Non-linear learning in the presence of label noise

Ata Kabán

Joint work with Jakramate Bootkrajang.

School of Computer Science, the University of Birmingham

LABELNOISE'17
30th November - 1st December 2017

Learning from imperfect labels

- A supervised learning machine relies on a set of labelled data.
- There is no guarantee that the provided class labels are all correct:
  - Scale of labelling task.
  - Lack of information to determine the true label.
  - Subjective nature of human experts.
- Encountered in bioinformatics, image classification, text classification.
- More recently, in crowd-sourced data
  - Amazon's Mechanical Turk
  - Galaxy Zoo
- A flexible non-linear model can easily end up learning the label noise, and then fail on new data.
  - E.g. Adaboost is notoriously sensitive to label noise.
  - From [Long & Servedio, 2010], random mislabelling defeats all boosters.

Main types of approaches

- De-noising the data by pre-processing or ‘cleansing’
- Formulating a model of the label noise.
  - We will assume that label flipping occurs at random, and is independent of the input data.
  - This gives us a latent variable model where the true labels are latent variables.
  - E.g. [Raykar et al., 2012] uses EM to infer the true labels from a set of noisy labels per input point, and a logistic regression likelihood model.
  - In [Bootkrajang & Kabán, 2012] we only observe one noisy label for each point.
    - In these works, the classifier is linear.
    - In this talk we are interested in non-linear models.

From linear to nonlinear

- A linear model is restrictive – if the label noise is close to symmetric, it won't even affect it.

- First idea: Use the kernel trick!
- But wait... how do we tune its parameters without access to the true labels?

Robust kernel logistic regression (rKLR)

- Training set $\mathcal{D} = \{(\mathbf{x}_n, \tilde{y}_n)\}_{n=1}^{N}$, where $\mathbf{x}_n \in \mathbb{R}^m$ and $\tilde{y}_n \in \{0, 1\}$ noisy.
- Kernel logistic regression's non-linear decision function:

$$f(\mathbf{x}) = \sum_{n=1}^{N} w_n \kappa(\cdot, \mathbf{x}_n) \tag{1}$$

where $\kappa(\cdot, \cdot)$ is a positive definite reproducing kernel that defines an inner product in the feature space, and $\mathbf{w} = (w_1, ..., w_N)$ parameter vector.

- Define the probability of observing a label of $\tilde{y}_n$ as a mixture:

$$p(\tilde{y} = k|\kappa(\cdot, \mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{1} \omega_{jk} p(y = j|\kappa(\cdot, \mathbf{x}_n), \mathbf{w}) \tag{2}$$

where $\omega_{jk} \equiv p(\tilde{y} = k|y = j)$ form a transition table, Ω (flip matrix)

- The full set of parameters is $\Theta = \{\mathbf{w}, \Omega\}$.

- We estimate the parameters by maximising the log-likelihood:

$$\mathcal{L}(\Theta) = \sum_{n=1}^{N} \sum_{k=0}^{1} \mathbb{1}(\tilde{y}_n = k) \log \underbrace{p(\tilde{y}_n = k|\kappa(\cdot, \mathbf{x}_n), \Theta)}_{\tilde{P}_n^k} - \zeta \sum_{n=1}^{N} w_n^2 \qquad (3)$$

where:

- $\mathbb{1}(\cdot)$ is the Kronecker delta function,
- the last term is $L2$ regularisation,
- $\tilde{P}_n^k \equiv p(\tilde{y} = k|\kappa(\cdot, \mathbf{x}_n, \Theta)) = \sum_{j=0}^{1} \omega_{jk} p(y = j|\kappa(\cdot, \mathbf{x}_n), \mathbf{w})$,
- $p(y = 1|\kappa(\cdot, \mathbf{x}_n), \mathbf{w}) = \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) = \frac{1}{1+\exp(-\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n))}$

Parameter estimation for rKLR

- Update $\mathbf{w}$ e.g. by conjugate gradients, where the gradient is:

$$\mathbf{g} = \sum_{n=1}^{N} \Bigg[\left(\frac{\mathbb{1}(\tilde{y}_n = 1)(\omega_{11} - \omega_{01})}{\tilde{P}_n^1} + \frac{\mathbb{1}(\tilde{y}_n = 0)(\omega_{10} - \omega_{00})}{\tilde{P}_n^0} \right) \times \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n))(1 - \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n))) \times \kappa(\cdot, \mathbf{x}_n) \Bigg] - 2\zeta \sum_{n=1}^{N} w_n \quad (4)$$

- Update the entries of Ω using the method of Lagrangian multipliers to ensure that probabilities sum to 1:

$$\omega_{00} = \frac{\omega_{00} \sum_{n=1}^N \left[\frac{\mathbb{1}(\tilde{y}_n=0)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right]}{\omega_{00} \sum_{n=1}^N \left[\frac{\mathbb{1}(\tilde{y}_n=0)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right] + \omega_{01} \sum_{n=1}^N \left[\frac{\mathbb{1}(\tilde{y}_n=1)}{\tilde{P}_n^1} (1 - \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right]} \quad (5)$$

$$\omega_{11} = \frac{\omega_{11} \sum_{n=1}^N \left[\frac{\mathbb{1}(\tilde{y}_n=1)}{\tilde{P}_n^1} \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right]}{\omega_{10} \sum_{n=1}^N \left[\frac{\mathbb{1}(\tilde{y}_n=0)}{\tilde{P}_n^0} \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right] + \omega_{11} \sum_{n=1}^N \left[\frac{\mathbb{1}(\tilde{y}_n=1)}{\tilde{P}_n^1} \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right]} \quad (6)$$

rKLR Algorithm

```
Require: $\kappa$, $\Omega$
    Initialise $\mathbf{w} \leftarrow 0$
    while Iteration < MaxIteration do
        Update $\mathbf{w}$ using eq.(4)
        Update $\Omega$ using eq.(5) and eq.(6)
    end while
Ensure: Optimised weight vector, $\mathbf{w}$. Optimised $\Omega$.
```

Relation to EM can be shown:

- The expressions of the posterior probabilities $P_n \equiv p(y_n = 1|\mathbf{x}, \mathbf{w}, \tilde{y}_n)$ from the E-step are are plugged into the M-step updates.
- With one subtle difference in that in EM P_n are computed from parameter values from the previous M-step, whereas here all updates use the most recent values of the parameters.

Determining the kernel width – a multi-kernel learning (MKL) approach

- In MKL a combination of several kernels is learnt.
  - Previously used to combine heterogeneous data sources
  - Here we use it to avoid the need for kernel width selection – as CV would need the true labels
- We use a conic combination:

$$\kappa(\cdot,\cdot) = \sum_{i=1}^{S} \eta_i \kappa_i(\cdot,\cdot) : \qquad \eta_i \geq 0 : \forall i \tag{7}$$

- and in addition want $\eta = (\eta_1, \ldots \eta_S)$ be sparse – hence impose an exponential prior / regularisation term. The new objective:

$$\sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1) \log \tilde{P}_n^1 + \mathbb{1}(\tilde{y}_n = 0) \log \tilde{P}_n^0 - \zeta \sum_{i=1}^{N} w_i^2 - \sum_{i=1}^{S} \xi_i \eta_i \tag{8}$$

Determining the kernel width – a multi-kernel learning (MKL) approach

- To ensure $\eta_i \geq 0, \forall i$, re-parametrise $\eta_i = u_i^2$, and optimise for u_i (e.g. using conjugate gradients). The derivative of the objective, eq.(8), w.r.t u_i is:

$$\sum_{n=1}^{N}\Bigg[\left(\frac{\mathbb{1}(\tilde{y}_n = 1)(\omega_{11} - \omega_{01})}{\tilde{P}_n^1} + \frac{\mathbb{1}(\tilde{y}_n = 0)(\omega_{10} - \omega_{00})}{\tilde{P}_n^0}\right) \times \sigma(\mathbf{w}^T\kappa(\cdot,\mathbf{x}_n))(1 - \sigma(\mathbf{w}^T\kappa(\cdot,\mathbf{x}_n))) \times (\mathbf{w}^T\kappa_i(\cdot,\mathbf{x}_n))\Bigg] - 2\xi_i u_i \qquad (9)$$

  We then recover η_i by squaring the optimised u_i.
- Remains to determine the hyper-parameters ζ and $\xi_i, i = 1, ..., S$ - note that cross-validation is not practical unless a trusted validation set is available.

Determining the kernel width – a multi-kernel learning (MKL) approach + Bayesian regularisation = rMKLR

Instead, interpret the two regularisation terms as $-\log$ of a spherical Gaussian prior on $\mathbf{w}$, and independent exponential priors on η_i. Then we can put hyper-priors on ζ, ξ_i and integrate them out.

