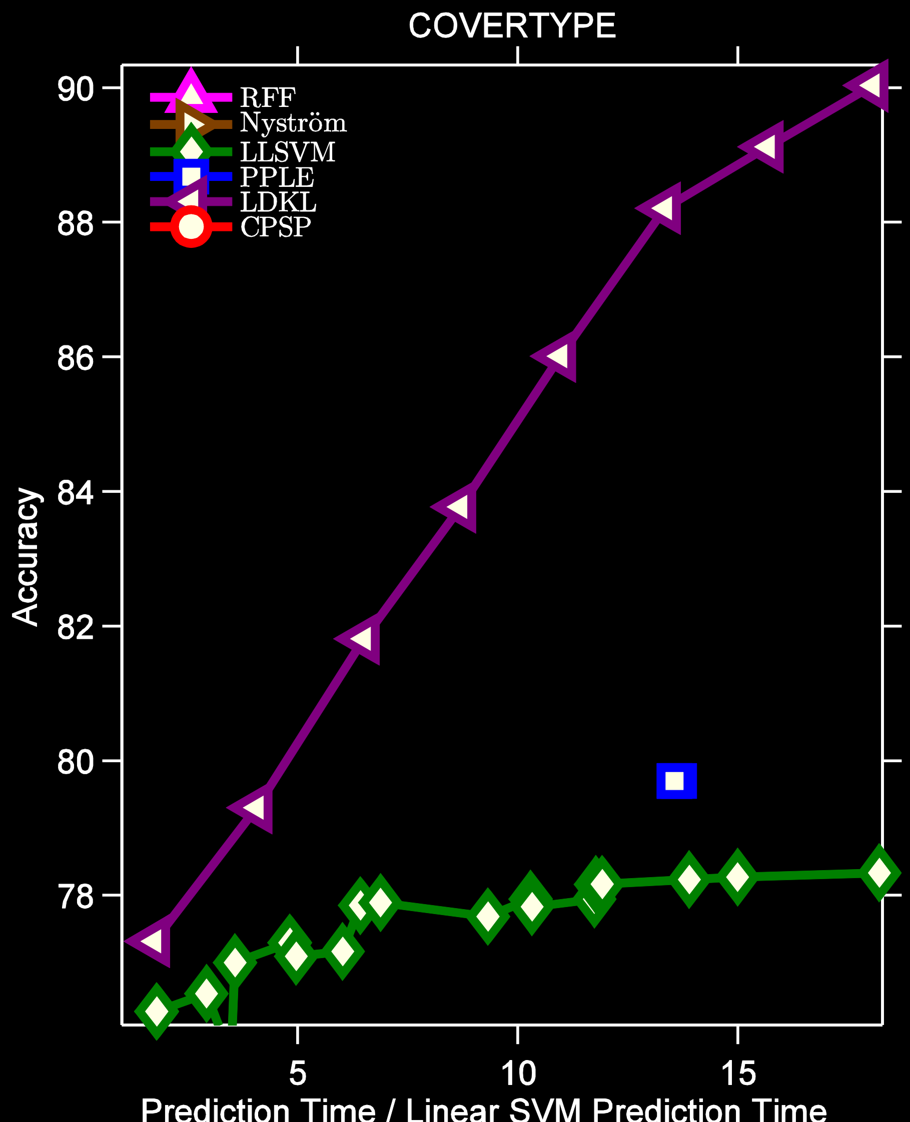
Data Set	Linear SVM	RBF-SVM	LDKL
BANANA	1.48E-05	0.006	0.01
CIFAR	6.76E-02	1283.68	0.278
CoverType	2.35	1369.96	7.990
IJCNN	3.93E-02	0.45	0.090
Letter	5.75E-04	0.43	2.200
Magic04	1.74E-03	0.17	0.047
MNIST	2.86E-01	39.12	1.376
USPS	5.97E-03	0.75	0.096
RCV1	0.13	-	0.5
MNIST8M	0.7	-	65.21

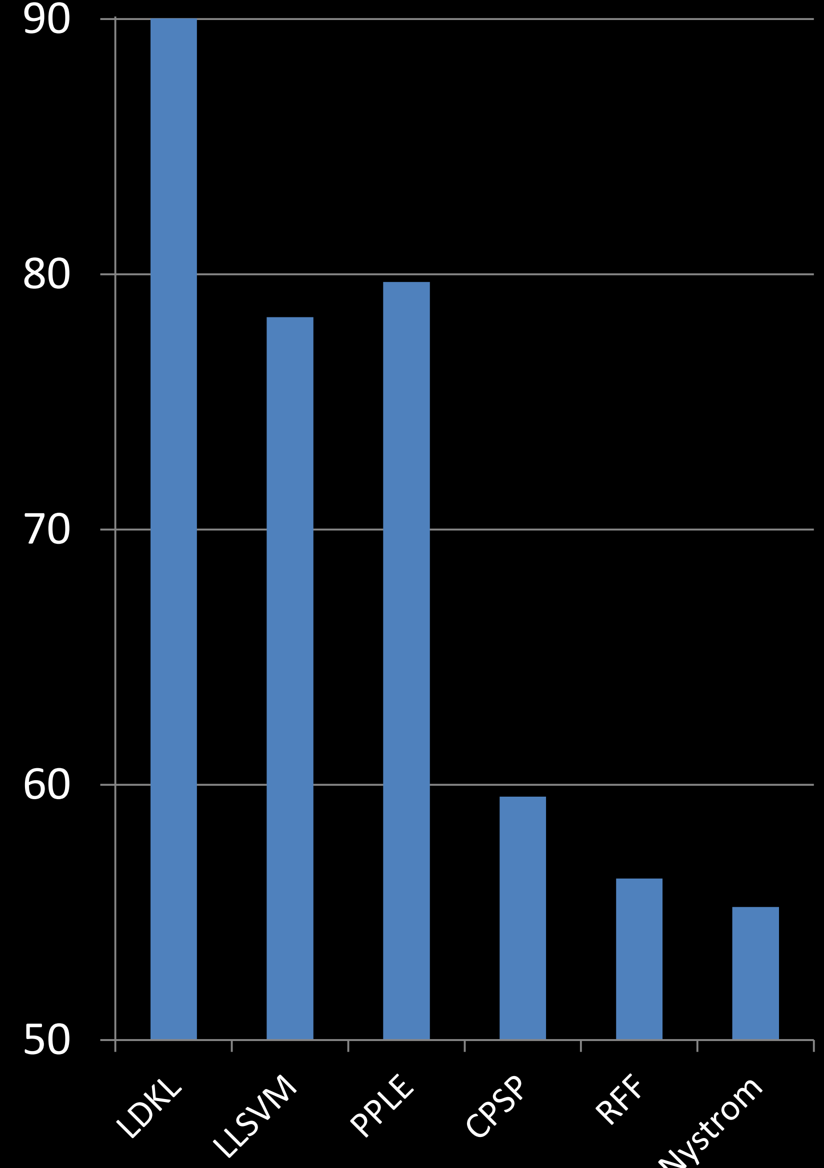
Banana


BANANA
Accuracy
Prediction Time / Linear SVM Prediction Time
RFF
Nyström
LLSVM
PPLE
LDKL
CPSP
90
85
80
75
70
65
60
5
10
15
20
25
30

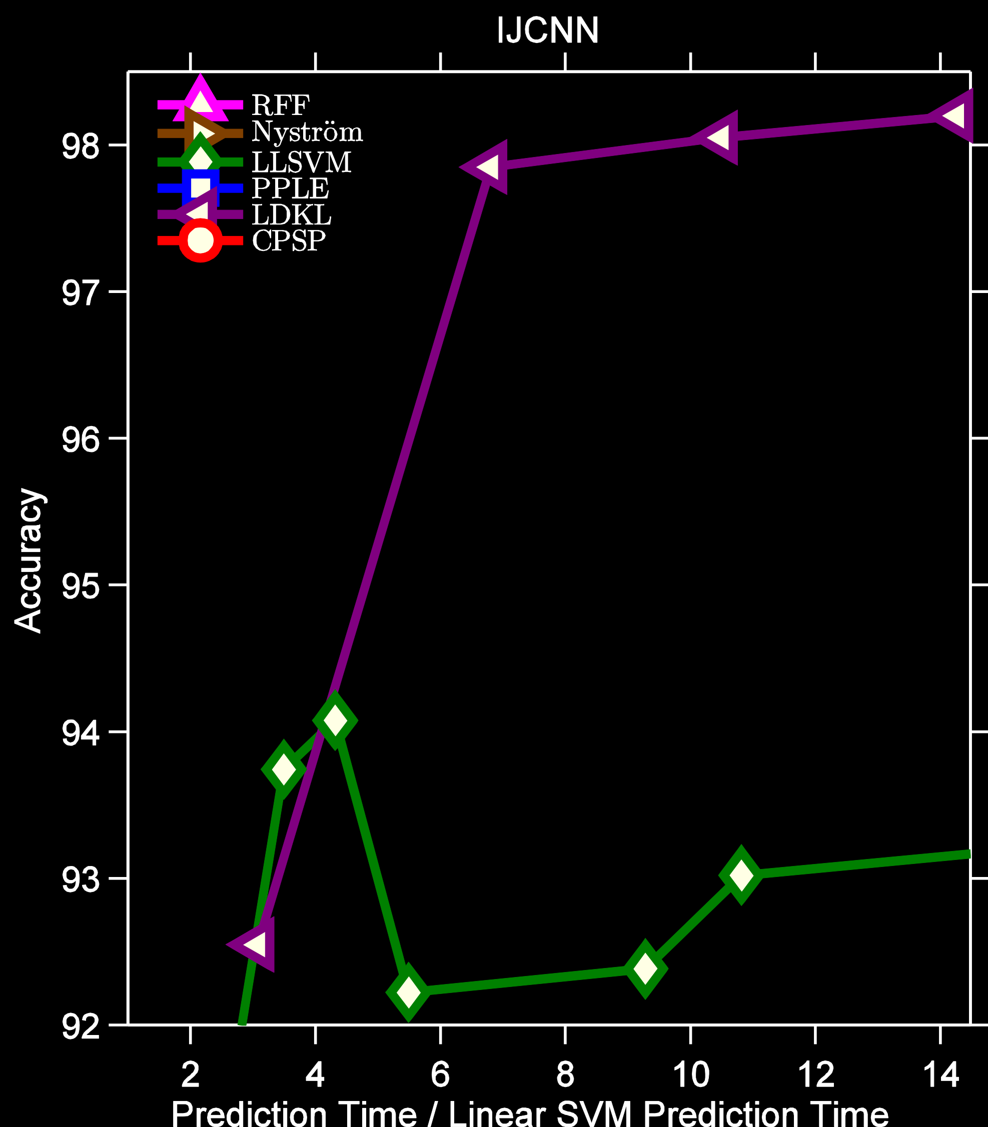

90
80
70
60
50
LDKL
LLSVM
PPLE
CPSP
RFF
Nystrom

CoverType

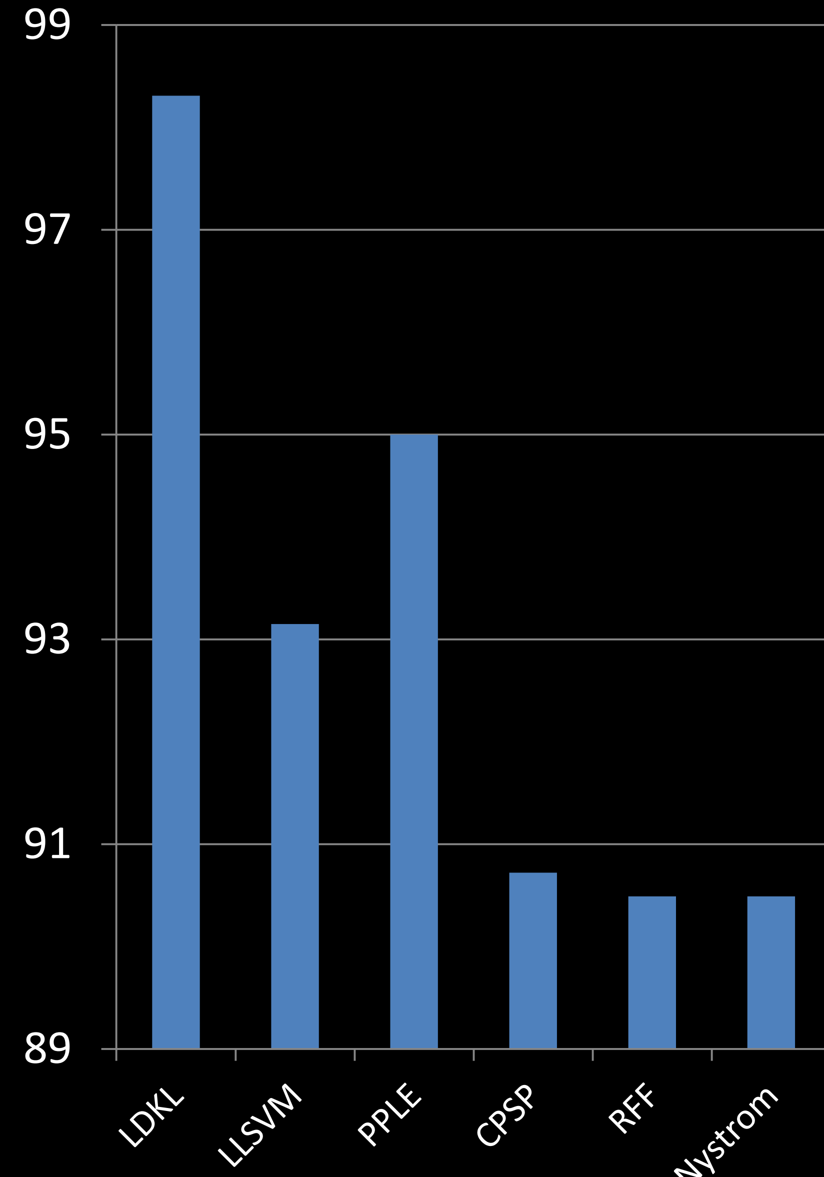

COVERTYPE
RFF
Nyström
LLSVM
PPLE
LDKL
CPSP
Accuracy
90
88
86
84
82
80
78
5
10
15
Prediction Time / Linear SVM Prediction Time


