Sparse Bayesian Learning: Analysis and Applications

Huanhuan Chen

School of Computer Science and Technology
University of Science & Technology of China

Outline

Outline

What is Bayesian Inference?

Bayesian inference: a method of inference using Bayesian' rule to incorporate likelihood and our belief (prior) distributions with proper model selection.

$$P(\boldsymbol{w}|D) = \frac{P(D|\boldsymbol{w})P(\boldsymbol{w})}{P(D)}$$

- $\boldsymbol{w}$: is the weight vector of the model, e.g. weight vector in neural networks. D is the observed data set.
- prior $P(\boldsymbol{w})$: the probability of $\boldsymbol{w}$ before data D is observed. This can be expert knowledge or preference about the model, e.g. sparseness.
- likelihood $P(D|\boldsymbol{w})$: the probability of observing data D given $\boldsymbol{w}$.
- posterior $P(\boldsymbol{w}|D)$: the probability of $\boldsymbol{w}$ after D is observed.
- $P(D)$: marginal likelihood or model evidence. It is crucial for model selection in Bayesian inference.

Parametric or Nonparametric Bayesian

- Parametric Bayesian model: Prior on parameter with fixed or bounded number of parameters.
  - Prior on parameter: sparseness generating prior $\rightarrow$ sparse model
  - Examples: Bayesian neural networks, Relevance Vector Machine, Probabilistic Classification Vector Machine (PCVM), etc.
- Nonparametric Bayesian model: ∞-dimensional parameter space
  - Prior on function $\rightarrow$ very flexible models.
  - Not sparse and computational intensive: training $O(N^3)$, testing $O(N^2)$.
  - Examples: Gaussian Processes, Dirichlet Processes, etc

This talk focuses on parametric/sparse Bayesian model.

What is sparse model?

In the estimated model $f(X; \boldsymbol{w}) = X\boldsymbol{w}$, if many weights, i.e. $w_i = 0$, are zero, the obtained mode is referred as sparse model.

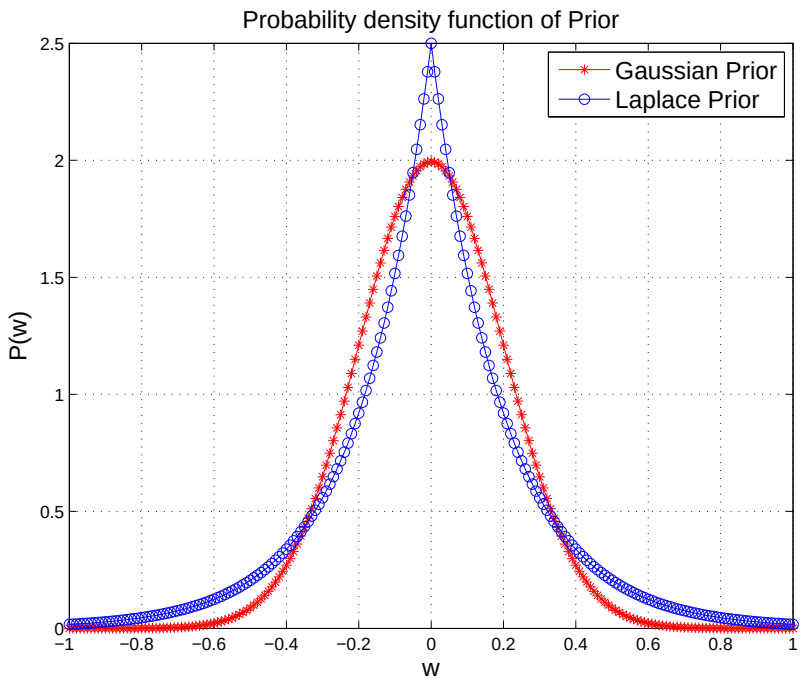
$$\overbrace{\begin{bmatrix} f_1 \\ f_2 \\ \cdots \\ f_N \end{bmatrix}}^{\mathbf{f}} = \overbrace{\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & \cdots & x_{1p} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & \cdots & x_{2p} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{N1} & x_{N2} & x_{N3} & x_{N4} & x_{N5} & \cdots & x_{Np} \end{bmatrix}}^{X} \cdot \overbrace{\begin{bmatrix} w_1 \\ 0 \\ 0 \\ w_4 \\ 0 \\ \cdots \\ w_p \end{bmatrix}}^{\mathbf{w}}$$

- Sparsity $\rightarrow$ variable selection $\rightarrow$ model interpretability.
- Sparsity $\rightarrow$ regularization $\rightarrow$ less overfitting & better prediction.