- Conditional prior: $p(\mathbf{w}|\zeta) = \mathcal{N}(0, 1/\zeta)$
- Hyper-prior: $p(\zeta|\beta) = \beta e^{-\beta\zeta}$
- Marginal prior: $p(\mathbf{w}) = \int_0^\infty p(\mathbf{w}|\zeta)p(\zeta)\mathrm{d}\zeta = \frac{\beta}{(2\pi)^{m/2}} \frac{\Gamma(\frac{m}{2}+1)}{(\frac{1}{2}\sum_{i=1}^m w_i^2+\beta)^{(m/2+1)}}$
- Replace $-\log$ of marginal prior with $-\log$ of the conditional prior. Gradient of this gives:

$$-\frac{\partial \log p(\mathbf{w})}{\partial \mathbf{w}} = \underbrace{\frac{\frac{m}{2}+1}{\frac{1}{2}\sum_{i=1}^m w_i^2 + \beta}}_{\zeta} \underbrace{\frac{\partial \sum_{i=1}^m w_i^2}{\partial \mathbf{w}}}_{-\frac{\partial \log p(\mathbf{w}|\zeta)}{\partial \mathbf{w}}}$$

Determining the kernel width – a multi-kernel learning (MKL) approach + Bayesian regularisation = rMKLR

Same approach for η

- Conditional prior: $p(\boldsymbol{\eta}|\mathcal{D}, \mathbf{w}, \boldsymbol{\xi}) \propto p(\mathcal{D}|\boldsymbol{\eta}) \prod_{i=1}^{S} p(\eta_i|\xi_i)$
- Hyper-prior: $p(\xi_i) = \psi e^{-\psi\xi_i}$; $\psi = 10^{-100}$
- Marginal prior: $p(\eta_i) = \int_0^\infty \xi_i e^{-\xi_i\eta_i} \cdot \psi e^{-\psi\xi_i} \mathrm{d}\xi_i = \psi \frac{\Gamma(2)}{(\eta_i+\psi)^2}$
- Replace $-\log$ of marginal prior with $-\log$ of the conditional prior. Gradient of this gives:

$$-\frac{\partial \log p(\eta_i)}{\partial u_i} = \underbrace{\frac{2}{(\eta_i + \psi)}}_{\xi_i} \quad \underbrace{\frac{\partial \eta_i}{\partial u_i}}_{-\frac{\partial \log p(\eta_i|u_i)}{\partial u_i}}$$

rMKLR Algorithm

```
Require: Set of predefined kernels $\kappa_{i=1:S}$, $\Omega$
  Initialise $\mathbf{w} \leftarrow 0$, $\boldsymbol{\eta} \leftarrow 1$, $\zeta \leftarrow 0$, $\boldsymbol{\xi} \leftarrow 0$
  while Iteration < MaxIteration do
    Update $\mathbf{w}$ using eq.(4)
    Update $\zeta$ by reading off from gradient of marginal prior
    Update $\eta_i$ by optimising $u_i$ using eq.(9) and set $\eta_i = u_i^2$
    Update $\xi$ by reading off from gradient of marginal prior
    Update $\Omega$ using eq.(5) and eq.(6)
  end while
Ensure: Optimised weight vector, $\mathbf{w}$. Optimised $\Omega$.
```

Experiments

- does rKLR improve classification accuracy of over KLR under label noise?
- assess the MKL approach to kernel width setting against:
    - CV (5 fold) with a trusted validation set (10% of data set)
- comparisons with:
    - robust Kernel Fisher Discriminants
    - Stochastic Programming for MKL
    - gold standard SVM

- 13 UCI data sets
    - noise injected 10% to 40%
    - 100 train/test splits except 20 on larger sets (Image, Splice)
    - 21 RBF kernels with widths $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$. Use this set also for selecting C in SVM
- Real applications
    - Recognising textual entailment
    - Image classification from cheaply acquired labels

Data sets

Data set	Training samples	Test samples	Pos. samples	Neg. samples	Dimensionality
Banana	400	4900	44.83%	55.17%	2
B.Cancer	200	77	29.28%	70.72%	9
Diabetes	468	300	34.90%	65.10%	8
German	700	300	30.00%	70.00%	20
Heart	170	100	44.44%	55.56%	13
Image	1300	1010	56.95%	43.05%	18
Ringnorm	400	7000	49.51%	50.49%	20
S.Flare	666	400	65.28%	34.72%	9
Splice	1000	2175	44.93%	55.07%	60
Thyroid	140	75	30.23%	69.77%	5
Titanic	150	2051	58.33%	41.67%	3
Twonorm	400	7000	50.04%	49.96%	20
Waveform	400	4600	32.94%	67.06%	21

Table: Characteristics of the UCI data sets used.

Illustrative experiment (1): Effect of 30% label noise

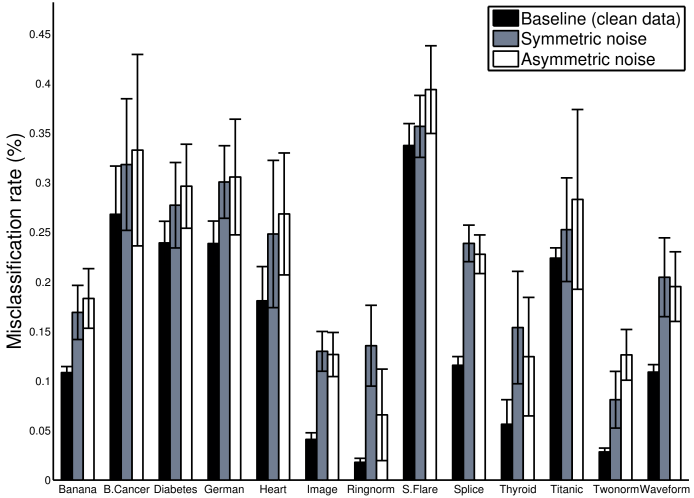


Figure: Effect of 30% symmetric and asymmetric noise to traditional KLR, compared against clean baseline. Trusted validation set used.

Illustrative experiment (2): rKLR vs. KLR with trusted validation set

Dataset	Noise level					
	10%			20%		
	KLR	rKLR	p-value	KLR	rKLR	p-value
Banana	**12.28 ± 1.49**	12.80 ± 2.10	0.03	14.58 ± 1.96	**12.91 ± 2.32**	$3.21e-22$
B.Cancer	**28.96 ± 5.62**	31.02 ± 6.20	$6.51e-4$	**30.19 ± 7.04**	32.16 ± 7.82	0.02
Diabetes	**24.86 ± 3.09**	26.00 ± 3.48	$3.05e-4$	26.17 ± 3.17	26.69 ± 4.28	0.79
German	**25.07 ± 3.23**	26.90 ± 3.46	$7.62e-8$	**26.64 ± 3.58**	27.78 ± 3.66	$3.27e-3$
Heart	19.84 ± 6.57	20.34 ± 5.73	0.17	22.31 ± 5.10	**20.63 ± 4.76**	$9.91e-4$
Image	6.58 ± 1.13	6.19 ± 1.52	0.12	8.06 ± 1.12	**6.80 ± 1.00**	$2.79e-6$
Ringnorm	4.51 ± 2.23	**3.11 ± 1.78**	$5.91e-13$	4.94 ± 3.23	**3.29 ± 1.86**	$1.87e-6$
S.Flare	34.45 ± 2.33	34.81 ± 2.83	0.24	35.87 ± 2.70	36.00 ± 2.92	0.96
Splice	14.90 ± 1.47	14.90 ± 1.69	0.89	17.33 ± 1.67	16.82 ± 1.63	0.14
Thyroid	9.25 ± 4.00	**7.76 ± 4.28**	$6.194e-5$	9.56 ± 4.55	**8.49 ± 4.60**	$9.86e-3$
Titanic	22.85 ± 1.33	22.88 ± 1.90	0.63	23.57 ± 2.55	23.30 ± 2.39	0.16
Twonorm	4.69 ± 1.16	**3.79 ± 0.78**	$1.51e-19$	7.82 ± 1.88	**4.38 ± 1.20**	$5.33e-54$
Waveform	12.91 ± 1.52	**12.21 ± 1.15**	$2.22e-6$	15.04 ± 2.18	**12.81 ± 1.54**	$1.05e-33$