90
80
70
60
50
LDKL
LLSVM
PPLE
CPSP
RFF
Nystrom

IJCNN

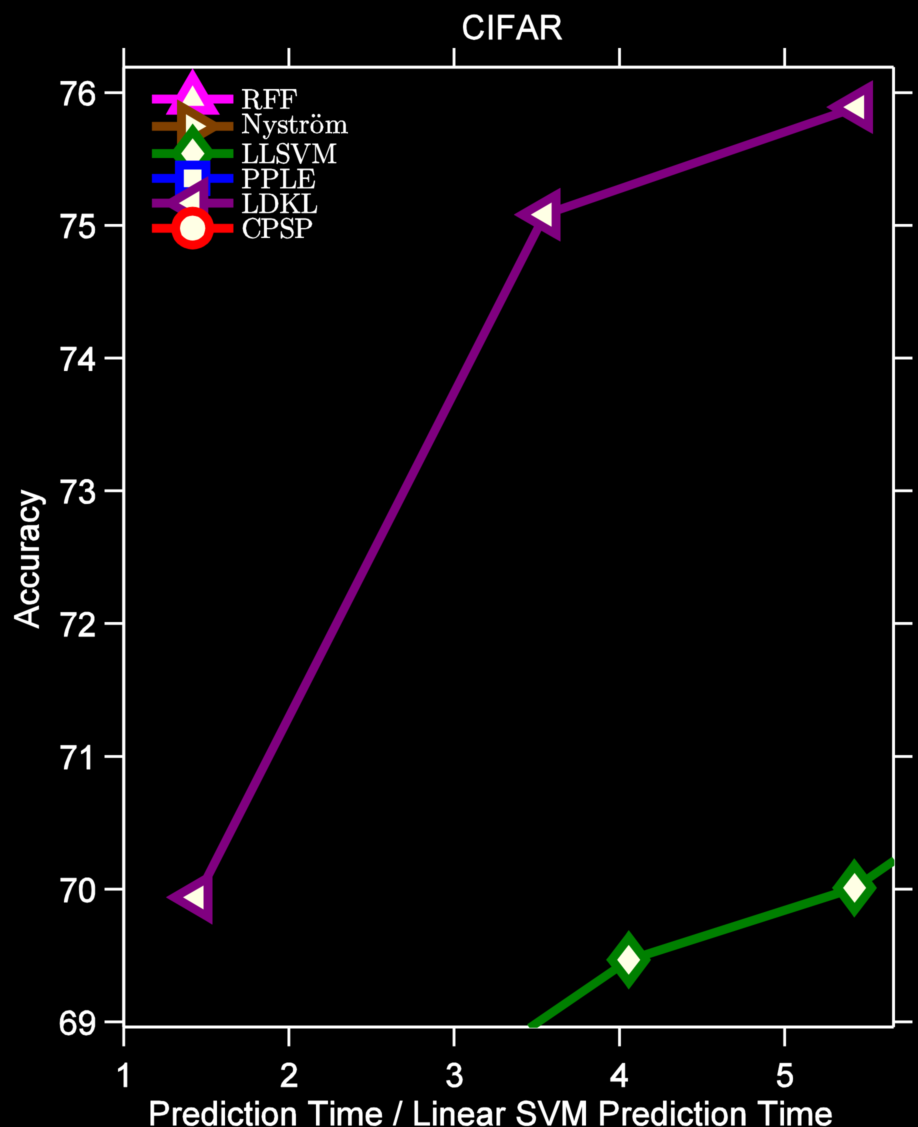


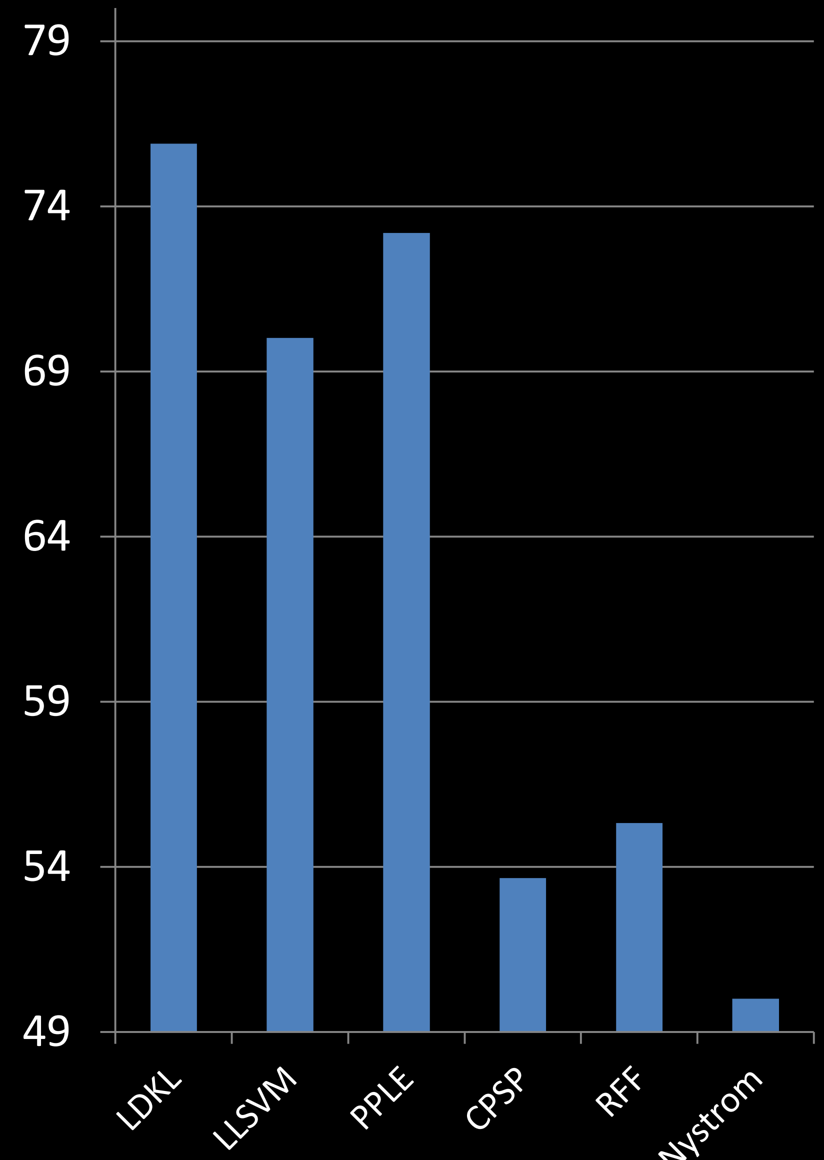
IJCNN
RFF
Nyström
LLSVM
PPLE
LDKL
CPSP
Accuracy
92
93
94
95
96
97
98
2
4
6
8
10
12
14
Prediction Time / Linear SVM Prediction Time



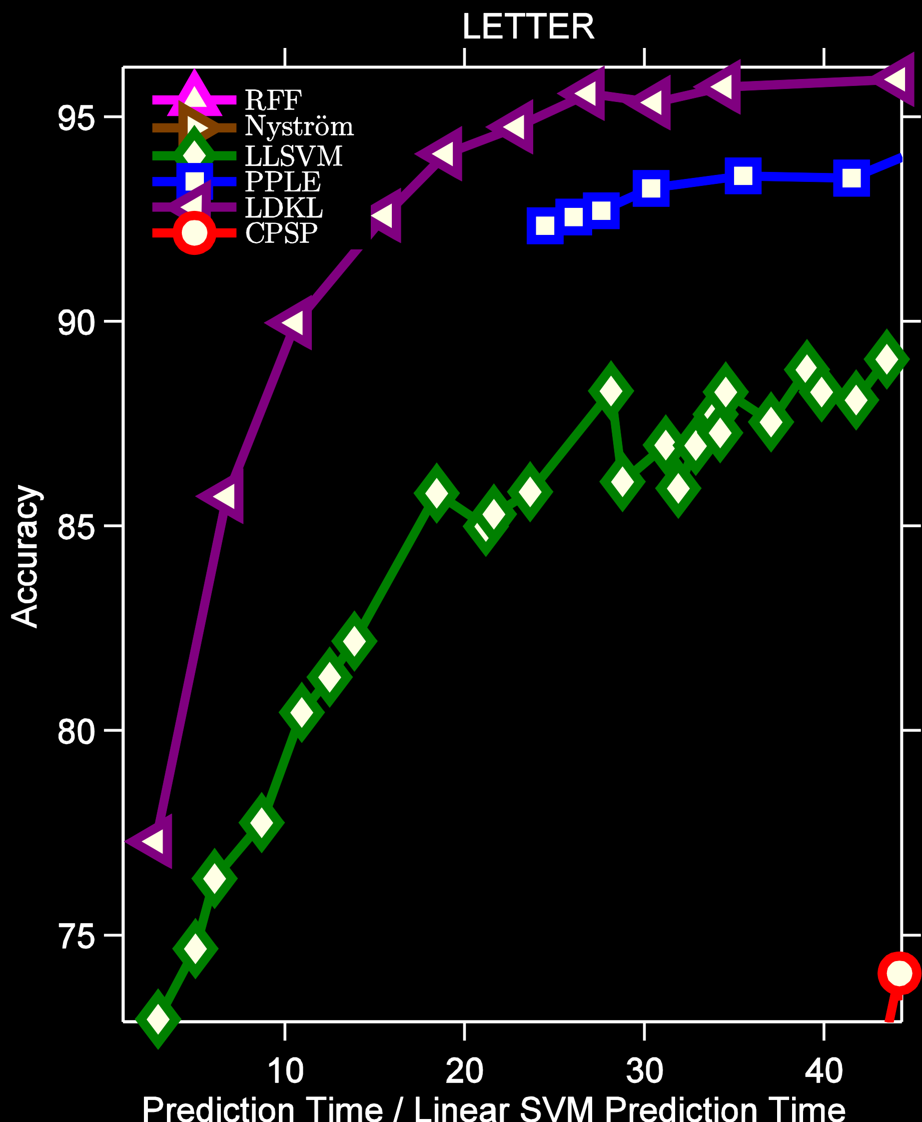
99
97
95
93
91
89
LDKL
LLSVM
PPLE
CPSP
RFF
Nystrom

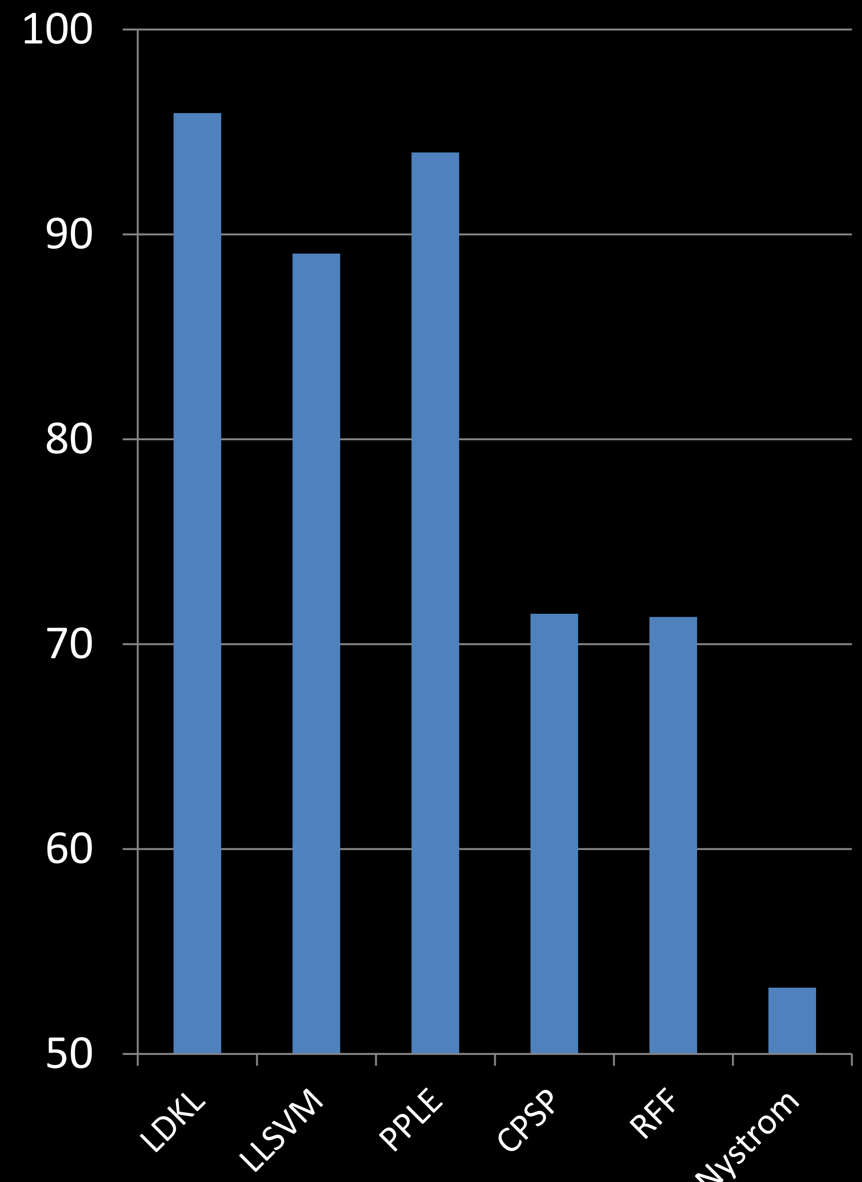
CIFAR10


CIFAR
RFF
Nyström
LLSVM
PPLE
LDKL
CPSP
Accuracy
76
75
74
73
72
71
70
69
1
2
3
4
5
Prediction Time / Linear SVM Prediction Time