How to generate sparsity in sparse Bayesian learning?

Sparseness generating prior encourages sparseness:

- $P(w)$ has the highest probability when $w = 0$.
- The higher $P(w)$ at 0, more sparse.
- Examples: Gaussian prior, Gaussian prior with hyperparameter Gamma prior ($P(w_i) \propto 1/|w_i|$); Laplace prior $\cdots$



A Regression Example: Parametric Bayesian Solution

Given a training set $D = (\boldsymbol{x}_n, y_n)_{n=1}^N$, $\boldsymbol{x}_n \in R^p$, $y_n \in R$.

- Likelihood: training mean square error (MSE) assuming zero-mean Gaussian noise

$$P(D|\mathbf{w}) = (2\pi\sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}\|f(\mathbf{x}_n;\mathbf{w})-y_n\|^2\right\},$$

- Prior: regularization term

$$P(\mathbf{w}|\alpha) = \prod_{n=0}^{N} N(w_i|0,\alpha_i^{-1}),$$

- Posterior: the optimized weight vector

$$\max_{\boldsymbol{w}} \log P(\boldsymbol{w}|D) \propto \min_{\boldsymbol{w}} \sum_{n=1}^{N} (f(\boldsymbol{x}_n;\boldsymbol{w}) - y_n)^2 + \sum_{i=1}^{p} \alpha_i w_i^2$$

Maximization of posterior in Bayesian inference is equivalent to regularized regression, with the prior as regularized term.

Relationship between Sparse Learning and Bayesian Inference

Sparse Learning

$\hat{\boldsymbol{w}} = \arg\min_w R(\boldsymbol{w}) + \lambda g(\boldsymbol{w})$

- $R(\boldsymbol{w})$– Likelihood (cost) function, e.g. MSE, cross entropy, etc
- $g(\boldsymbol{w})$– (prior) sparse regularization, e.g. l_0, l_1-lasso
- Parameter λ need to be tuned by cross validation

Spare Bayesian Learning (SBL)

- $\arg\max_w \log P(\boldsymbol{w}|D) \propto \min_{\boldsymbol{w}} R(\boldsymbol{w}) + \sum_{n=1}^{N} \alpha_n g(w_n)$
- Parameter α_n is equivalent to the trade-off parameter λ.

The benefits of Bayesian Inference

- Automatic model selection, i.e. regularization parameters α_i and (potential) kernel parameters, feature selection, $\cdots$, by maximizing the model evidence $P(D|\boldsymbol{\alpha})$.
- Expert knowledge or preferences of the models can be easily incorporated into the model by prior distribution.
- Probabilistic outputs with confidence intervals (covariance matrix)

Ignore the normalization term, or not?

$$P(\boldsymbol{w}|D) = \frac{P(D|\boldsymbol{w})P(\boldsymbol{w}|\boldsymbol{\alpha})}{P(D|\boldsymbol{\alpha})}$$

- To simplify calculation, the normalization term $P(D|\boldsymbol{\alpha}) = \int P(D|\boldsymbol{w})P(\boldsymbol{w}|\boldsymbol{\alpha})d\boldsymbol{w}$ is often ignored to save calculation.
- In fact, $P(D|\boldsymbol{\alpha})$ is crucial for automatic model selection, i.e. automatically choose best α_n.

How to automatically select model in SBL?

For best hyperparameter α after observing data D, we need to maximize the posterior of $P(\boldsymbol{\alpha}|D)$:

$$P(\boldsymbol{\alpha}|D) = \frac{P(D|\boldsymbol{\alpha})P(\boldsymbol{\alpha})}{P(D)}$$

If an uniform prior $P(\boldsymbol{\alpha})$ is adopted. Then,

$$P(\boldsymbol{\alpha}|D) \propto P(D|\boldsymbol{\alpha}).$$

Maximization of evidence $P(D|\boldsymbol{\alpha})$ is to maximize the posterior $P(\boldsymbol{\alpha}|D)$ of hyperparameter.