Dataset	Noise level					
	30%			40%		
	KLR	rKLR	p-value	KLR	rKLR	p-value
Banana	17.60 ± 2.95	**16.13 ± 3.99**	$7.38e-7$	25.63 ± 6.01	**23.08 ± 10.49**	$1.85e-5$
B.Cancer	32.54 ± 8.29	32.92 ± 9.26	0.87	36.89 ± 10.39	35.52 ± 10.36	0.15
Diabetes	28.67 ± 4.37	**27.42 ± 4.56**	$2.60e-4$	33.77 ± 6.19	**31.14 ± 7.03**	$7.70e-6$
German	30.30 ± 4.86	**28.81 ± 4.69**	$9.78e-4$	33.86 ± 8.42	**30.11 ± 4.69**	$2.19e-4$
Heart	25.82 ± 6.87	26.64 ± 8.15	0.62	34.99 ± 8.76	**30.98 ± 11.65**	$2.34e-5$
Image	12.82 ± 2.10	**10.45 ± 3.21**	$1.20e-3$	20.29 ± 3.78	**15.98 ± 7.24**	$8.48e-3$
Ringnorm	10.06 ± 5.57	9.57 ± 6.00	0.43	16.92 ± 8.78	15.78 ± 9.67	0.11
S.Flare	37.51 ± 4.25	36.82 ± 3.63	0.13	41.04 ± 4.71	**38.61 ± 4.37**	$1.50e-7$
Splice	23.31 ± 1.95	**21.20 ± 4.06**	0.03	31.20 ± 3.85	**26.74 ± 8.49**	0.04
Thyroid	13.91 ± 5.99	13.76 ± 8.10	0.24	22.44 ± 11.01	**19.16 ± 13.41**	$8.82e-5$
Titanic	26.77 ± 7.54	**25.19 ± 5.55**	0.03	34.64 ± 12.49	**29.47 ± 11.39**	$4.18e-9$

MKL with Bayesian regularisation vs. CV on clean data

Data set	Cross validated rKLR	rMKLR	p-value
Banana	10.96 ± 0.81	10.72 ± 0.52	0.06
B.Cancer	29.94 ± 4.65	**27.73 ± 4.19**	$9.17e-4$
Diabetes	24.53 ± 2.21	24.24 ± 1.85	0.47
German	25.38 ± 2.63	**23.52 ± 2.26**	$2.09e-7$
Heart	18.63 ± 4.00	**16.30 ± 3.39**	$7.48e-5$
Image	**3.73 ± 0.71**	5.65 ± 0.96	$2.78e-6$
Ringnorm	1.80 ± 0.43	**1.48 ± 0.10**	$4.16e-12$
S.Flare	**33.50 ± 2.15**	34.33 ± 1.75	$1.05e-3$
Splice	**11.47 ± 0.82**	13.30 ± 1.13	$7.52e-6$
Thyroid	5.96 ± 2.78	5.91 ± 2.70	0.95
Titanic	**22.26 ± 0.94**	22.73 ± 0.83	$4.28e-5$
Twonorm	2.86 ± 0.36	**2.47 ± 0.16**	$4.58e-17$
Waveform	10.78 ± 0.85	**10.58 ± 0.45**	0.03

Table: Comparison between standard cross-validation and MKL with Bayesian regularisation technique on clean datasets.

MKL with Bayesian regularisation vs. CV: Computation time

Dataset	CPU time (seconds)		Dataset	CPU time (seconds)	
	rKLR	rMKLR		rKLR	rMKLR
Banana	48.44 ± 6.13	**10.16 ± 0.28**	S.Flare	1033.25 ± 66.55	**26.73 ± 0.74**
B.Cancer	26.61 ± 0.90	**2.96 ± 0.25**	Splice	245.61 ± 1.68	**64.12 ± 1.36**
Diabetes	151.78 ± 44.26	**14.84 ± 0.42**	Thyroid	23.04 ± 1.25	**1.64 ± 0.29**
German	192.84 ± 54.27	**31.63 ± 3.34**	Titanic	15.81 ± 1.74	**1.68 ± 0.12**
Heart	24.23 ± 0.93	**2.15 ± 0.09**	Twonorm	49.19 ± 0.81	**10.75 ± 0.36**
Image	1218.50 ± 451.54	**106.99 ± 1.41**	Waveform	49.69 ± 1.03	**10.07 ± 1.14**
Ringnorm	51.21 ± 0.78	**10.54 ± 0.34**			

Table: Running times of one training/testing split, on a 2.67GHz Intel Core i5 CPU averaged over 10 random splits. The MKL (rMKLR) is 5 to 10 times faster than the traditional CV approach.

Sensitivity to choice of number of kernels

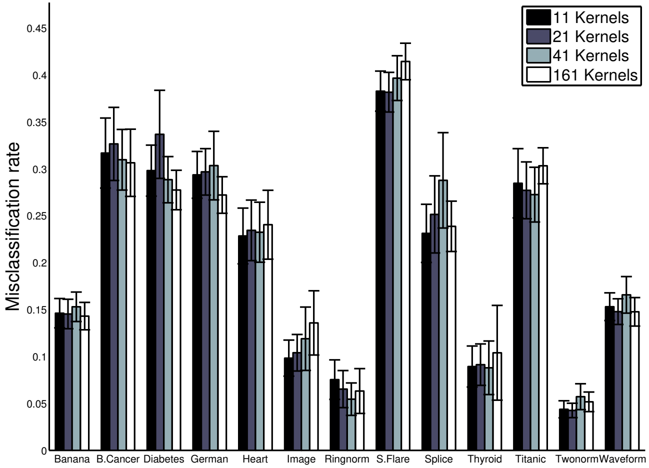


Figure: Comparison of the different kernel set size at 30% level, averaged over 20 random runs.

Comparisons: rMKLR vs. rKFD vs. SVM

Dataset	10% Noise			
	rKFD	SVM	rMKLR	p-value
Banana	12.39 ± 1.13	11.55 ± 1.07	11.39 ± 0.79	0.35
B.Cancer	28.71 ± 4.81	27.90 ± 5.05	27.93 ± 4.50	0.61
Diabetes	27.15 ± 2.51	24.21 ± 2.07	24.56 ± 2.00	0.12
German	26.93 ± 2.64	24.73 ± 2.57	**24.12 ± 2.38**	1.58e-2
Heart	18.96 ± 4.10	17.60 ± 3.92	17.27 ± 3.48	0.37
Image	5.25 ± 0.88	**4.95 ± 0.85**	6.09 ± 1.19	1.21e-5
Ringnorm	2.32 ± 0.44	**1.84 ± 0.49**	2.20 ± 0.48	5.16e-19
S.Flare	35.37 ± 1.91	**34.25 ± 2.24**	34.93 ± 1.96	1.34e-3
Splice	15.09 ± 1.47	**13.12 ± 1.14**	15.11 ± 1.80	6.54e-8
Thyroid	7.07 ± 3.96	6.03 ± 3.13	6.15 ± 2.75	0.31
Titanic	24.22 ± 2.23	**22.88 ± 1.27**	23.00 ± 1.12	4.04e-3
Twonorm	**2.61 ± 0.29**	2.88 ± 0.52	2.91 ± 0.36	1.67e-20
Waveform	12.82 ± 1.40	11.12 ± 1.06	10.93 ± 0.76	0.27
Dataset	30% Noise			
	rKFD	SVM	rMKLR	p-value
Banana	20.42 ± 6.07	17.63 ± 5.21	**14.92 ± 2.83**	1.35e-10
B.Cancer	33.50 ± 8.21	32.95 ± 8.39	32.36 ± 8.98	0.31
Diabetes	34.47 ± 4.77	29.60 ± 3.94	**26.87 ± 3.75**	2.30e-12
German	32.34 ± 4.56	29.80 ± 4.11	**27.75 ± 4.68**	2.60e-8
Heart	26.49 ± 9.18	25.31 ± 8.21	**23.49 ± 8.69**	1.66e-3
Image	12.41 ± 3.00	10.24 ± 2.33	10.31 ± 3.12	0.75
Ringnorm	6.88 ± 2.33	**3.24 ± 1.95**	4.51 ± 2.43	5.02e-10
S.Flare	38.51 ± 4.12	38.65 ± 4.21	**37.07 ± 3.77**	9.46e-5
Splice	29.89 ± 5.52	**21.46 ± 2.34**	26.98 ± 6.61	1.39e-4
Thyroid	15.93 ± 8.76	12.65 ± 7.99	**8.87 ± 8.89**	1.40e-12
Titanic	29.25 ± 8.86	**27.80 ± 8.36**	28.12 ± 7.41	3.38e-2
Twonorm	**3.08 ± 1.48**	5.38 ± 2.57	4.42 ± 1.77	9.62e-38
Waveform	19.75 ± 3.38	16.40 ± 3.33	**14.15 ± 2.39**	9.54e-15

Comparisons: rMKLR vs. StPMKLR

Data set	# of Examples	Dimensionality	# of Kernels
Ionosphere	351	34	350
Heart	270	13	140
Australia	690	14	150

Table: Data sets used in this comparison

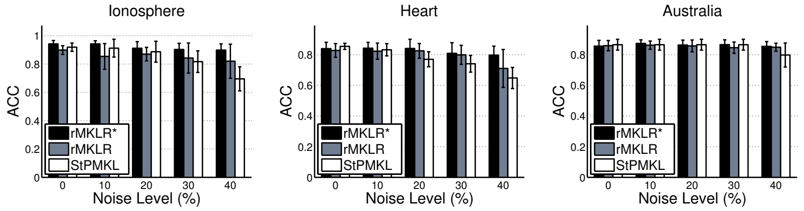


Figure: Comparison of classification accuracy (ACC) with noise level ranging from 0% to 40%.