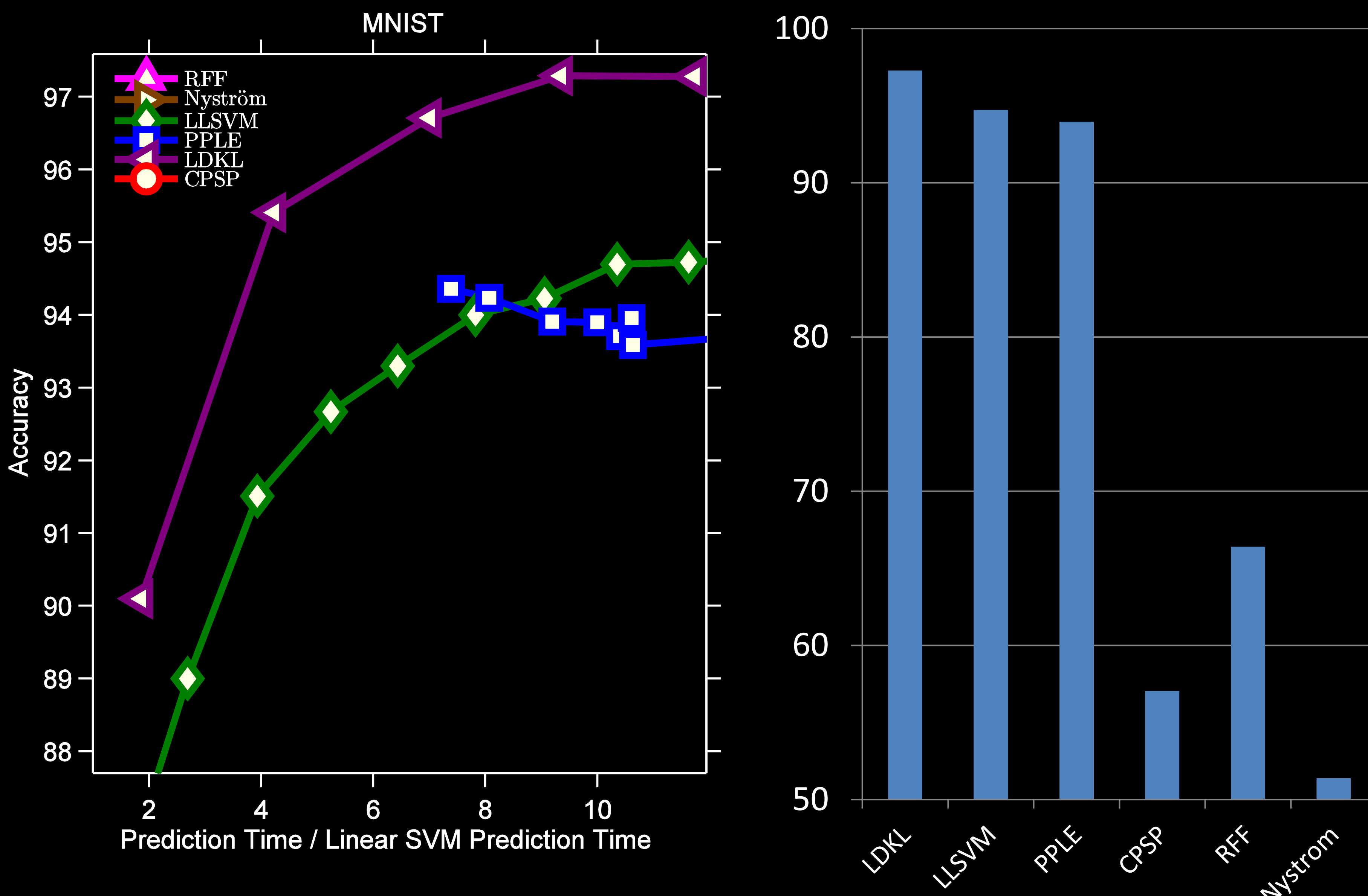

79
74
69
64
59
54
49
LDKL
LLSVM
PPLE
CPSP
RFF
Nystrom

Letter

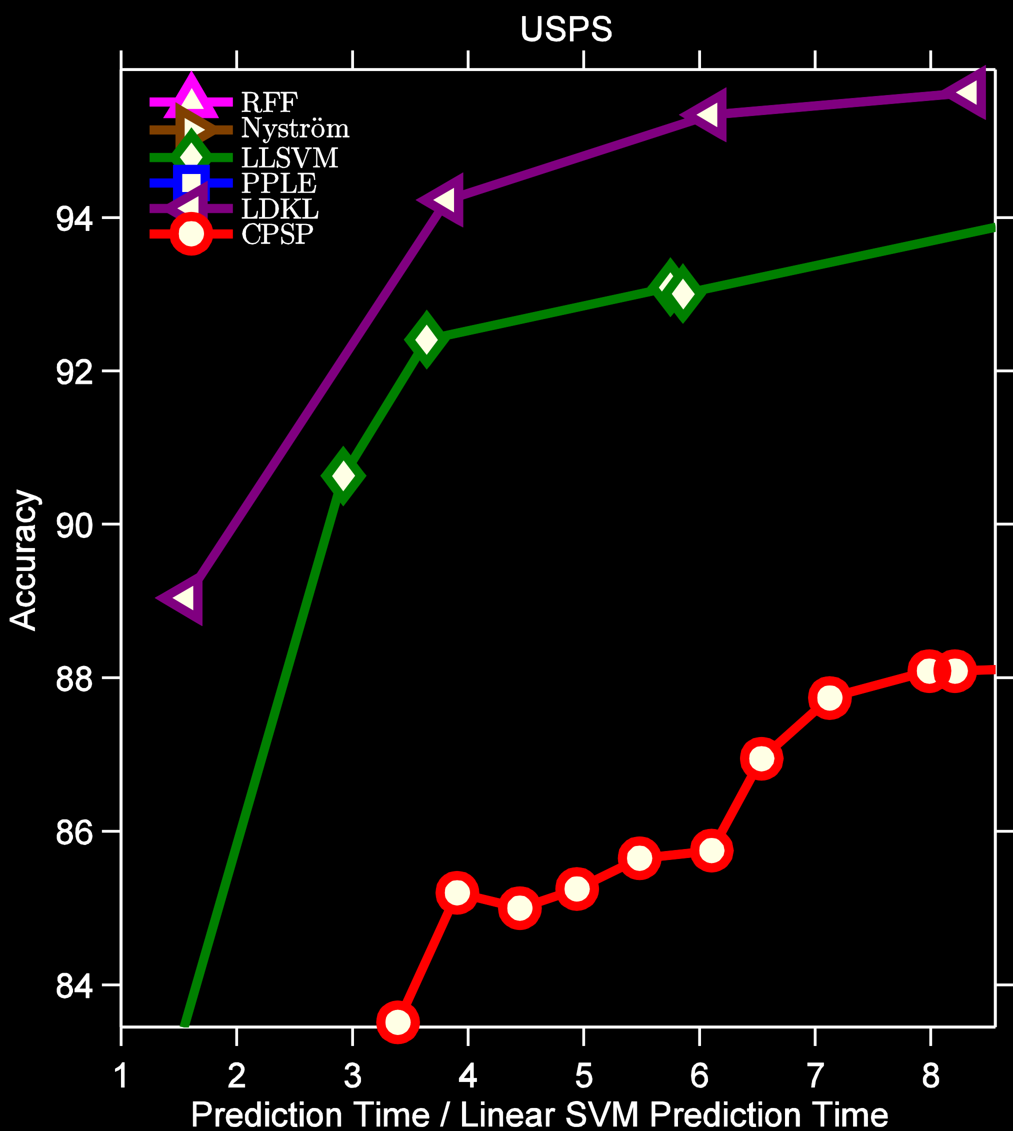

LETTER
RFF
Nyström
LLSVM
PPLE
LDKL
CPSP
95
90
85
80
75
Accuracy
10
20
30
40
Prediction Time / Linear SVM Prediction Time

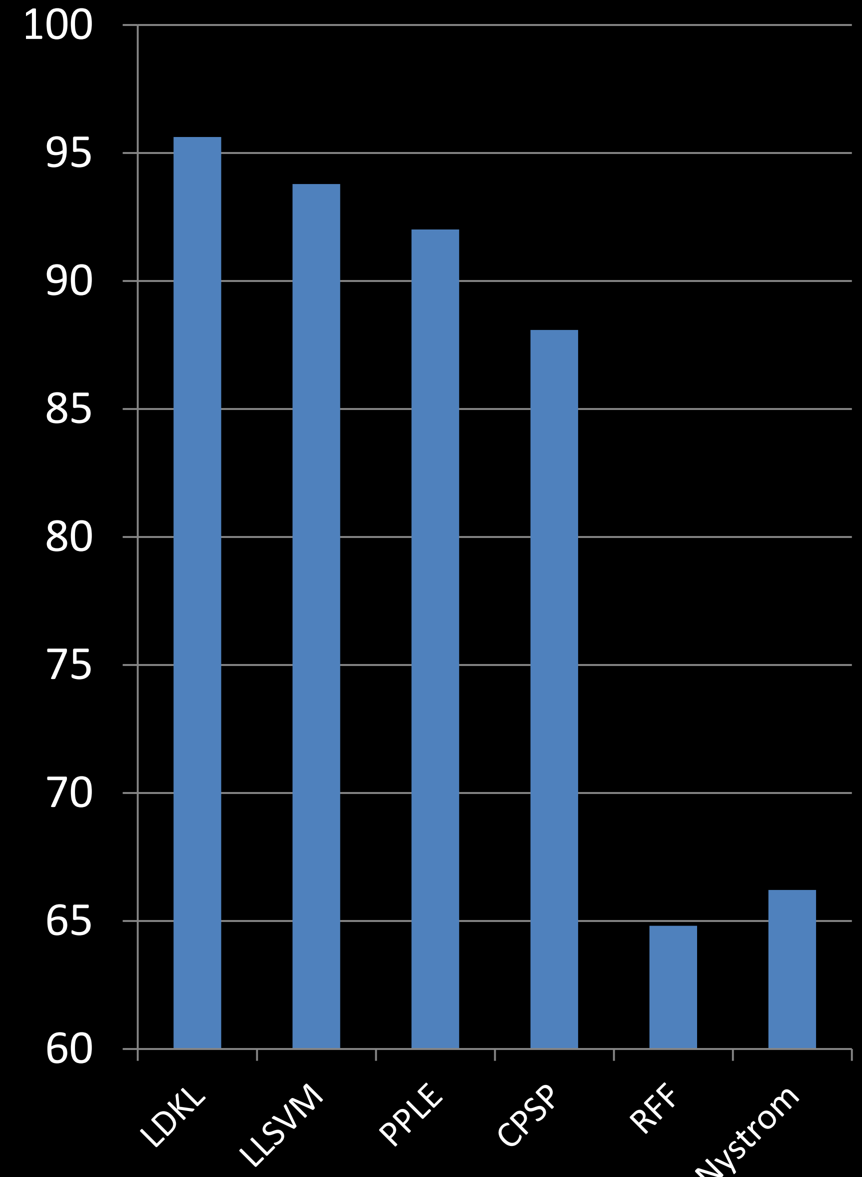

100
90
80
70
60
50
LDKL
LLSVM
PPLE
CPSP
RFF
Nystrom

MNIST

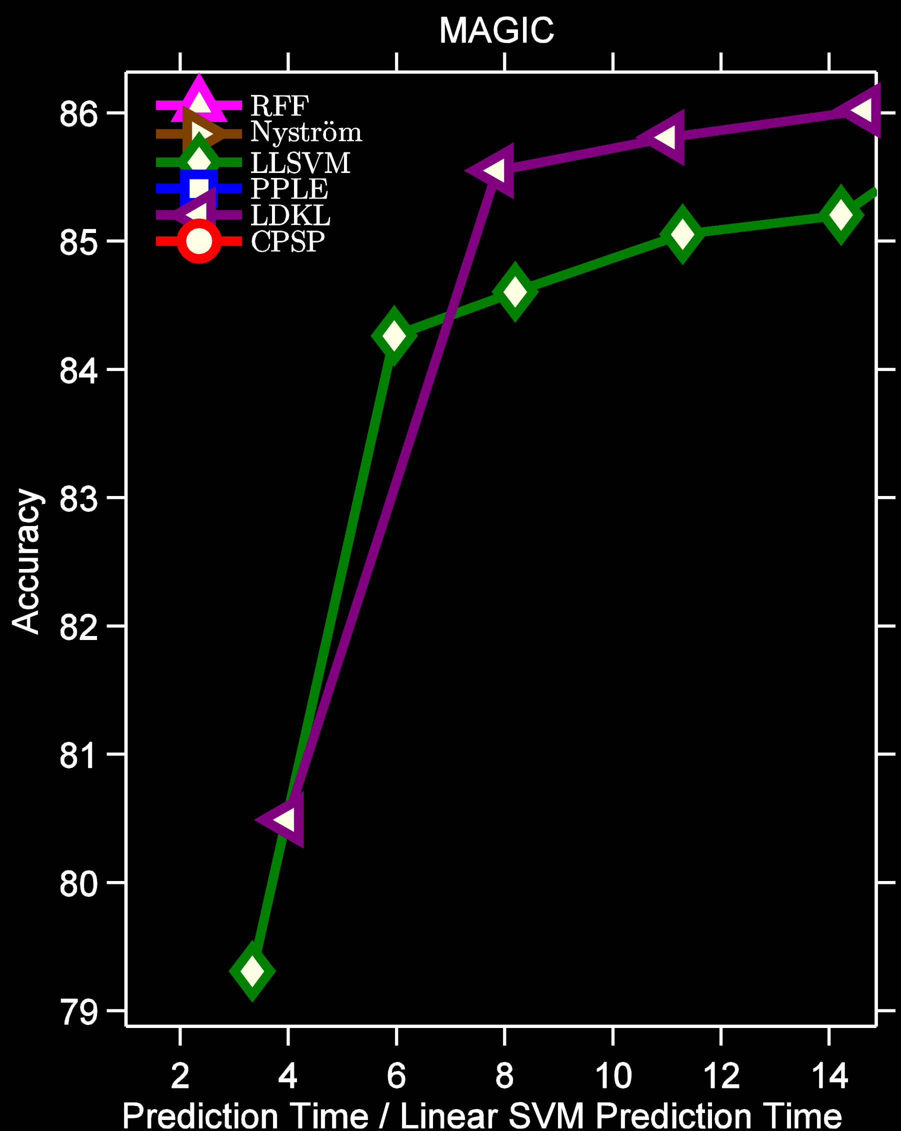

MNIST
RFF
Nyström
LLSVM
PPLE
LDKL
CPSP
Accuracy
Prediction Time / Linear SVM Prediction Time
88
89
90
91
92
93
94
95
96
97
2
4
6
8
10
100
90
80
70
60
50
LDKL
LLSVM
PPLE
CPSP
RFF
Nystrom