Iteratively optimize posterior and evidence in SBL

1. Initialization: choose an initial hyperparameter $\boldsymbol{\alpha}$ value.
2. Posterior maximization: update the optimal weight vector $\boldsymbol{w}$ by maximizing the posterior of weights $P(\boldsymbol{w}|D)$ with previous $\boldsymbol{\alpha}$.
3. Evidence maximization: update the hyperparameter α by maximizing the evidence $P(D|\boldsymbol{\alpha})$.
4. Loop steps (2) and (3) until converged.

What are the critical problems in SBL?

- Choose proper prior and likelihood distributions for specific problems.
- Effective optimization approaches to maximize the posterior of parameters and model evidence: gradient based approaches, coordinate descent, etc.
- Posterior and evidence $P(D|\boldsymbol{\alpha}) = \int P(D|\boldsymbol{w})P(\boldsymbol{w}|\boldsymbol{\alpha})d\boldsymbol{w}$ are important but often intractable if prior or likelihood is not Gaussian!
  - Hidden variable solutions: Expectation Maximization. Pros: simple derivations and implementations; cons: sensitive to initializations, local minima.
  - Integral approximation techniques for analytical solutions: Laplace approximation, Variational Bayesian, Expectation Propagation (EP)

Rethinking of two questions in SBL

Is Gaussian prior appropriate for all problems?

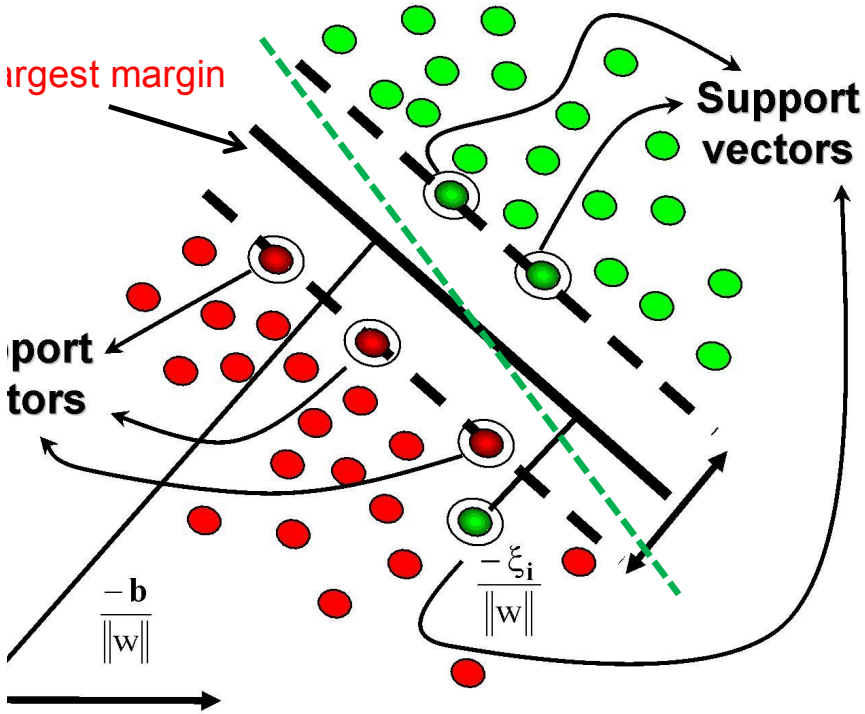
- Bayesian methods are the most powerful when your prior adequately captures your beliefs. Improper prior yield unreasonable inferences.
- Gaussian prior used for several decades. Is it proper for classification?

Does more sparsity mean better solutions?

- More sparsity: simpler model, but might lack of freedom to approximate the feature-label mapping.

Outline

Support Vector Machines: Margin Maximisation



- SVM maximises the margin of two classes and try to minimise the generalisation error.
- The training points that are nearest to the separating function are called support vectors. The model is immune to removal of any non-support-vector data points.

Formulation

SVM makes predictions based on the function:

$$f(\boldsymbol{x}; \boldsymbol{w}) = sign\left(\sum_{n=1}^{N} y_n w_n K(\boldsymbol{x}, \boldsymbol{x}_n) + b\right)$$

- x_n are training examples
- $K(x, x_n)$ is the kernel function
- $y_n \in \{-1, +1\}$ is the label for x_n
- N is the total number of training examples
- w_n is non-negative Lagrange multiplier: w_n is either zero or positive.

Advantages

- Good generalization
- Sparse solution: some weights w_n are 0.

Disadvantages

- Non-probabilistic but hard binary decisions.
- Kernel parameters and parameter C (control the error tolerance) need to be tuned by cross validation: time consuming.