Real applications: Textual entailment recognition

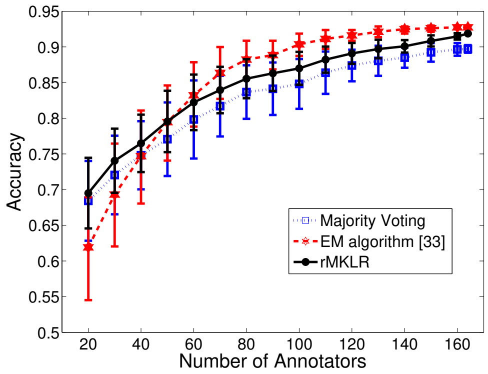


Figure: Data from PASCAL2 competition: 800 sentence pairs, 164 annotators labelling on average 53 of the pairs. rMKLR has advantage when number of annotators is small.

Image classification from cheaply acquired labels

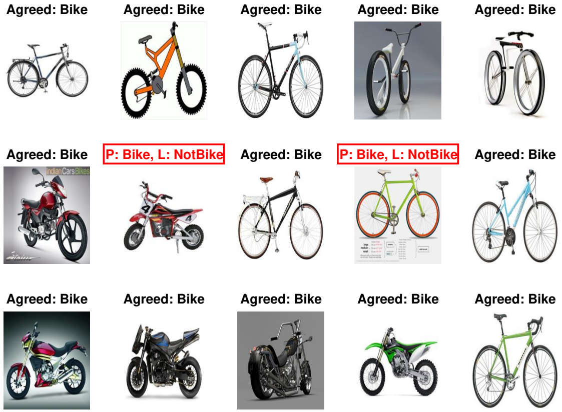


Figure: Examples of positive class (‘Bike’) predictions sorted by their posterior probability. In disagreement cases, P = classifier, L = labels from Google.

Image classification from cheaply acquired labels

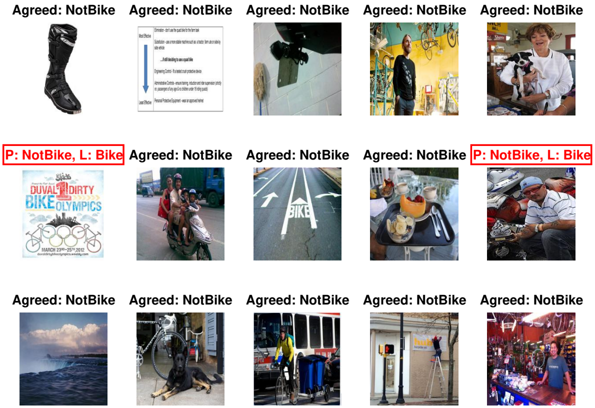


Figure: Examples of negative class (‘NotBike’) predictions sorted by their posterior probability. In disagreement cases, P = classifier, L = labels from Google.

Image classification from cheaply acquired labels

Classifier	rLR	KLR	rMKLR
Error rate	18.17% $\pm$ 0.02	21.44% $\pm$ 0.03	**14.19%** $\pm$ **0.02**

Table: Comparative results between rMKLR, KLR and linear rLR on the noisy label image classification task. The proposed rMKLR is the best performer. Interestingly, linear rLR also outperforms the traditional KLR.

Robust boosting

From boosting to robust boosting (rBoost)

Various ways exist to work around the problem

- ORBoost [Karmaker & Kwek, 2006]
  - Removes the difficult points, i.e. with the points very high weights, from the training set, using a predefined threshold.
- Modest-AdaBoost [Vezhnevets et al., 2005]
  - Penalises the ensemble when it makes a correct prediction on previously correctly predicted instances.
- Robustboost algorithm [Freund, 2009]
  - Optimises a non-convex potential function instead of the traditional exponential loss function.
  - Early stopping + a mechanism to give up on a point that is too far on the wrong side of decision boundary.
- BB algorithm [Krieger et al., 2001]
  - Combines bagging with boosting to average out the effect of wrongly labelled data.

How to robustify boosting? Two ideas

- Idea 1: Employ robust logistic regression [Bootkrajang & Kaban, 2012] as base learner of the existing AdaBoost algorithm.
    - Idea 1 is easy: Plug the previously presented robust logistic regression (linear kernel) as a base learner in Adaboost.
- Idea 2: Create a new robust boosting algorithm ('rBoost') by explicitly modelling the label noise using a convex combination of two exponential losses as the objective function.
    - The next slide(s) detail the approach for Idea 2.

Our new loss function for rBoost

- The objective (loss) function of traditional Adaboost:

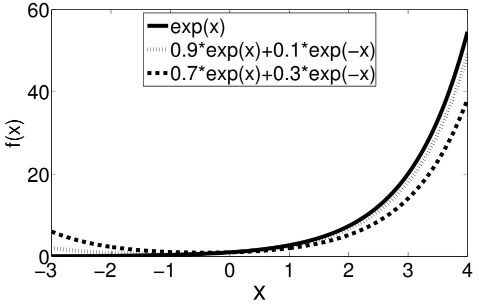
$$\sum_{i=1}^{n} \mathbb{1}(\tilde{y}_i = 1)e^{-H(\mathbf{x}_i)} + \mathbb{1}(\tilde{y}_i = -1)e^{H(\mathbf{x}_i)} \tag{10}$$

- A new loss function for rBoost:

$$\begin{aligned}\sum_{i=1}^{n} \ \mathbb{1}(\tilde{y}_i = 1)\left\{\gamma_{00}\, e^{-H(\mathbf{x}_i)} + \gamma_{01}\, e^{H(\mathbf{x}_i)}\right\} \\ + \mathbb{1}(\tilde{y}_i = -1)\left\{\gamma_{11}\, e^{H(\mathbf{x}_i)} + \gamma_{10}\, e^{-H(\mathbf{x}_i)}\right\}\end{aligned} \tag{11}$$

- Here, $\gamma_{jk} = p(\tilde{y} = k | y = j)$ are probabilistic factors representing uncertainties in labels. $\tilde{y}$ denotes noisy label while y denotes true label.

Illustration of the new loss



- For example, $\gamma_{01} = 0.3$ and $\gamma_{10} = 0$ indicates the situation where labels in the negative class (or class 0) are all correct but labels in the positive class are contaminated.
- The new loss accounts for this by adjusting the loss for the positive class (class 1) to: $0.7 * e^{-H} + 0.3 * e^{H}$.
- Meanwhile the loss of the negative class (class 0), which is $e^{H} + 0 * e^{-H} = e^{H}$, reduces to traditional boosting.

Adding a new base learner

- At iteration t, minimising the loss in eq.(11) w.r.t the new $h_t(\mathbf{x})$ is equivalent to minimising the following:

$$\begin{aligned}\arg\min_{h,\alpha}\ 2\sinh(\alpha)\sum_{i=1}^{n}\left\{w_i\mathbb{1}(h(\mathbf{x}_i)\neq\tilde{y}_i)\right\}\\ +e^{-\alpha}\sum_{i=1}^{n}\left\{\mathbb{1}(\tilde{y}_i=1)w_{00}+\mathbb{1}(\tilde{y}_i=-1)w_{11}\right\}\\ +e^{\alpha}\sum_{i=1}^{n}\left\{\mathbb{1}(\tilde{y}_i=1)w_{01}+\mathbb{1}(\tilde{y}_i=-1)w_{10}\right\}\end{aligned}\tag{12}$$

where

$$w_i = \begin{cases}(w_{00}-w_{01}), & \text{if } \tilde{y}_i=+1\\ (w_{11}-w_{10}), & \text{if } \tilde{y}_i=-1\end{cases}\tag{13}$$

$$w_{jk} = \gamma_{jk}\cdot d_{jk}\tag{14}$$

and d_{jk} are the exp terms: If $\tilde{y}=1, d_{00}=e^{-H(x)}, d_{01}=e^{H(x)}$; if $\tilde{y}=-1, d_{11}=e^{H(x)}, d_{10}=e^{-H(x)}$.