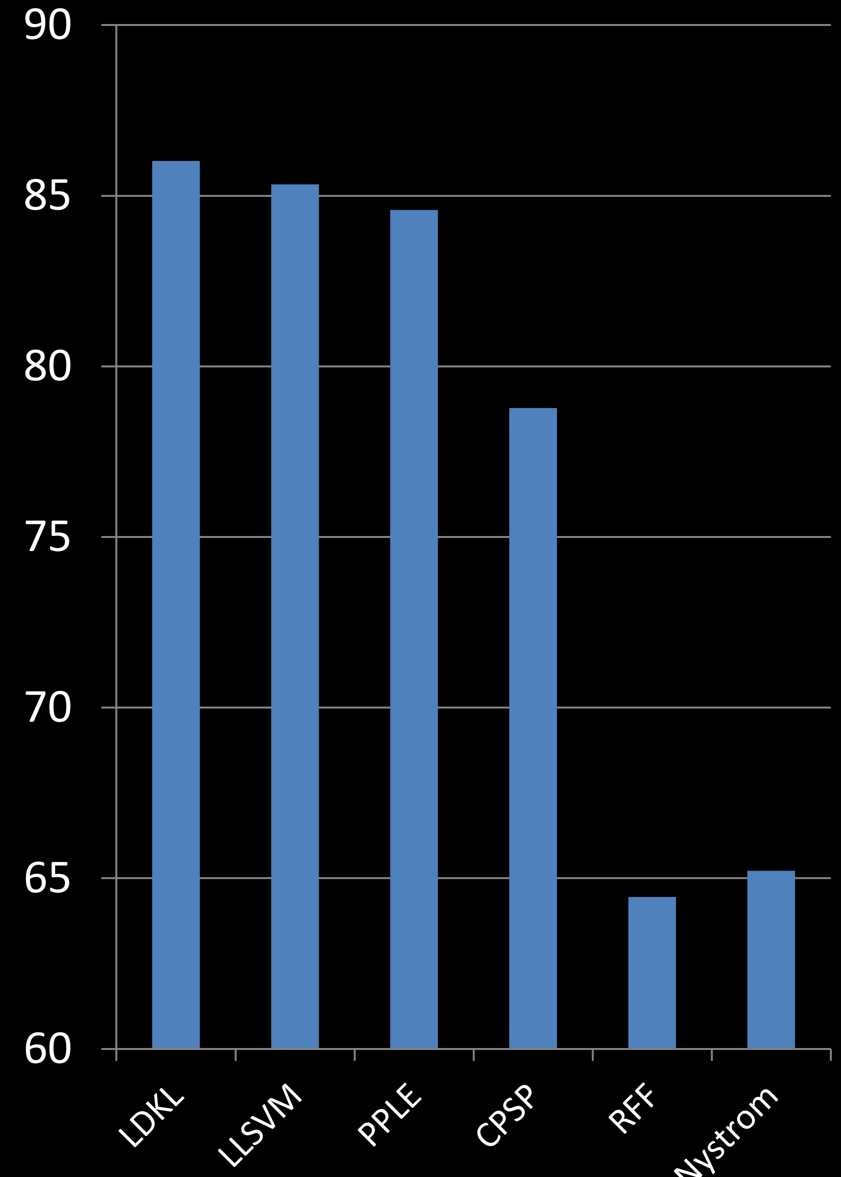
USPS


USPS
RFF
Nyström
LLSVM
PPLE
LDKL
CPSP
Accuracy
84
86
88
90
92
94
1
2
3
4
5
6
7
8
Prediction Time / Linear SVM Prediction Time

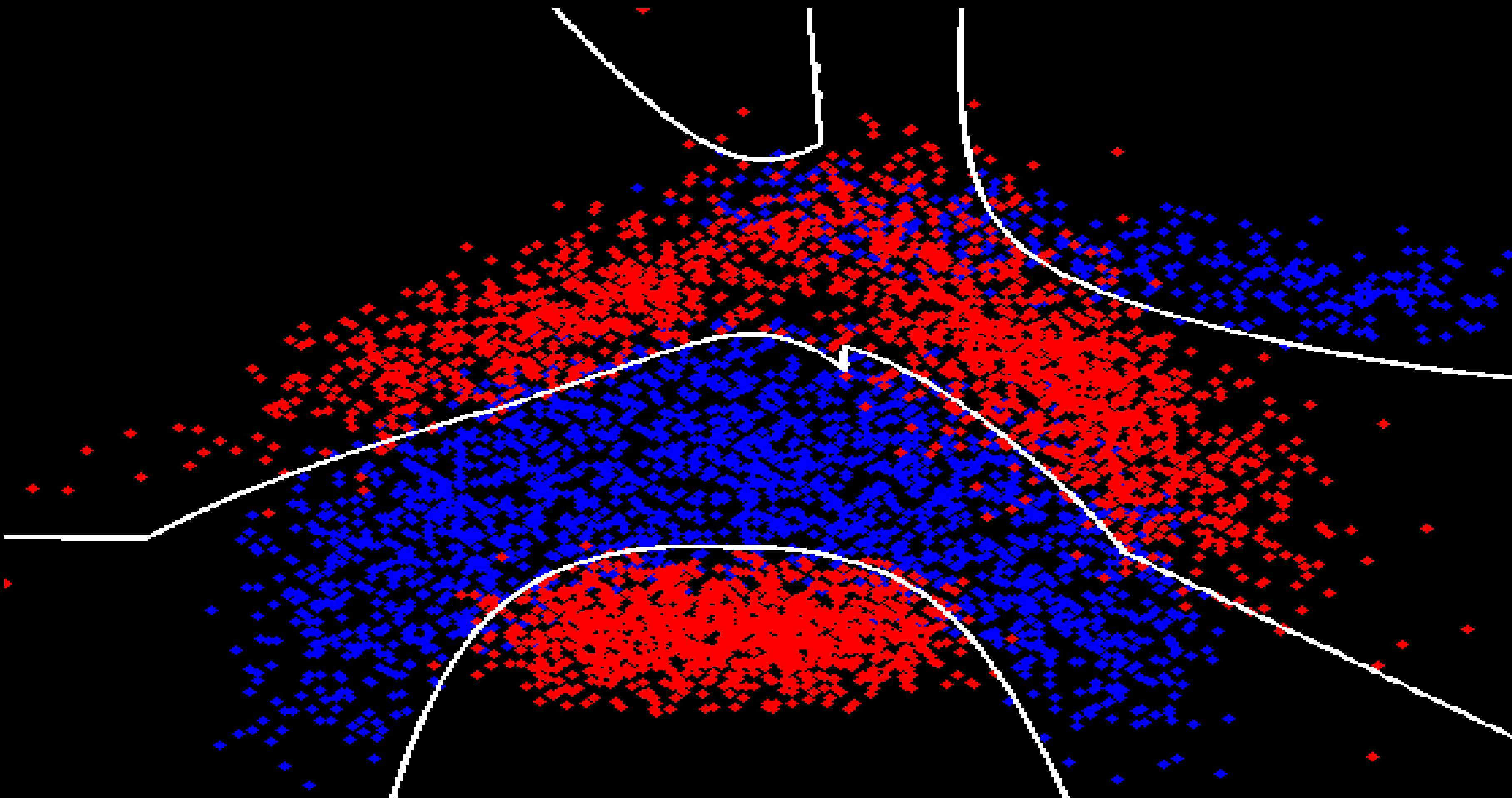

100
95
90
85
80
75
70
65
60
LDKL
LLSVM
PPLE
CPSP
RFF
Nystrom

Magic04

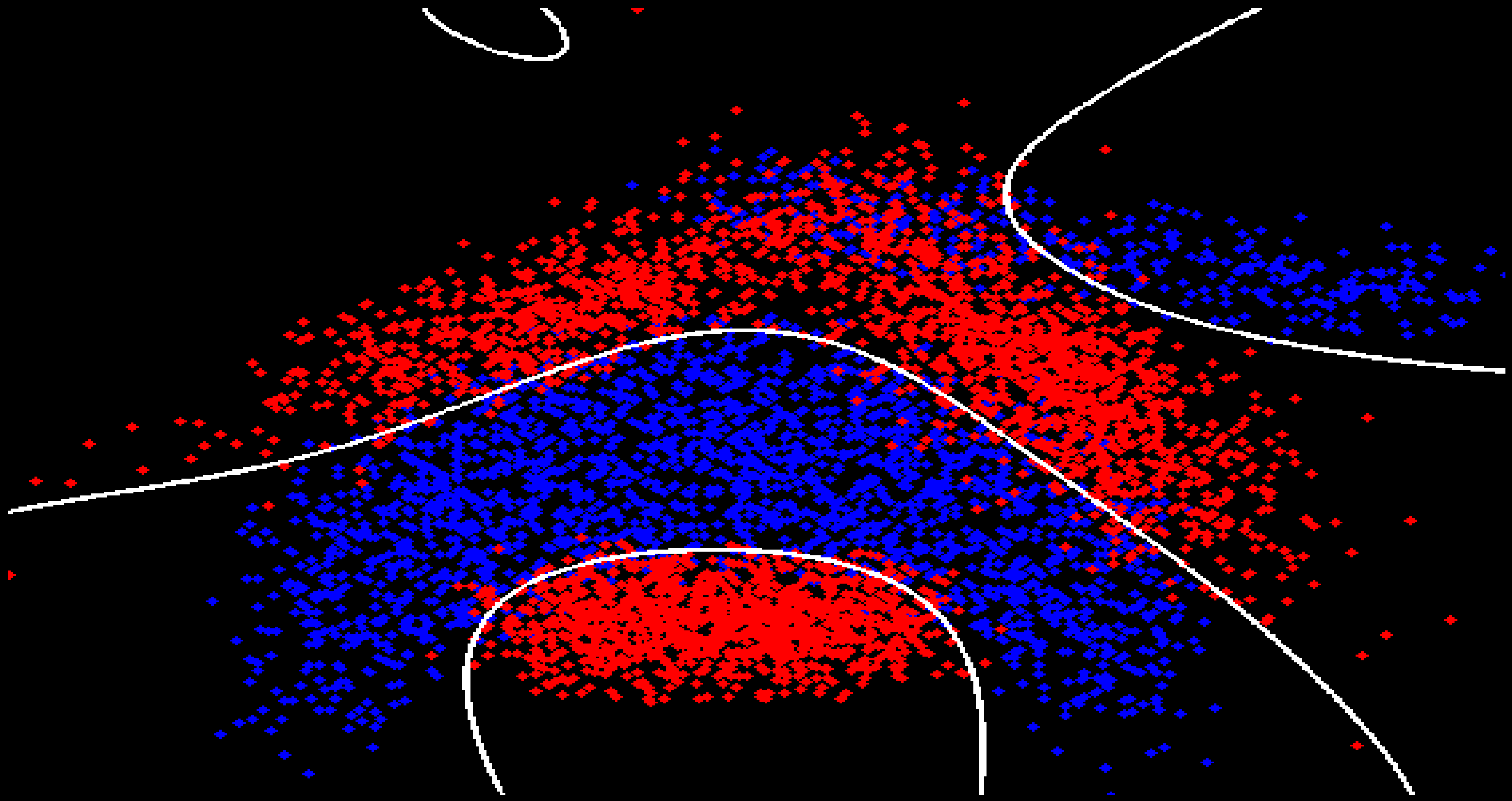

MAGIC
RFF
Nyström
LLSVM
PPLE
LDKL
CPSP
Accuracy
86
85
84
83
82
81
80
79
2
4
6
8
10
12
14
Prediction Time / Linear SVM Prediction Time