Relevance Vector Machine

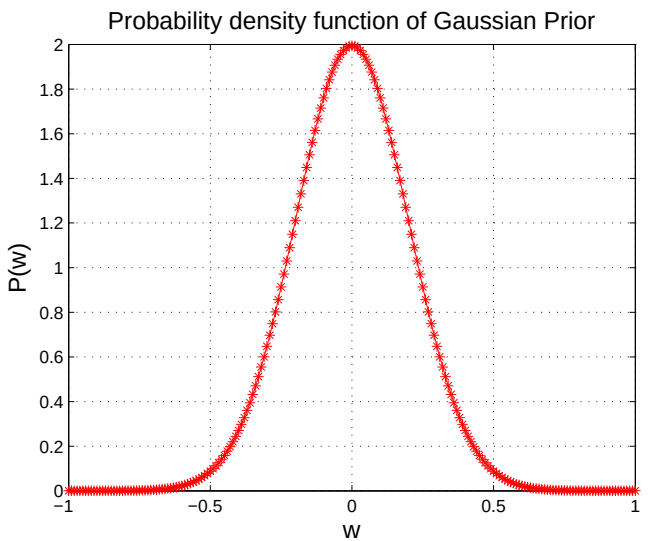
- A Bayesian treatment of a generalized linear model

$$f(\boldsymbol{x}; \boldsymbol{w}) = \sigma\left(\sum_{n=1}^{N} w_n \phi_n(\boldsymbol{x}) + b\right),$$

  where $\sigma(\cdot)$ is the sigmoid function for probabilistic outputs.
- RVM is a Bayesian linear model with sparse prior on weights $\boldsymbol{w}$

$$p(w_n|\alpha_n) = N(w_n|0, \alpha_n^{-1}).$$

  where α_n is the inverse variance of Gaussian.



Analysis of RVM

Advantages

- probabilistic output
- sparser than SVM

Disadvantages

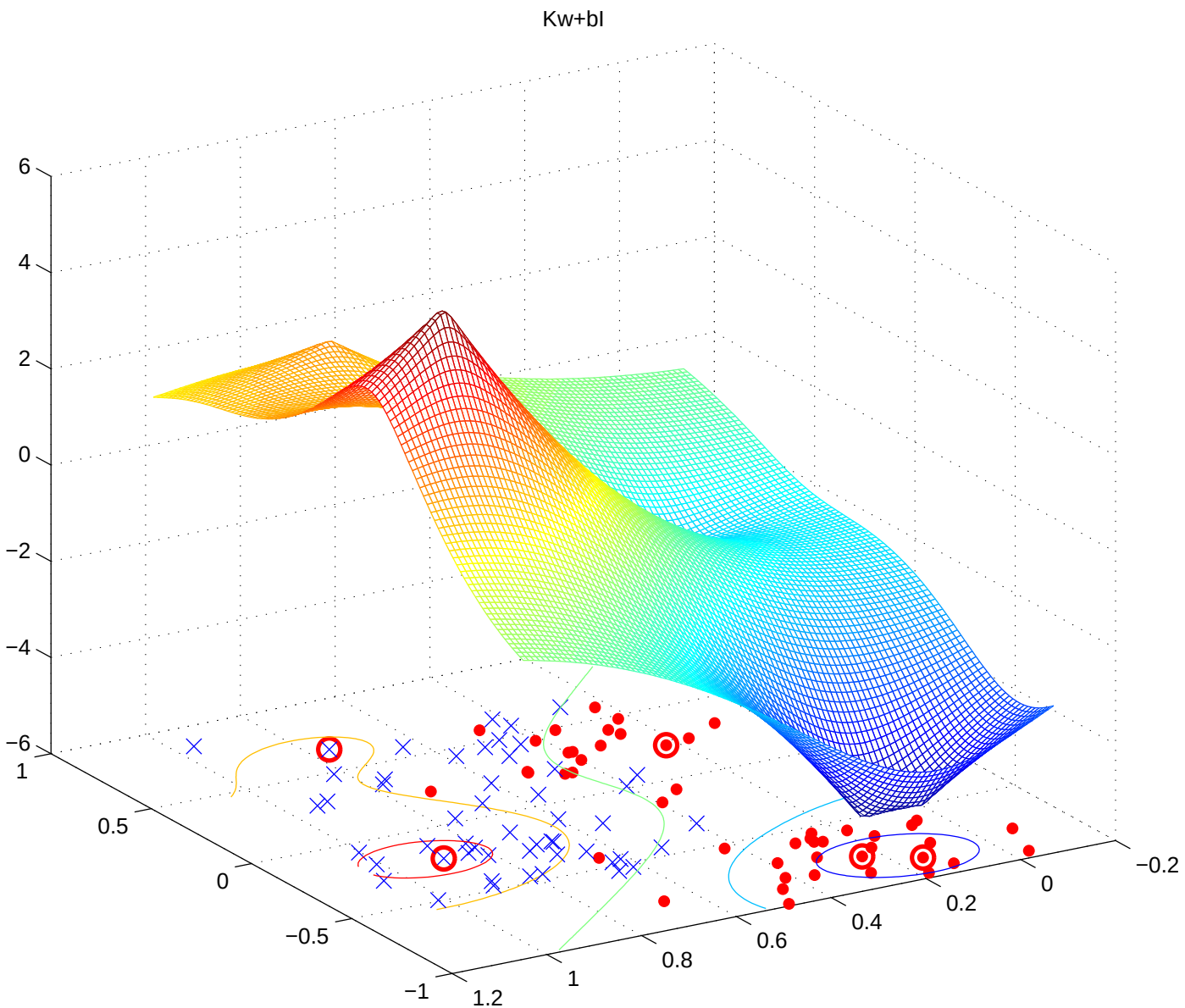
- Some training points that belong to positive class ($y_n = +1$) may have negative weights and vice versa, leading to the situation that the decision of RVM is based on some untrustworthy vectors, and thus is sensitive to the kernel parameter (even with well-selected kernel parameters.)