Determining base learner's weight

- No closed form update for α_t.
- Take derivative of eq.(12) w.r.t α_t, equate it to zero to get

$$
\begin{aligned}
&2\cosh(\alpha)\sum_{i=1}^{n}\left\{w_i \mathbb{1}(h(\mathbf{x}_i)\neq \tilde{y}_i)\right\}\\
&-e^{-\alpha}\sum_{i=1}^{n}\left\{\mathbb{1}(\tilde{y}_i=1)w_{00}+\mathbb{1}(\tilde{y}_i=-1)w_{11}\right\}\\
&+e^{\alpha}\sum_{i=1}^{n}\left\{\mathbb{1}(\tilde{y}_i=1)w_{01}+\mathbb{1}(\tilde{y}_i=-1)w_{10}\right\}=0
\end{aligned}
\tag{15}
$$

- The above gradient is used to optimise α_t numerically.

Updating the sample weights

- It follows that the update for w_{00} can be written as:

$$\begin{aligned} w_{00}^{t+1} &= \gamma_{00} e^{-\tilde{y}_i(H+\alpha h)} \\ &= \gamma_{00} e^{-\tilde{y}_i H} \cdot e^{-\tilde{y}_i \alpha h} \\ &= \gamma_{00} d_{00}^{t} \cdot e^{\alpha(2\mathbb{1}(h(\mathbf{x}_i)\neq\tilde{y}_i)-1)} \\ &= \gamma_{00} d_{00}^{t} \cdot e^{2\alpha\mathbb{1}(h(\mathbf{x}_i)\neq 1)} \cdot e^{-\alpha} \\ &\propto \gamma_{00} d_{00}^{t} \cdot e^{2\alpha\mathbb{1}(h(\mathbf{x}_i)\neq 1)} \end{aligned} \tag{16}$$

using, $-\tilde{y}h = 2\mathbb{1}(h(\mathbf{x}) \neq \tilde{y}) - 1$.

- Similarly for the rest of the weight vectors we get:

$$w_{01}^{t+1} = \gamma_{01} d_{01}^{t} \cdot e^{2\alpha\mathbb{1}(h(\mathbf{x}_i)\neq -1)} \tag{17}$$

$$w_{11}^{t+1} = \gamma_{11} d_{11}^{t} \cdot e^{2\alpha\mathbb{1}(h(\mathbf{x}_i)\neq -1)} \tag{18}$$

$$w_{01}^{t+1} = \gamma_{10} d_{10}^{t} \cdot e^{2\alpha\mathbb{1}(h(\mathbf{x}_i)\neq 1)} \tag{19}$$

Updating label flipping probabilities

- We convert the output of boosting, H, into a probability using Platt's calibration: $p(y = 1|x, H) = 1/(1 + \exp(AH + B))$ [Platt, 1999]
- Define $P(x) = p(y = 1|x, H)$ and $\bar{P}(x) = 1 - P(x)$, and we can then estimate the gamma from the binomial log-loss.

$$-\sum_{i=1}^{n} \mathbb{1}(\tilde{y}_i = 1) \log \left\{ \gamma_{11} P(\mathbf{x}_i) + \gamma_{01} \bar{P}(\mathbf{x}_i) \right\} + \mathbb{1}(\tilde{y}_i = -1) \log \left\{ \gamma_{00} \bar{P}(\mathbf{x}_i) + \gamma_{10} P(\mathbf{x}_i) \right\} \quad (20)$$

- the multiplicative updates for γ_{jk} are found to be:

$$\gamma_{10} = \frac{g_{10}}{g_{10} + g_{11}}, \quad \gamma_{11} = \frac{g_{11}}{g_{10} + g_{11}} \quad (21)$$

$$\gamma_{00} = \frac{g_{00}}{g_{00} + g_{01}}, \quad \gamma_{01} = \frac{g_{01}}{g_{00} + g_{01}} \quad (22)$$

where

$$g_{11} = \gamma_{11} \sum_{i=1}^{n} \left(\frac{\mathbb{1}(\tilde{y}_i = 1) P_i}{\gamma_{11} P_i + \gamma_{01} \bar{P}_i} \right), \quad g_{10} = \gamma_{10} \sum_{i=1}^{n} \left(\frac{\mathbb{1}(\tilde{y}_i = -1) P_i}{\gamma_{10} P_i + \gamma_{00} \bar{P}_i} \right)$$

$$g_{01} = \gamma_{01} \sum_{i=1}^{n} \left(\frac{\mathbb{1}(\tilde{y}_i = 1) \bar{P}_i}{\gamma_{11} P_i + \gamma_{01} \bar{P}_i} \right), \quad g_{00} = \gamma_{00} \sum_{i=1}^{n} \left(\frac{\mathbb{1}(\tilde{y}_i = -1) \bar{P}_i}{\gamma_{10} P_i + \gamma_{00} \bar{P}_i} \right)$$

rBoost Algorithm

```
Input: data $\{\mathbf{x}, \tilde{y}\}^n$, boosting round $T$
Initialize $w_{jk} = \gamma_{jk}$
for $t = 1$ to $T$ do
    (1)$h_t$ = a base learner trained from a data weighted by $w_i$.
    (2)Calculate the error of the base learner w.r.t $w_i$ defined in eq.(13)
        $\epsilon_t = \sum_{i=1}^{n} w_i \mathbb{1}(\tilde{y}_i \neq h_t(\mathbf{x}_i))$
    (3)Optimise $\alpha_t$ numerically from eq.(15)
    (4)Update $w_{jk}$ according to eq.(16)–(19).
    (5)Calculate $p(y = 1|\mathbf{x}, H)$ using Platt's method.
    (6)Update $\gamma_{jk}$ using eq.(21)–(22).
end for
Output the final classifier $sign(\sum_{t=1}^{T} \alpha_t h_t)$.
```

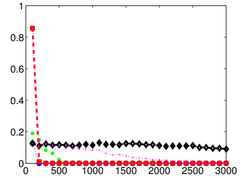
Datasets

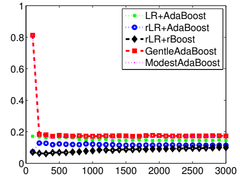
Table: Characteristics of the datasets used.

Dataset	# of positive samples	# of negative samples	dimensionality
Banana	2375(45%)	2924(55%)	2
Diabetes	268(35%)	500(65%)	8
Heart	120(44%)	150(56%)	13
Image	1188(57%)	898(43%)	18
Titanic	14(58%)	10(42%)	3
Twonorm	3703(50%)	3697(50%)	20
Waveform	1647(33%)	3353(67%)	21

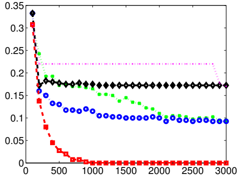
Result: robust base learner (rLR) + AdaBoost

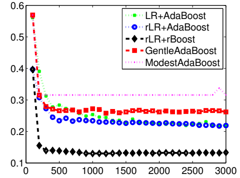
- Twonorm: training error [left] and test error [right] vs boosting iterations





- Banana: training error [left] and test error [right] vs boosting iterations





- An ensemble of robust base learners
  - is still susceptible to label noise.
  - but converges faster, which could be useful in low noise cases.

Result: rBoost at 30% label noise

Table: Mean classification errors and standard deviations at 30% symmetric noise.

Dataset	AdaBoost		rBoost-Fixed gamma		rBoost		Gentle	Modest
	LR	rLR	LR	rLR	LR	rLR	Boost	Boost
Banana	18.71±3.1	14.73±3.0	18.14±1.7	14.47±2.1	18.05±1.6	14.94±2.7	20.62±1.6	24.69±2.5
Diabetes	25.37±2.3	27.47±1.9	24.90±2.3	29.57±2.4	25.23±2.7	28.57±2.3	30.60±2.9	26.67±2.3
Heart	22.10±5.7	21.50±4.0	22.70±4.0	22.60±6.5	22.90±4.9	21.90±4.3	30.00±5.5	24.80±3.7
Image	14.67±1.4	6.94±1.0	15.23±1.0	6.67±1.0	14.30±0.9	6.52±0.9	7.51±1.0	20.10±4.4
Titanic	23.12±1.6	23.01±1.8	23.27±1.5	22.92±1.4	23.09±1.4	23.11±1.8	22.80±1.9	23.41±1.3
Twonorm	8.53±1.1	6.67±0.9	8.77±1.0	6.87±1.3	8.63±1.0	6.60±1.1	16.06±2.0	8.84±0.9
Waveform	21.02±2.1	16.88±1.8	20.80±2.4	18.16±2.0	20.41±2.2	16.96±1.6	20.26±2.2	15.57±0.9
All	19.07±2.5	**16.74±2.1**	19.11±1.9	**17.32±2.4**	18.94±2.1	**16.94±2.1**	21.12±2.4	20.58±2.3

Table: Mean classification errors and standard deviations at 30% asymmetric noise.