90
85
80
75
70
65
60
LDKL
LLSVM
PPLE
CPSP
RFF
Nystrom

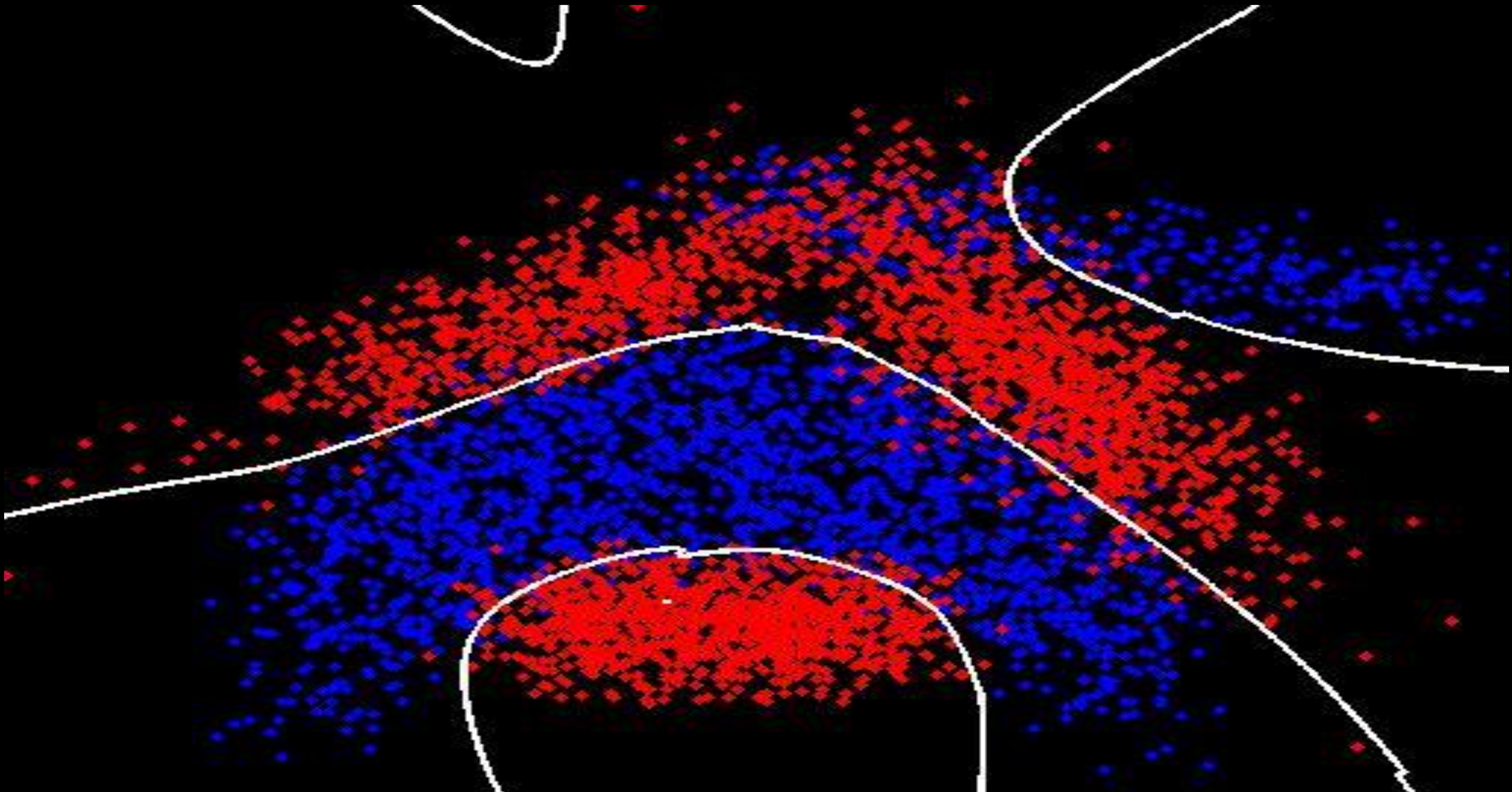
LDKL's Decision Boundaries



The RBF-SVM's Decision Boundaries



LDKL: Mimicking the RBF-SVM

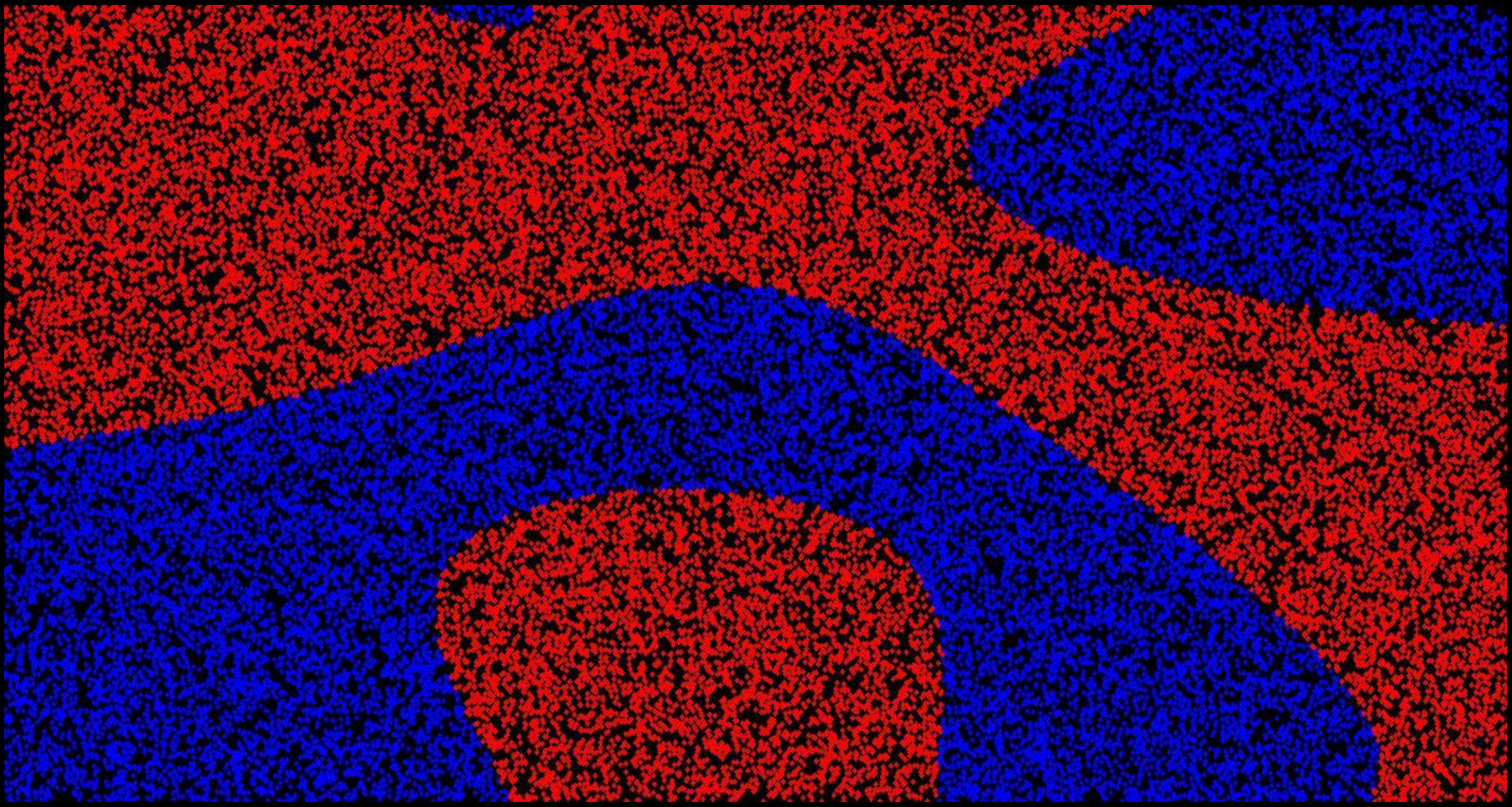


Pruning LDKL

Dataset	Accuracy (%)		Prediction Time (ms)		# Nodes	
	Original Tree	Pruned Tree	Original Tree	Pruned Tree	Original Tree	Pruned Tree
Banana	89.53	89.50	1.08	1.07	15	12
CoverType	90.22	90.26	104.38	96.9	255	230
CIFAR	76.14	76.14	38.53	37.41	7	7
IJCNN	98.20	98.20	43.2	38.38	15	9
Letter	95.94	95.83	6.68	4.82	1023	115
Magic	85.93	86.01	1.49	1.28	15	9
MNIST	97.27	97.26	128.43	123.48	31	24
USPS	95.51	95.46	6.95	6.58	15	13

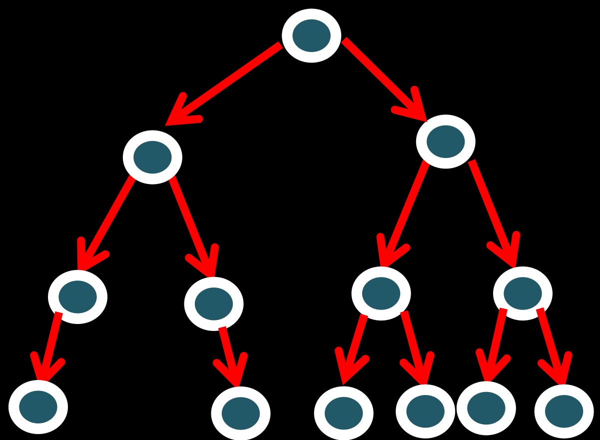
Generating Training Data

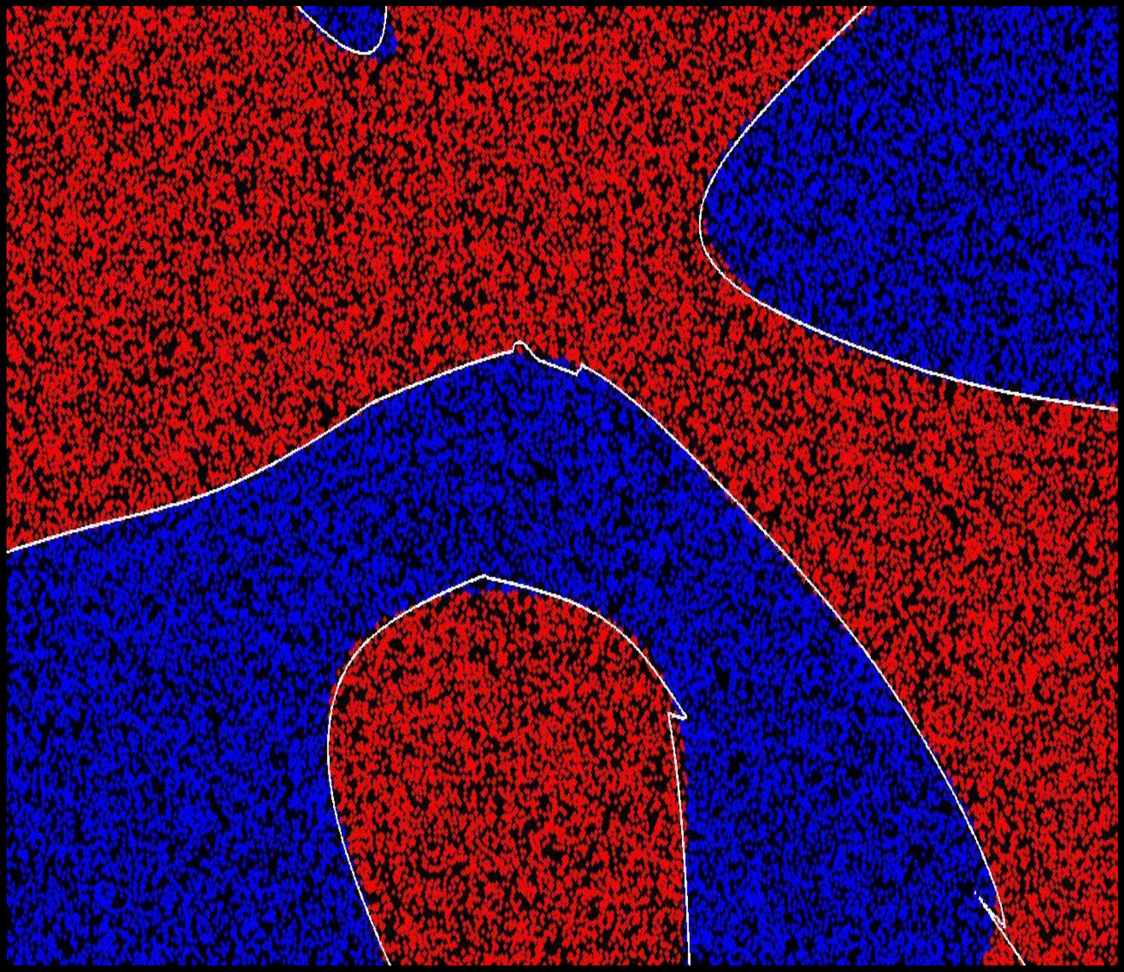
- Sample points and have them labelled by the RBF-SVM



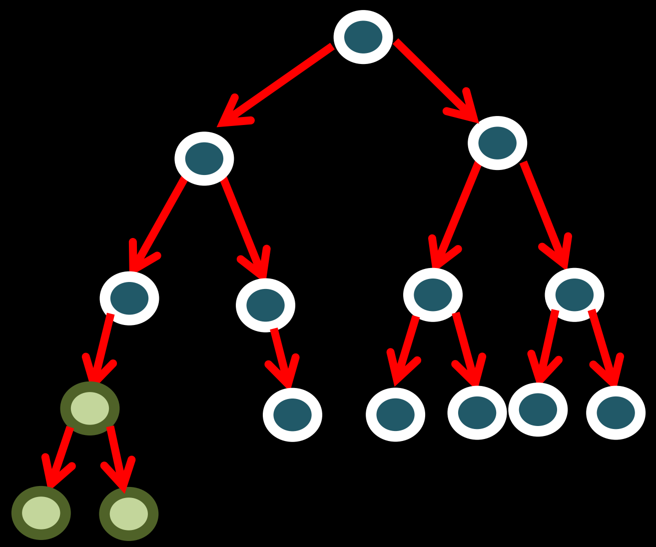
Training LDKL on the Extended Data Set

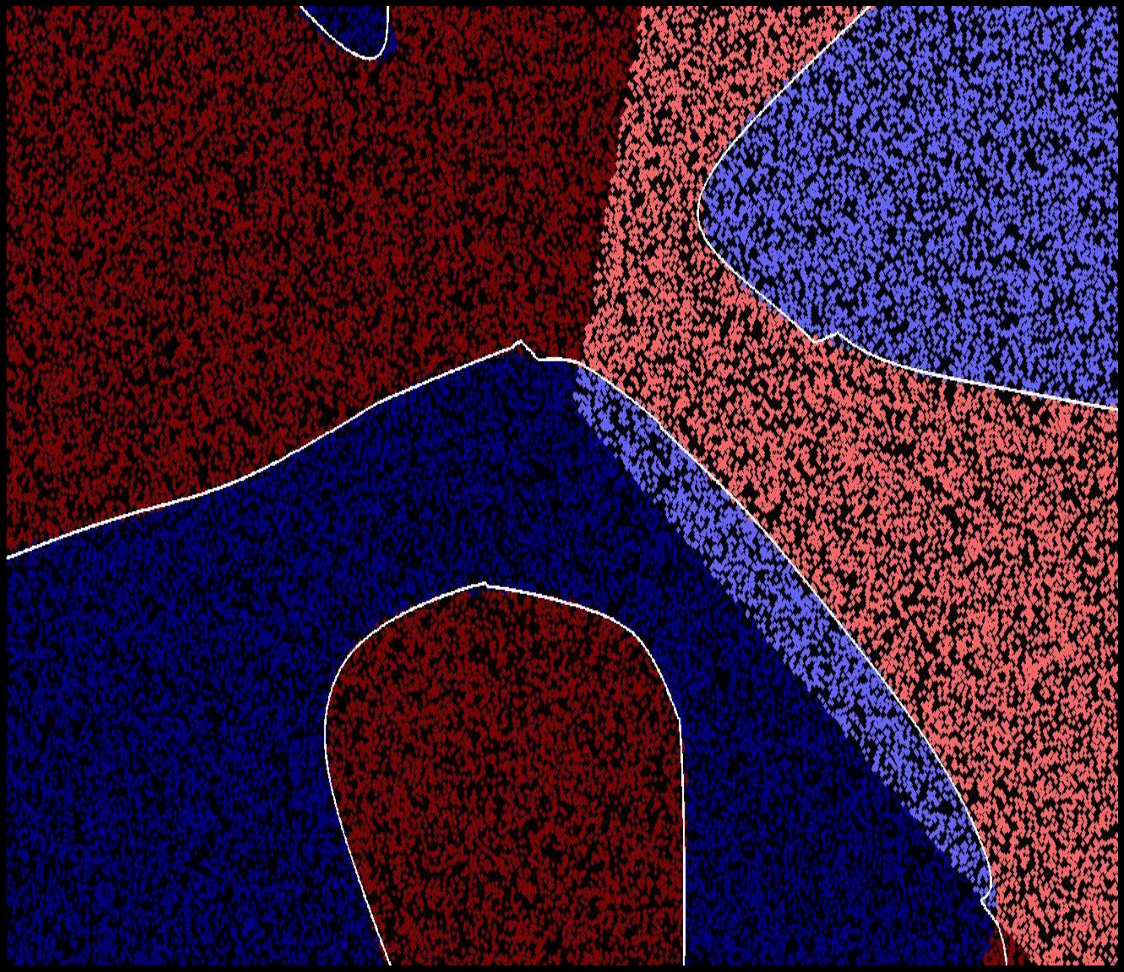
- Learn a balanced LDKL tree and then prune irrelevant nodes