- In kernel methods, every point makes impacts to the decision boundary.
- In SVM, every point will either "vote" for decision domain by class label, or do not vote.
- In RVM, every point can "vote" for and against decision domain.

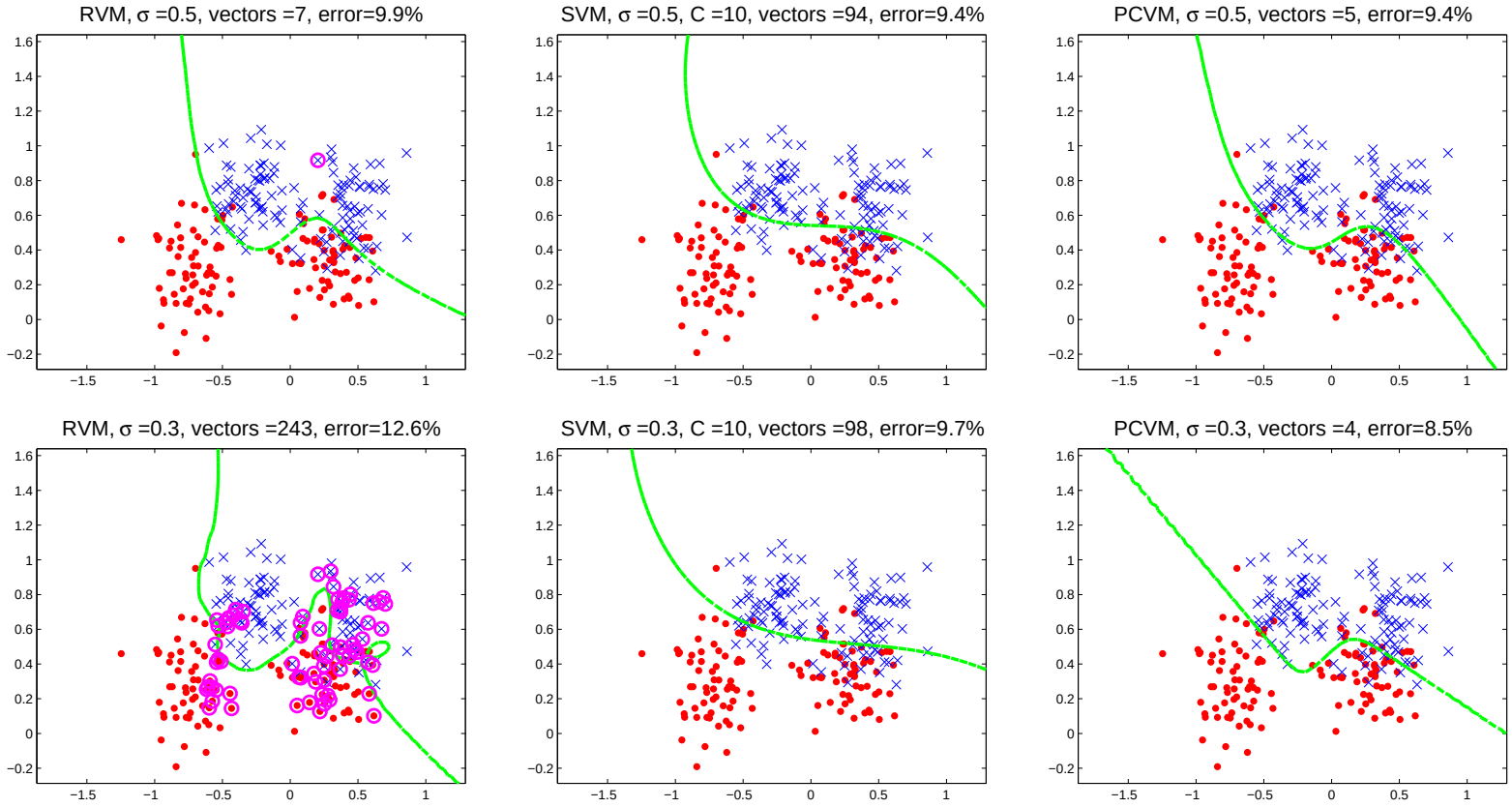
Voting for or/and Against?