Dataset	AdaBoost		rBoost-Fixed gamma		rBoost		Gentle	Modest
	LR	rLR	LR	rLR	LR	rLR	Boost	Boost
Banana	31.45±5.2	23.53±4.7	27.31±4.5	**14.27±1.0**	32.39±3.9	23.83±4.1	25.38±2.7	33.04±6.8
Diabetes	32.20±2.1	33.43±3.9	29.47±3.0	**30.20±2.6**	32.80±3.3	33.27±3.1	38.37±3.6	32.07±3.5
Heart	27.60±5.5	27.30±6.8	23.00±4.3	**24.30±3.8**	28.20±6.3	28.00±6.9	32.00±7.2	29.60±11.7
Image	22.48±1.6	10.70±0.9	16.96±1.8	**5.47 ±1.0**	20.53±1.6	9.82 ±1.5	11.94±1.1	26.44±1.3
Titanic	32.60±8.4	31.21±8.7	23.88±1.8	**22.14±1.5**	33.17±9.3	30.73±8.7	32.94±9.0	33.49±13.7
Twonorm	16.02±2.4	12.07±2.0	8.89 ±1.5	**6.51 ±1.3**	14.68±2.3	12.19±2.0	17.85±1.8	16.62±3.1
Waveform	28.83±2.8	23.43±2.5	24.27±2.1	**19.95±1.6**	28.39±3.1	23.02±2.5	27.31±2.5	21.10±2.2
All	27.31±3.9	23.09±4.2	21.96±2.7	**17.54±1.8**	27.16±4.2	22.97±4.1	26.54±3.9	27.47±6.0

Discussion

- When γ_{jk} is known, the 'rBoost-fixed gamma' outperforms other approaches in asymmetric label noise conditions.
- However, the high complexity of boosting makes it difficult to estimate the label flipping probabilities.
- To circumvent this, extra knowledge of the problem can be incorporated into rBoost by supplying verified labels to Platt's calibration.
- The more accurate the probability calibration is the better the γ_{jk} estimates are.
- In our experience even a few verified labels can greatly improve the gamma estimates, and this leads to better classification performance.

Summary

- We presented robust Kernel Logistic Regression, and robust Adaboost.
- In the former, the underlying linear model (in the feature space) gave us some guidance + Bayesian regularisation provided the tools to derive a robust algorithm.
- But boosting robust classifiers seems not sufficiently robust. Rather interestingly, its effect is to speed up the boosting process. This could be advantageous in cases of low noise.
- Robustifying the boosting loss helps.
- In rBoosting, the estimation of the label noise parameters is greatly facilitated by a small trusted validation set for Platt's calibration algorithm.
- Much work remains to be done on better understanding boosting, and devising more principled robust boosters.

References

- J. Bootkrajang, A. Kabán. Learning kernel logistic regression in the presence of class label noise, Pattern Recognition 47 (11), 3641-3655, 2014.
- J. Bootkrajang, A. Kabán. Boosting in the presence of label noise. UAI 2013.

Induced label flipping for learning theory

- A. Kabán and R.J. Durrant. Structure-aware error bounds for linear classification with the zero-one loss, arXiv:1709.09782, 2017.

Linear Classification

- Given:

· $\mathcal{T}^N = \{(x_n, y_n) : (x_n, y_n) \overset{\text{i.i.d}}{\sim} \mathcal{D}\}_{n=1}^N$, where $\mathcal{D}$ is an unknown distribution over $\mathcal{X} \times \mathcal{Y}, \mathcal{X} \subseteq \mathbb{R}^d, \mathcal{Y} = \{-1, 1\}$

· $\mathcal{H} := \{x \to \text{sign}\,(h^T x) : h \in \mathbb{R}^d, x \in \mathcal{X}\}$

· $\ell : \mathcal{Y} \times \mathcal{Y} \to \{0, 1\}, \ell(\hat{y}, y) = \mathbf{1}(\hat{y} \neq y)$

- Goal: Find $\hat{h} \in \mathcal{H}$ s.t. its risk is as small as possible

$\mathsf{E}[\ell \circ \hat{h}] := \mathsf{E}_{(x,y)\sim\mathcal{D}}[\ell(\hat{h}(x), y) | \mathcal{T}^N]$

- Optimal classifier: $h^* := \underset{h \in \mathcal{H}}{\arg\min}\ \mathsf{E}_{(x,y)\sim\mathcal{D}}[\ell(h(x), y)]$

- Known: $|\mathsf{E}[\ell \circ \hat{h}] - \hat{\mathsf{E}}[\ell \circ \hat{h}]| = \tilde{\Theta}(\sqrt{d/N})$ in general.
- Q: What if $d > N$?

Compressive ERM Classifier

- Let $R \in \mathbb{R}^{k\times d}$, $k \leq d$ be a random matrix with i.i.d. 0-mean (sub-)Gaussian rows.
  · $\mathcal{T}_R^N = \{(Rx_n, y_n)\}_{n=1}^N$ RP of the training set
  · $\mathcal{H}_R := \{Rx \rightarrow \text{sign}\,(h_R^T Rx) : h_R \in \mathbb{R}^k \in \mathbb{R}, x \in \mathcal{X}\}$

- Compressive ERM: $\hat{h}_R = \underset{h_R \in \mathcal{H}_R}{\text{arg min}} \frac{1}{N}\sum_{n=1}^N \ell(h_R(Rx_i), y_i)$

- What is the generalization error of $\hat{h}_R$:

$$\mathsf{E}[\ell \circ \hat{h}_R] := \mathsf{E}_{(x,y)\sim\mathcal{D}}\left[\ell(\hat{h}_R(Rx), y)|\mathcal{T}^N, R\right] \leq ?$$

Two goals

- Generalization guarantees for compressive ERM classification & what structural characteristics of the data generator they depend on ('New Q')

- From the answers, obtain better worst-case guarantees for classification in the original data space, which adapts to the compressibility of the particular problem ('Old Q')

Our anaytic tool: Sign flipping probability

Lemma [Flip probability - Gaussian case] Let R be a 0-mean Gaussian RP matrix. Let $h, x \in \mathbb{R}^d$, and let $\theta = \theta_x^h \in [0, \pi)$ be the angle between them. Assume $h^T x \neq 0$, Then,

1. Exact form:

$$f_k(\theta) := \frac{\Gamma(k)}{(\Gamma(k/2))^2} \int_0^{\frac{1-\cos(\theta)}{1+\cos(\theta)}} \frac{z^{(k-2)/2}}{(1+z)^k} \mathrm{d}z = \Pr\left\{(Rh)^T Rx \leq 0\right\}$$

$$\Pr\left\{\frac{(Rh)^T Rx}{h^T x} \leq 0\right\} = f_k(\theta) \cdot \mathbf{1}(h^T x > 0) + (1 - f_k(\theta)) \cdot \mathbf{1}(h^T x < 0)$$