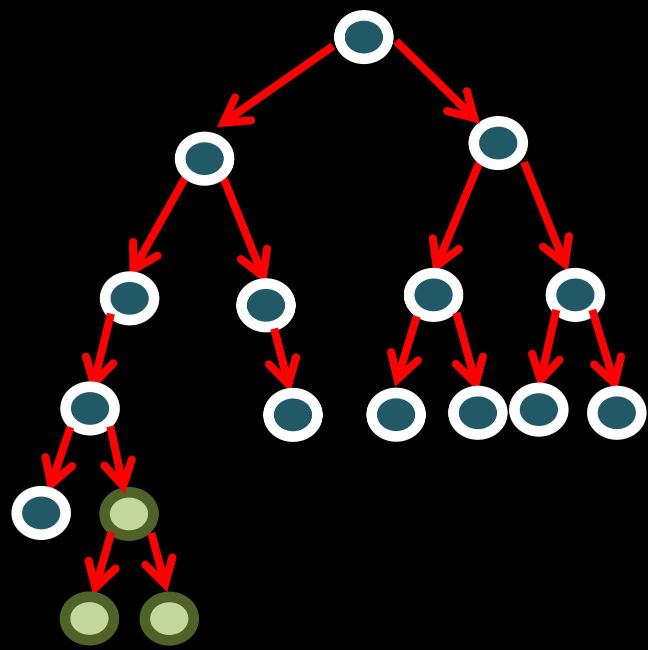


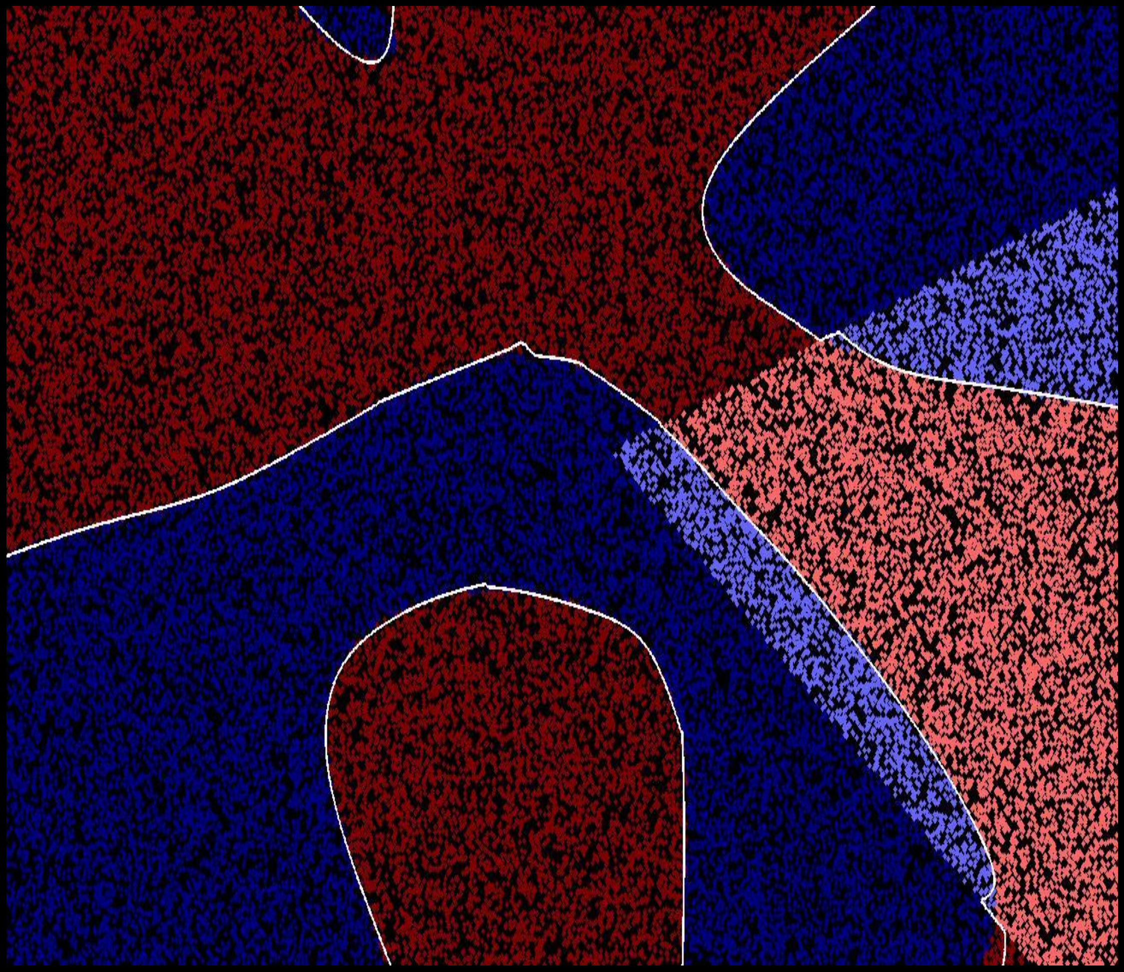
Adding One Node



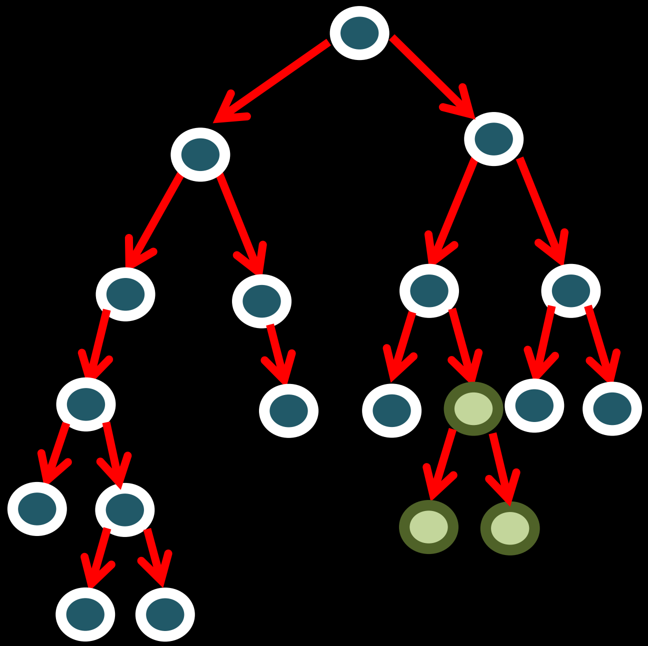


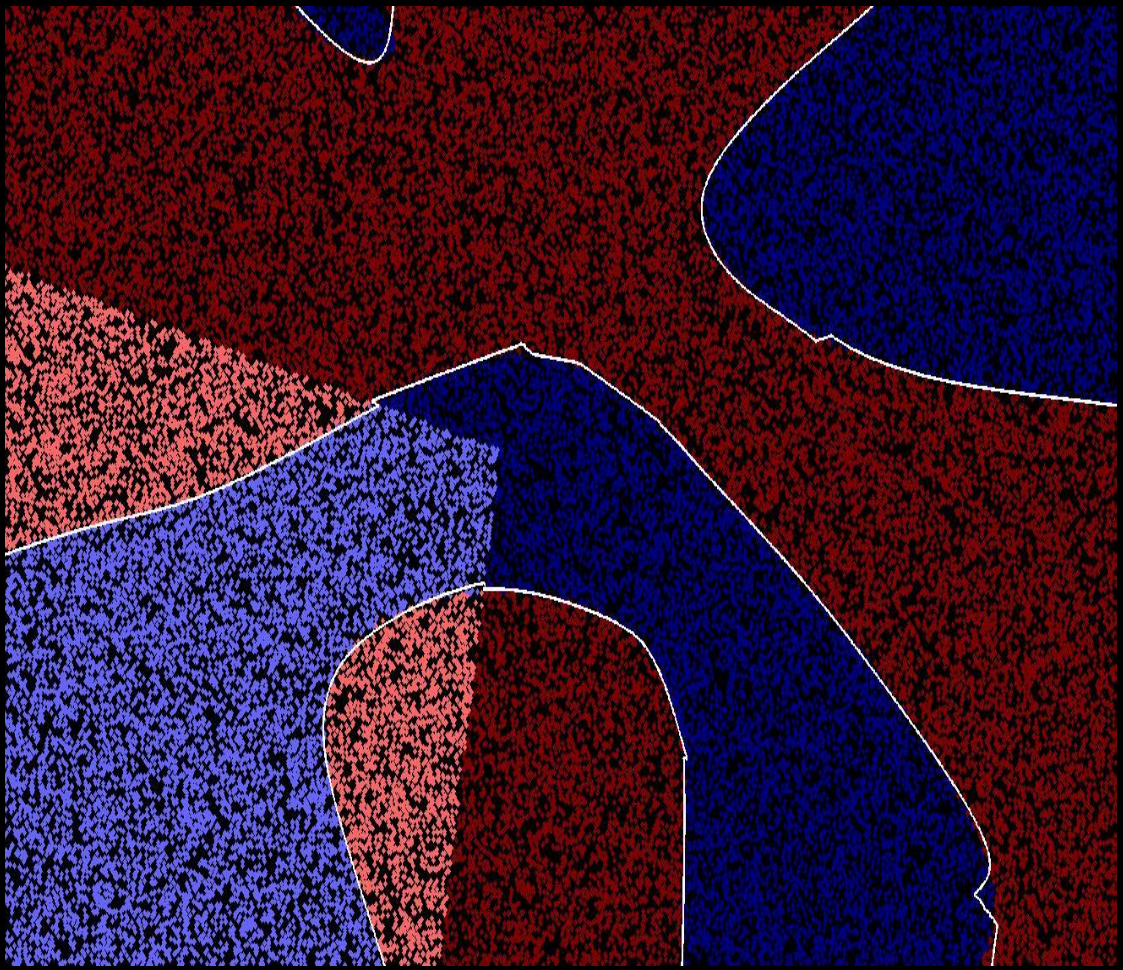
Adding Two Nodes



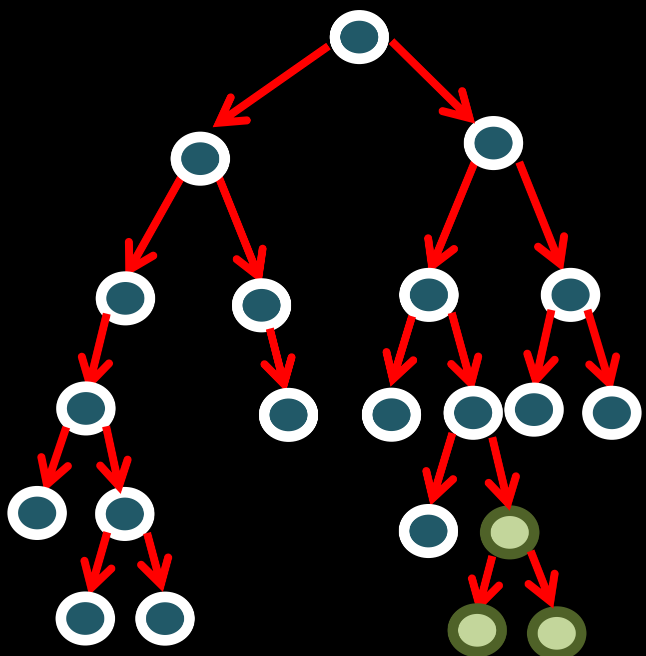


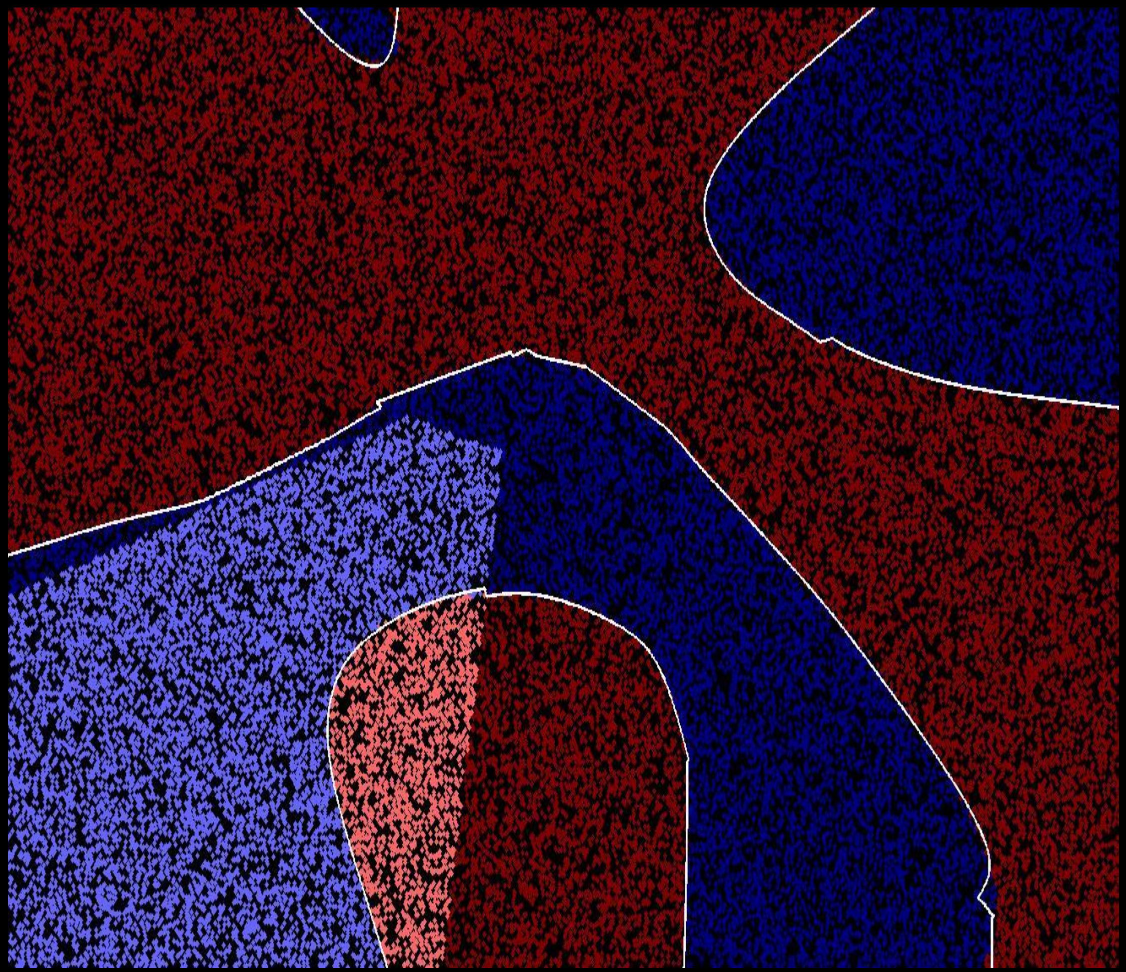
Adding Three Nodes



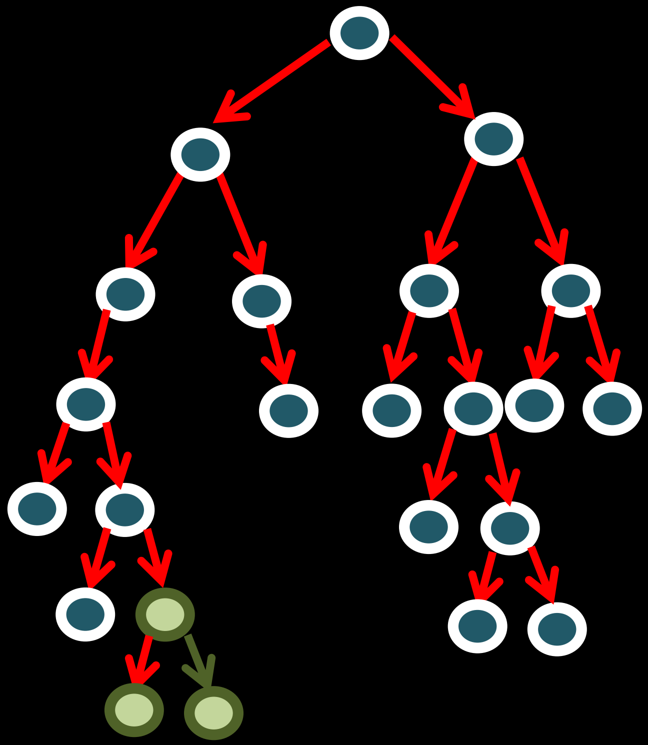


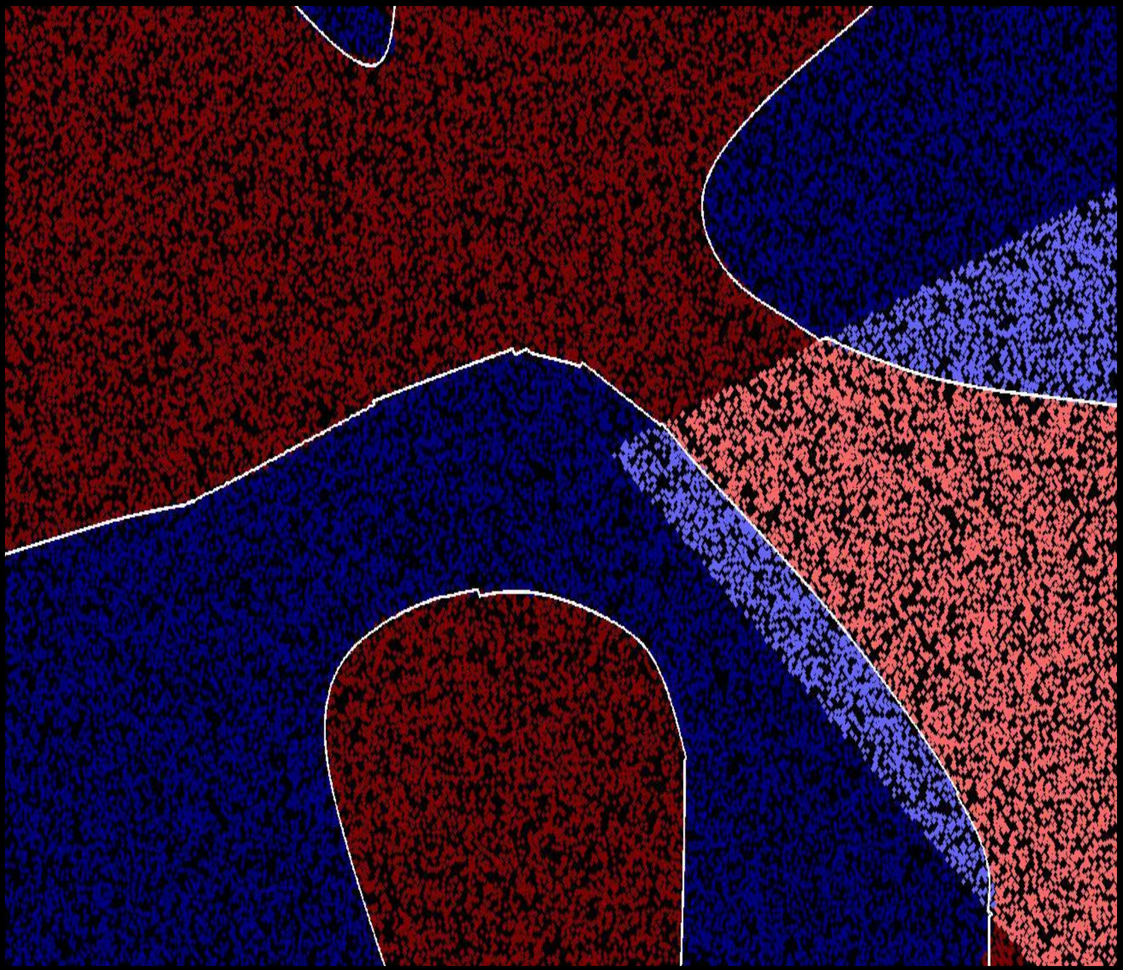
Adding Four Nodes



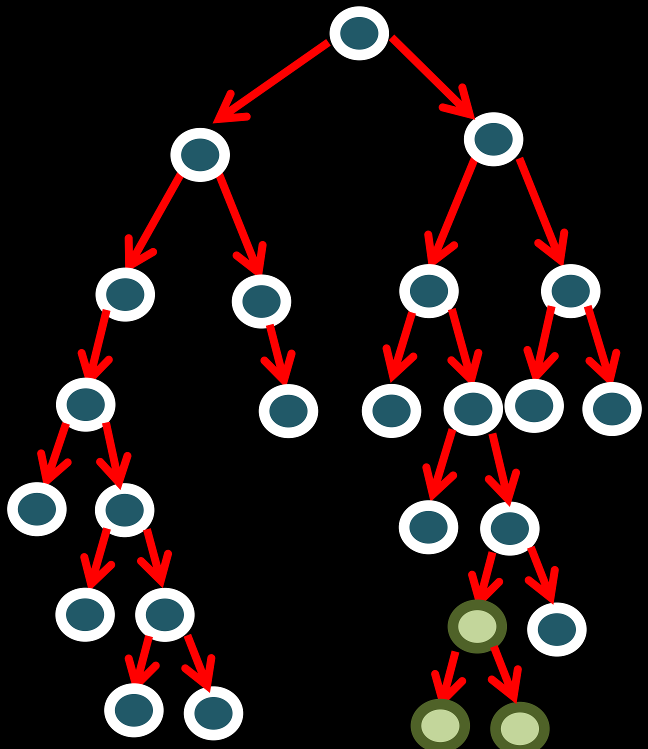