- "Any voting system permits some expression of disapproval, but these are necessarily confused with expressions of choice or approval, leading some to conclude that separating these expressions is best." (wikipedia)
- Is this the same in machine learning?

Illustration

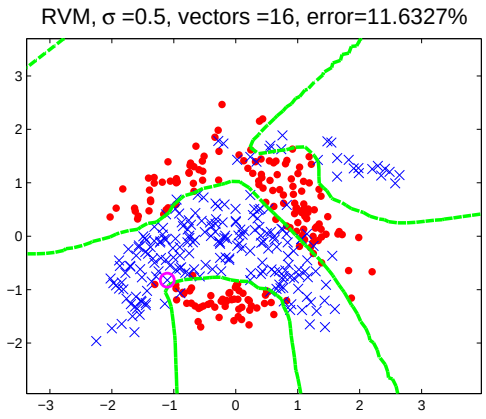


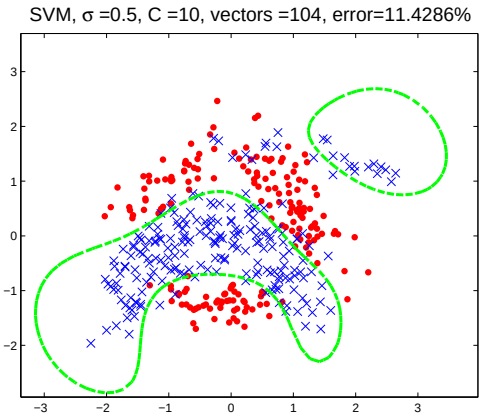
Unstable RVM with respect to kernel parameter (Gaussian kernel)

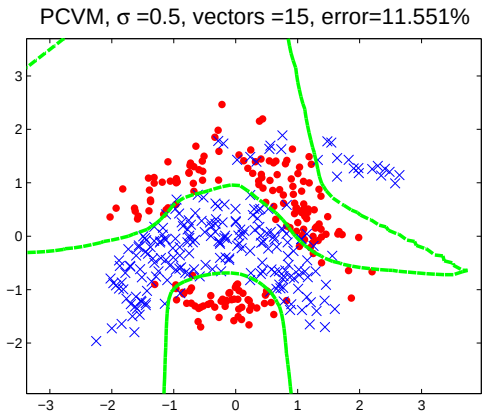


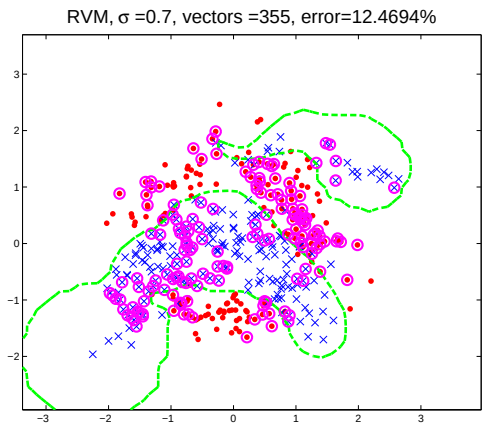
- The used vectors ($w_i \neq 0$) whose weights have opposite signs are shown circled.
- More redundant vectors with small opposite signs might lead to unstable solutions.