2. Upper bound: $\Pr\left\{\frac{(Rh)^T Rx}{h^T x} \leq 0\right\} \leq \exp(-k \cos^2(\theta)/2)$

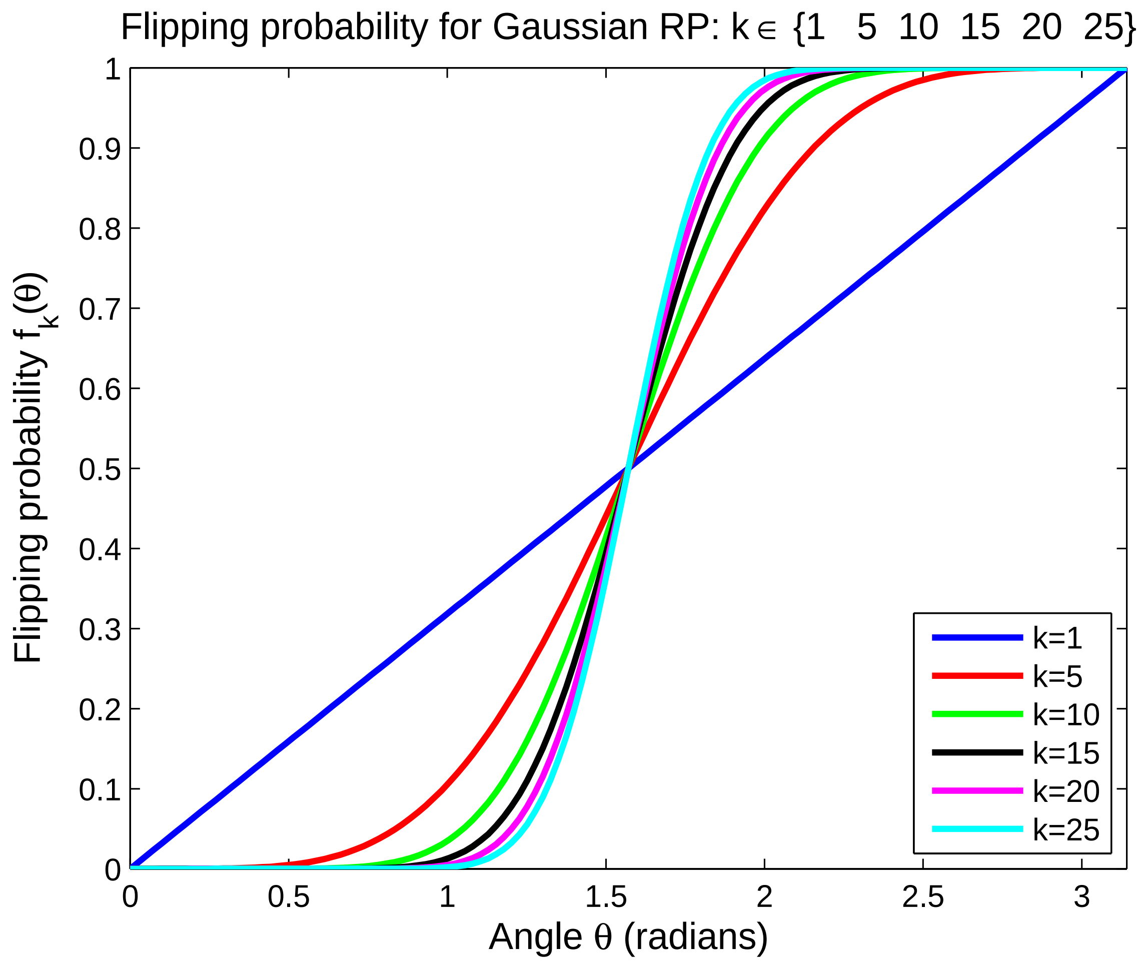


Illustration of the function $f_k(\theta)$ as a function of θ.

Definition [sub-Gaussian random variable] A zero-mean random variable X is subgaussian with parameter σ^2 if $\exists \sigma^2 > 0$ such that:

$$\mathsf{E}\left\{\exp(\lambda X)\right\} \leq \exp\left\{\sigma^2 \lambda^2 / 2\right\}$$

Lemma [Flip probability - sub-Gaussian case] Let R be a RP matrix with entries r_{ij} drawn i.i.d. from a zero-mean subgaussian distribution, let $h, x \in \mathbb{R}^d$, and let $\theta = \theta_x^h$ be the angle between them. If $h^T x \neq 0$, then:

$$\Pr\left\{\frac{(Rh)^T Rx}{h^T x} \leq 0\right\} \leq \exp(-k \cos^2(\theta)/8) \tag{1}$$

Risk bounds for compressive ERM classification

Theorem Take any $h \in \mathbb{R}^d$. Let R be a $k \times d$ sub-Gaussian random matrix with i.i.d. entries, $k \leq d$. Then, for all $\delta \in (0,1)$, the following holds for the compressive ERM classifier $\hat{h}_R$ with probability $1 - 2\delta$:

$$\begin{aligned}
\mathsf{E}_{x,y}\{\mathbf{1}(\hat{h}_R^T R x y \leq 0)\} \quad &\leq \quad \frac{1}{N}\sum_{n=1}^{N} \mathbf{1}\{h^T x_n y_n \leq 0\} + c\sqrt{\frac{k + \log(1/\delta)}{N}} \ldots \\
&+ \quad \frac{1}{N}\sum_{n=1}^{N} f_k^+(\theta_{x_n y_n}^h) + \min\left\{\frac{1-\delta}{\delta}\cdot\frac{1}{N}\sum_{n=1}^{N} f_k^+(\theta_{x_n y_n}^h), \sqrt{\frac{1}{2}\log\frac{1}{\delta}}\right\}
\end{aligned}$$

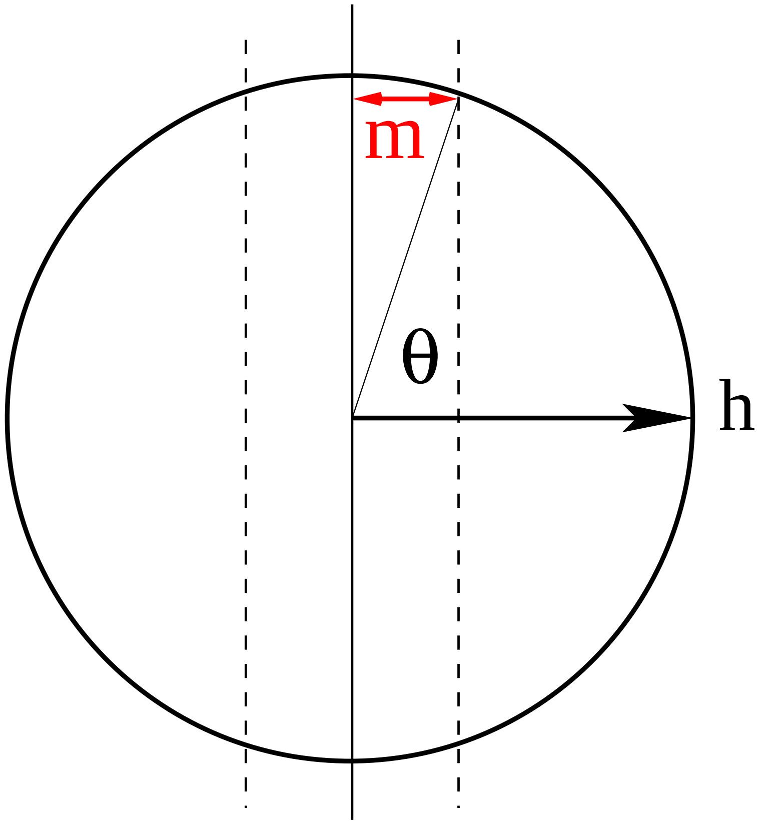
where $c > 0$ is a constant, and $f_k^+(\theta_u^h) := f_k(\theta_u^h) \cdot \mathbf{1}(h^T u > 0)$, where $f_k(\theta_u^h) = \mathsf{Pr}_R\left\{h^T R^T R u \leq 0\right\}$.

Likewise, w.p. $1-2\delta$,

$$\begin{aligned} \mathsf{E}_{x,y}[\mathbf{1}(\hat{h}_R^T Rxy \leq 0)] &\leq \mathsf{E}_{x,y}[\mathbf{1}(h^{*T}xy \leq 0)] + 2c\sqrt{\frac{k+\log(1/\delta)}{N}}... \\ &+ \mathsf{E}_{x,y}[f_k^+(\theta_{xy}^{h^*})] + \min\left\{\frac{1-\delta}{\delta}\cdot \mathsf{E}_{x,y}[f_k^+(\theta_{xy}^{h^*})], \sqrt{\frac{1}{2}\log\frac{1}{\delta}}\right\} \end{aligned}$$

• On RHRs, first 2 terms match a VC bound for k-dimensional linear classifier – complexity reduced from d to $k < d$.
• Last 2 terms pay the price.
• If k grows to d, we recover classical VC bound.
• But, last 2 terms can be small with $k << d$ if we are 'lucky'. Smaller than ϵ for $k \geq \frac{8\log(1/(\epsilon\delta))}{\min_n \cos^2(\theta_{x_n y_n}^h)}$, provided $\min_n \cos(\theta_{x_n y_n}^h) > 0$

Relation of Sign Flipping Probability to Margin



Flip probability and Margins

$$f_k^+(\theta) \leq \exp(-\tfrac{1}{8} k \cos^2(\theta))$$

$$\cos(\theta) = m$$

Large margin $\Rightarrow$
small flip probability (no $\Leftarrow$)

Back in the original data space

Theorem Fix any positive integer $k(\leq d)$. For any $\delta > 0$, with probability at least $1-\delta$ with respect to the random draws of $\mathcal{T}^N$ of size N, uniformly $\forall h \in \mathcal{H}$ it holds:

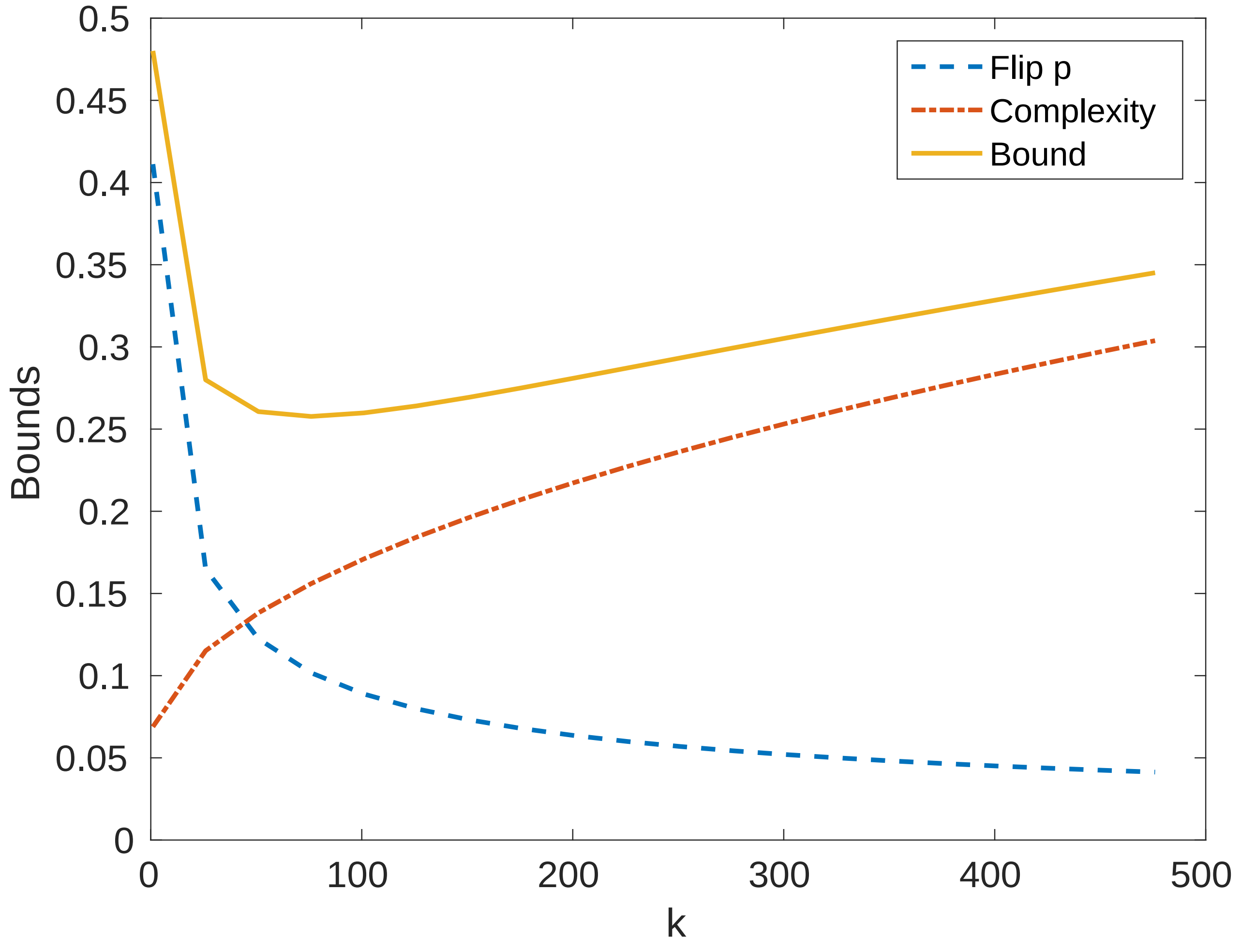
$$\mathsf{E}_{x,y}[\mathbf{1}(h^T xy \leq 0)] \quad \leq \quad \frac{1}{N}\sum_{n=1}^{N} \min(1, 2f_k(\theta^h_{x_n y_n})) + \frac{2\sqrt{2}}{\sqrt{\pi}}\sqrt{\frac{k}{N}} + 3\sqrt{\frac{\log(2/\delta)}{2N}}$$

• The 'complexity-term' (k) replaces VC-dimension at the expense of the new empirical error term.

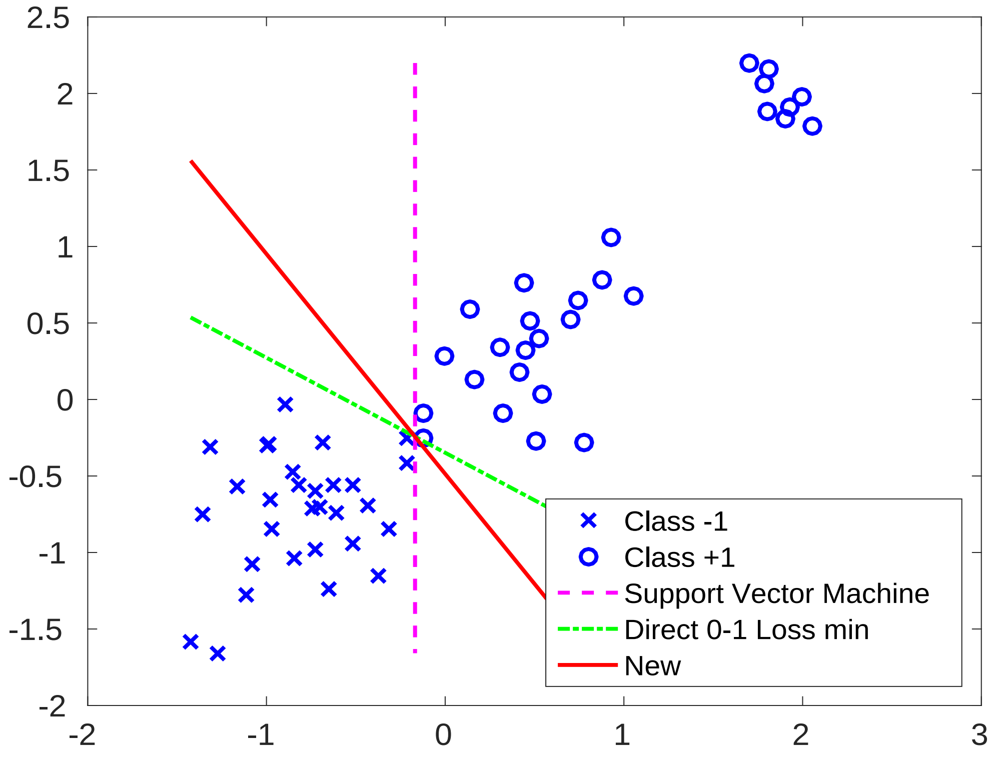
• Empirical error small if 'benign structure' present (for instance, margin).

• If $k \to d$ we recover the classical VC bound (with specified constant).

• Informative even if N small: Rather than wishing N was large, choose the matching complexity k & measure the error from the empirical term.


Flip p
Complexity
Bound
Bounds
k
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0
100
200
300
400
500

Bound minimising classifier



Bound minimising classifier

Test error rates ± std for the bound optimizer in comparisons. Bold font indicates significant win against SVM at the 0.05 level cf. a paired t-test. Underline in last two columns indicates statistically significant loss of competing methods. No statistically significant loss of our approach has been observed on the data sets tested,

Data set	N	d	New	SVM	0-1 Loss	LDM
Australian	690	14	**0.137± 0.015**	0.148± 0.013	0.156±0.077	$\underline{0.149\pm 0.014}$
German	1000	24	**0.260± 0.018**	0.280± 0.016	0.264±0.021	$\underline{0.315\pm 0.015}$
Haberman	306	3	**0.265± 0.025**	0.285± 0.050	0.268±0.024	0.276±0.030
Parkinsons	195	22	**0.141± 0.032**	0.221± 0.049	0.141± 0.036	0.135±0.034
PlRelax	182	12	**0.285± 0.029**	0.361± 0.166	$\underline{0.299\pm 0.035}$	0.290±0.051
Sonar	208	60	0.256± 0.045	0.271± 0.036	0.245±0.044	0.264±0.044

Summing up label flipping for linear classification

- The task is solvable in a random linear subspace (i.e. with performance guarantees) if the label flipping probabilities under a RP are small. This requirement is more general than large margin.

- The dataspace ERM classifier's error is small under the same conditions.

- We did not require any sparse representation for our bounds to hold, as usually compressed learning approaches do.