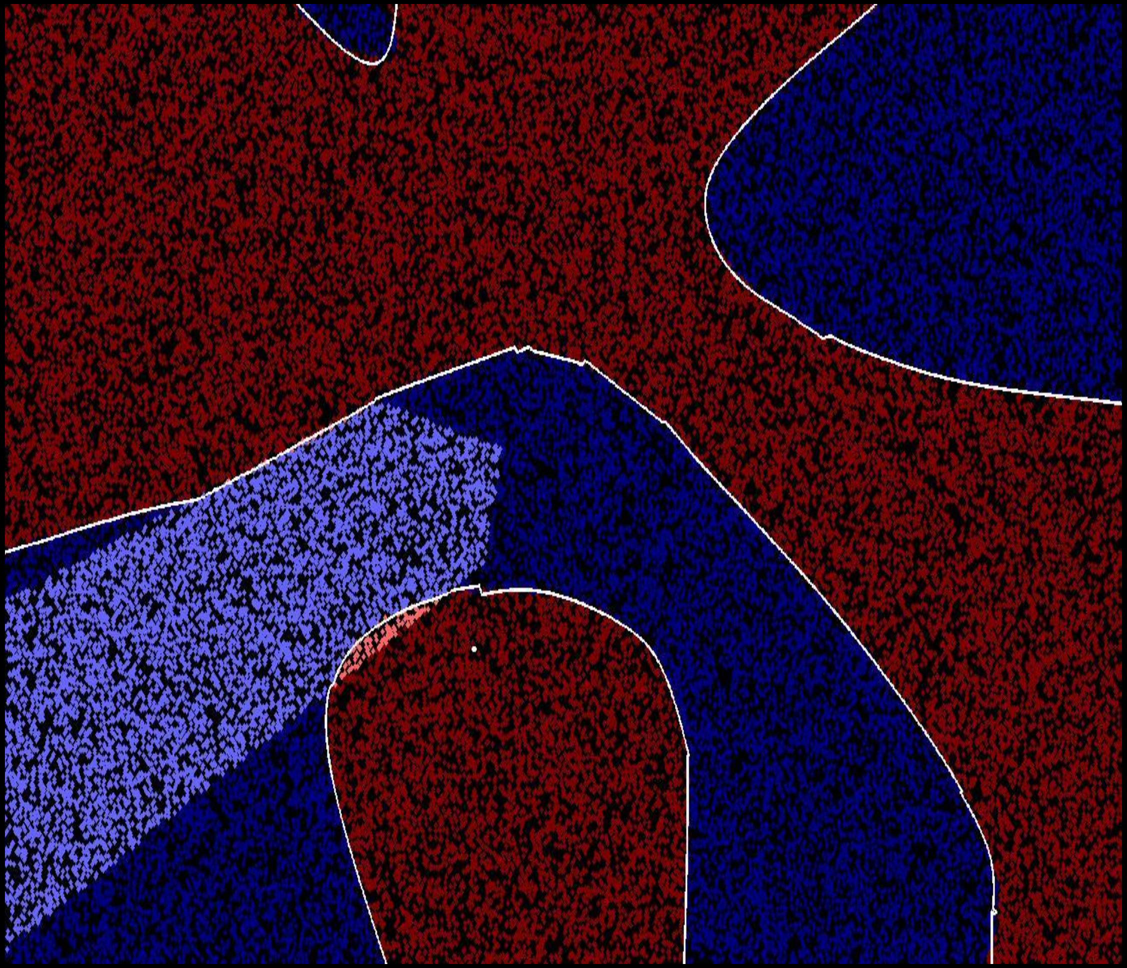
Adding Five Nodes



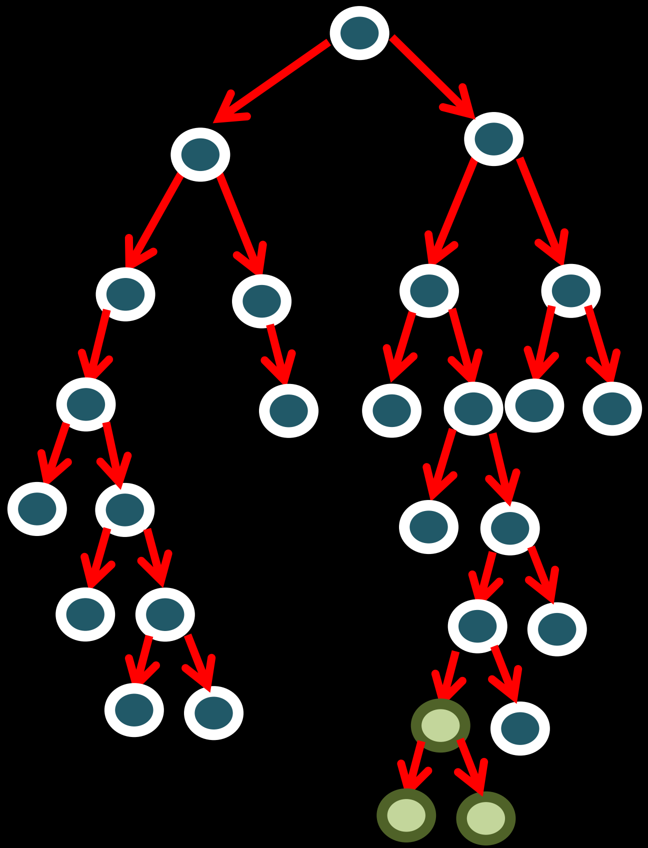


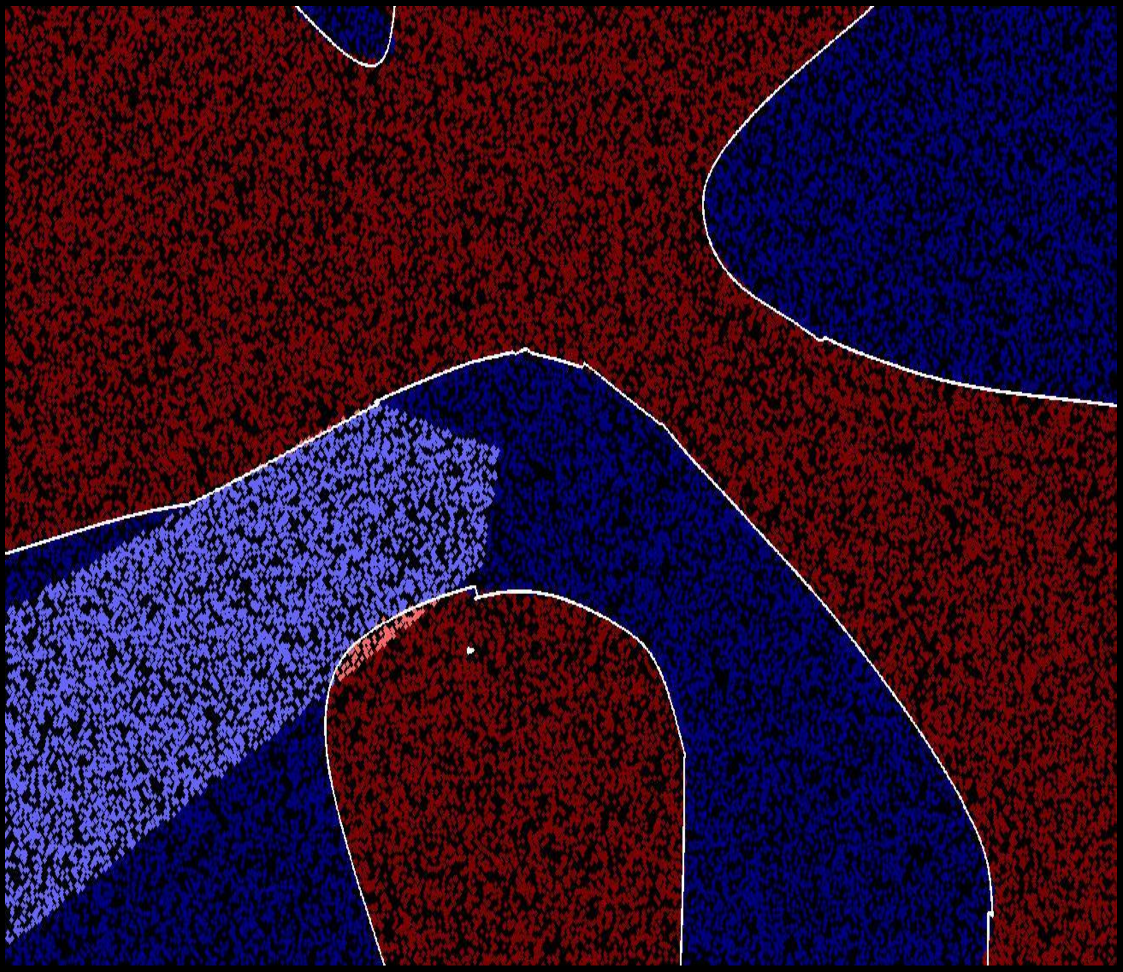
Adding Six Nodes



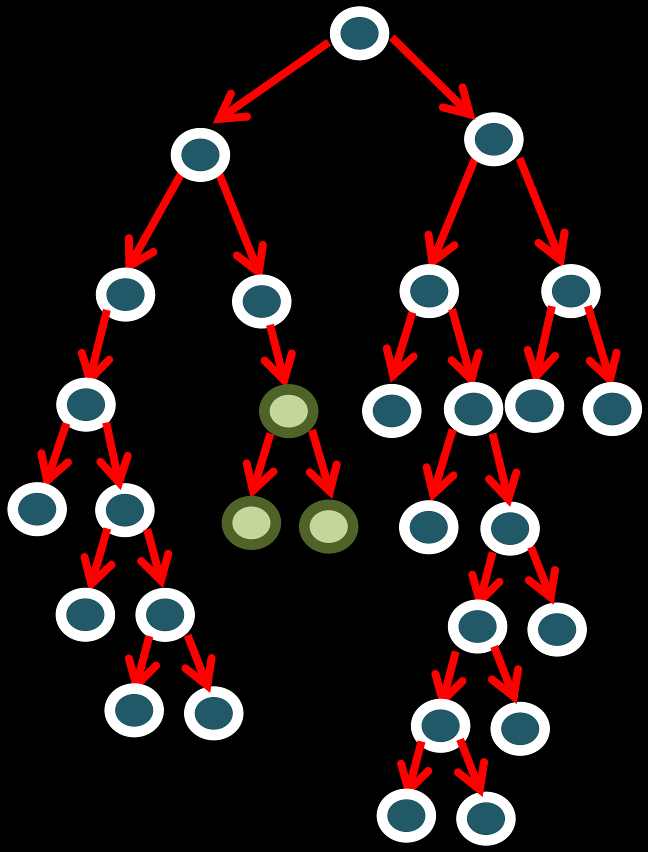


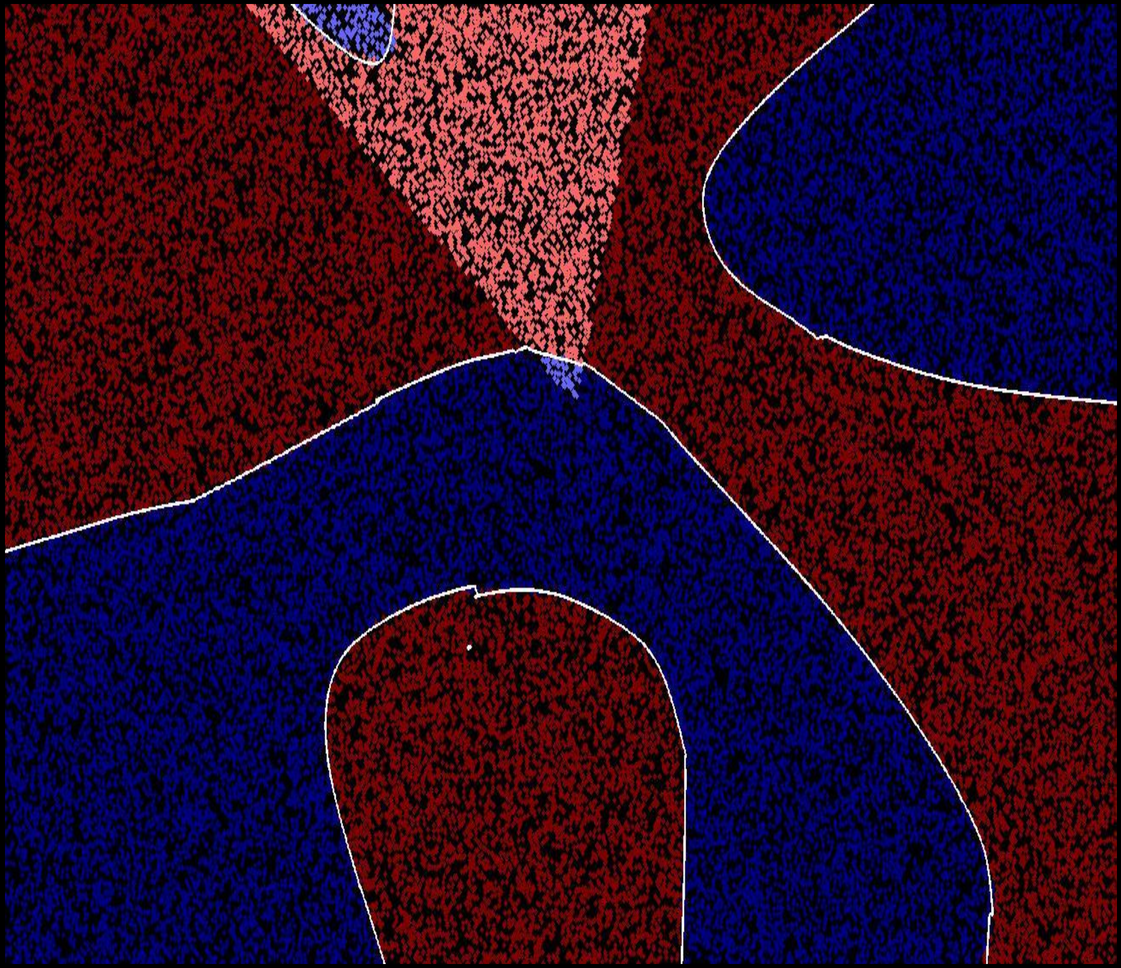
Adding Seven Nodes



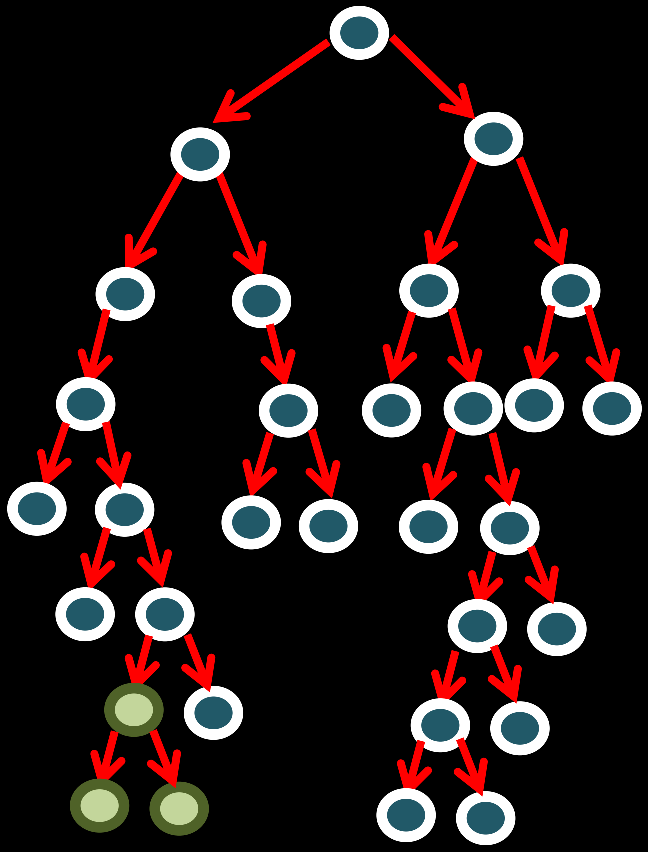


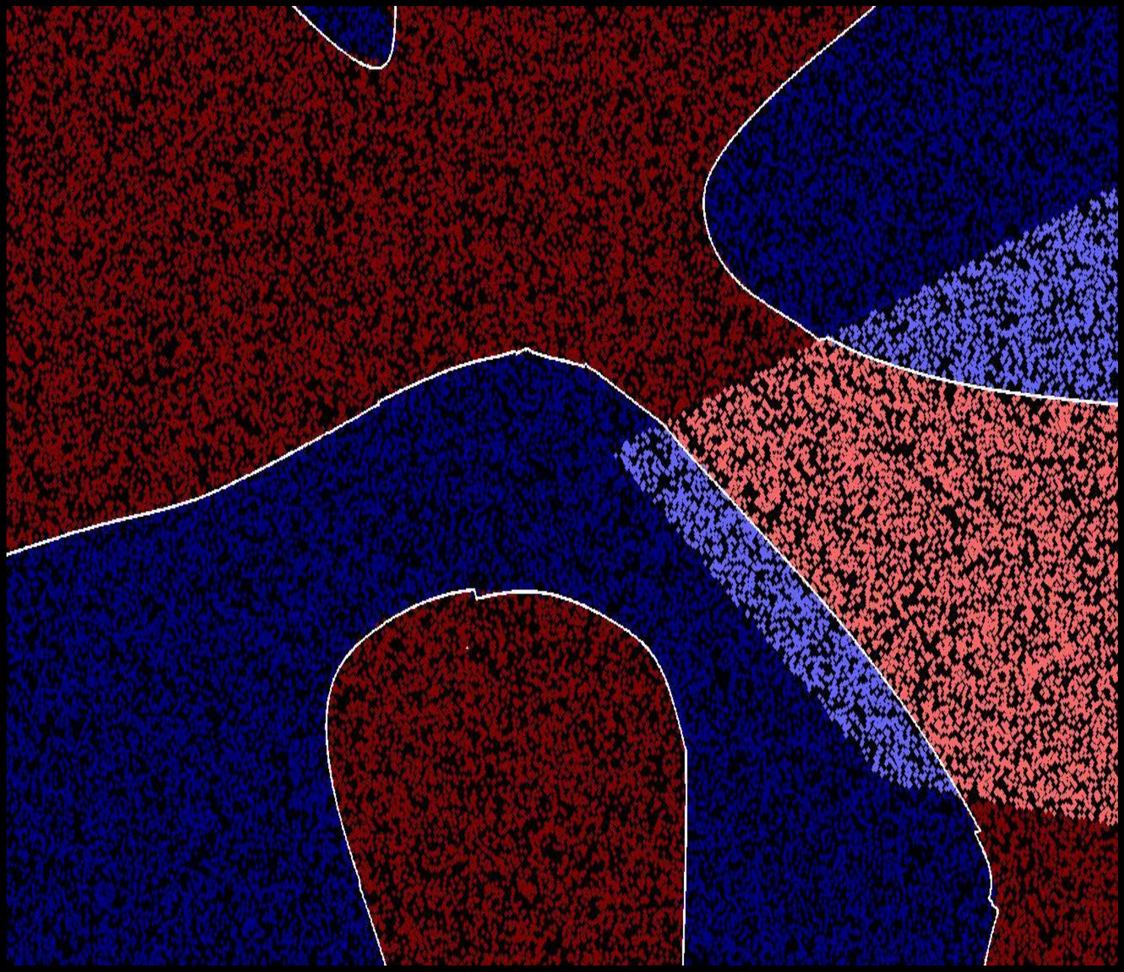
Adding Eight Nodes



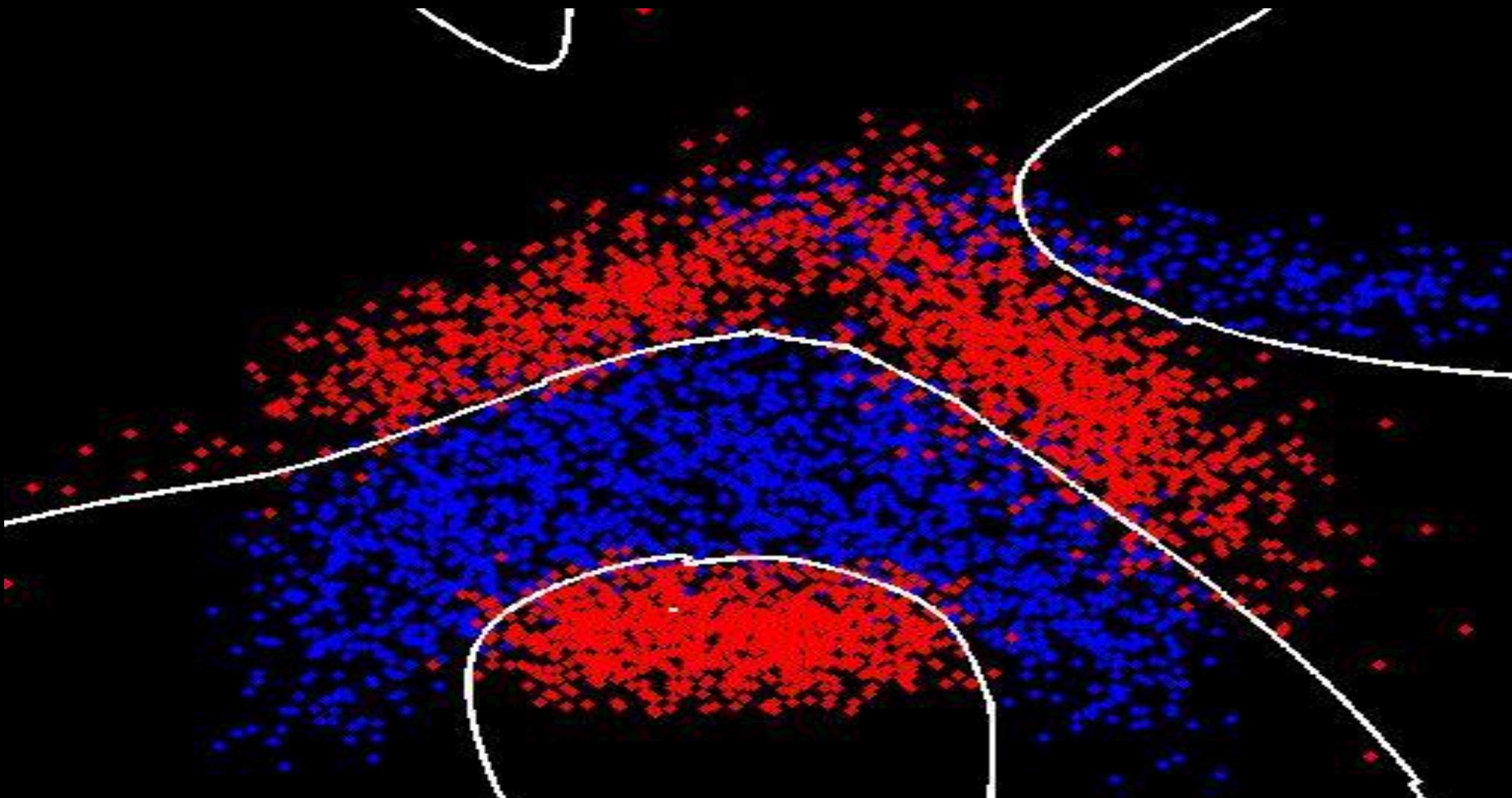


Adding Nine Nodes





LDKL's Final Decision Boundaries



Conclusions

- Publications and code
  - ICML 2013 paper
  - Code: http://research.microsoft.com/~manik/code/LDKL/download.html
- LDKL learns a local, deep composite kernel for efficient non-linear SVM prediction
- LDKL can be exponentially faster than the state-of-the-art
- Efficiency is important during both training and prediction

Acknowledgements

- Samy Bengio
- Purushottam Kar
- Prateek Jain
- Yann Lecun
- Vinod Nair
- Yashoteja Prabhu
- Nishal Shah