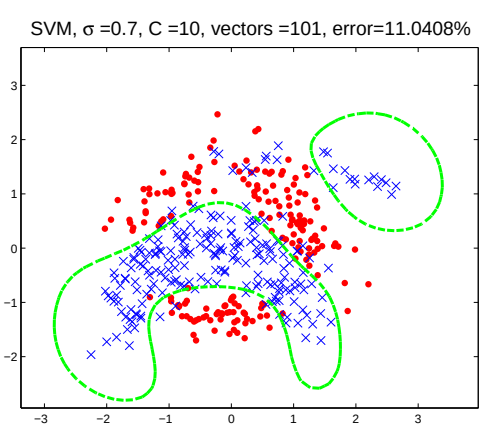
Unstable RVM with respect to kernel parameter

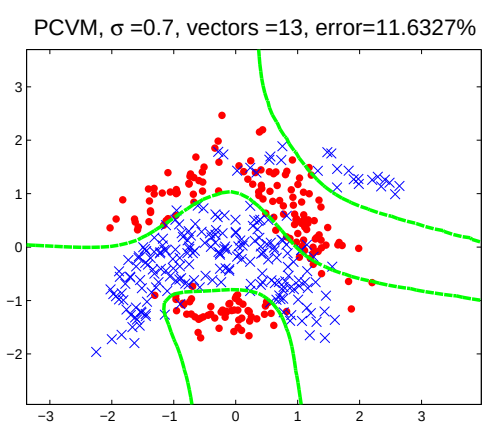












Theoretical Analysis (Chen09)

- Maximum-a-posterior (MAP) analysis: PCVM with truncated priors has higher posterior than models with Gaussian priors.
- Margin analysis: PCVM with truncated priors has larger margin than models with Gaussian priors, especially with a localized basis function.

Discussions

- SVM always assign positive/negative weights to positive/negative "points". This principle is implemented in SVM by enforcing the Lagrange multipliers to be non-negative.

- How to combine the advantages of RVM and SVM and discard the unstable characteristics?

- Combine the advantages of SVM and RVM

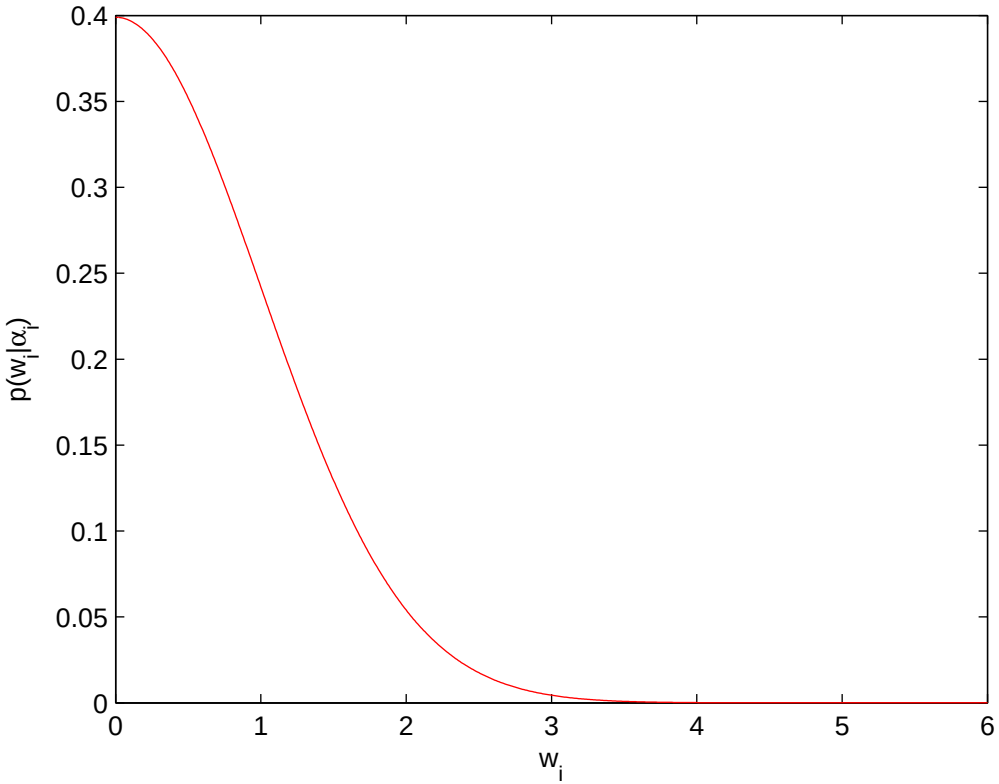
$$y(\boldsymbol{x}; \boldsymbol{w}) = \sigma\left(\sum_{n=1}^{N} y_n w_n \phi_n(\boldsymbol{x}) + b\right),$$

- Left-truncated Gaussian Prior on w_n for non-negative w_n

$$p(w_n|\alpha_n) = \begin{cases} 2N(w_n|0, \alpha_n^{-1}) & \text{if } w_n \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- Hyper-parameters: parameter α_n.

Truncated Prior for PCVM



- The mean of truncated Gaussian prior is larger than zero.
- It is less sparse than RVM with (hierarchical) student-t prior.
- Question: more spareness = better generalization?

Non-negative Prior

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = N(w_0|0, \alpha_0^{-1}) \prod_{i=1}^{N} 2N(w_i|0, \alpha_i^{-1}) \cdot \delta(w_i)$$

where $\delta(\cdot)$ is the indicator function $\mathbf{1}_{x \geq 0}(x)$.

Bernoulli Likelihood

$$p(\boldsymbol{t} \mid \boldsymbol{w}) = \prod_{n=1}^{N} \sigma_i^{t_i} \left[1 - \sigma_i\right]^{1-t_i},$$

where $\sigma_n = \sigma\left(\sum_{n=0}^{N} w_n \phi_n(\boldsymbol{x}_n)\right)$, $\boldsymbol{t} = (t_1, t_2, \cdots, t_N)^T$ is a vector of targets, $t_i = \frac{y_i+1}{2} \in \{0, 1\}$ is the probabilistic target.

- According to Bayesian's theorem, the posterior is:

$$p(\boldsymbol{w}|\boldsymbol{t}) = \frac{p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})}{p(\boldsymbol{t}|\boldsymbol{\alpha})}.$$

- The integral to calculate posterior $p(\boldsymbol{w}|\boldsymbol{t})$ and model evidence $p(\boldsymbol{t}|\boldsymbol{\alpha}) = \int p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})d\boldsymbol{w}$ are intractable due to the truncated prior.

Solutions

Hidden variables

- Expectation Maximization (EM): simple derivations, simultaneously optimize kernel parameters, but sensitive to initialization and may converge to local minima (Chen09)

Integral Approximation

- Laplace Approximation: deterministic and fast, and the performance is acceptable (verified by MCMC) (Chen13)
- Expectation Propagation (EP): accurate but slow (Chen13)
- Markov chain Monte Carlo (MCMC): the most accurate but very slow (Chen13)

The most probable $\boldsymbol{w}$, i.e. posterior, can be obtained by maximizing the following log likelihood

$$
\begin{aligned}
Q &= \log\{p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})\} - \log p(\boldsymbol{t}|\boldsymbol{\alpha}) \\
&= \sum_{i=1}^{N}[t_i \log \sigma_i + (1-t_i)\log(1-\sigma_i)] - \frac{1}{2}\sum_{i=0}^{N}\alpha_i w_i^2 \\
&\quad + \sum_{i=1}^{N}\log \delta(w_i) - \text{const.}
\end{aligned}
$$

Posterior of weight vector

Analyzing the first/second gradient of the above equation and we obtain the optimal value

$$\begin{aligned} \boldsymbol{w}_{MAP} &= \boldsymbol{A}^{-1}\left(\Phi^T(\boldsymbol{t}-\boldsymbol{\sigma})+\boldsymbol{k}\right) \\ \Sigma_{MAP} &= (\Phi^T\boldsymbol{B}\Phi+\boldsymbol{A}+\boldsymbol{D})^{-1}. \end{aligned}$$

where $\sigma_i = \sigma\left(\sum_{n=0}^{N} y_n w_n \phi_n(\boldsymbol{x}_i)\right)$,

- $\boldsymbol{A} = diag(\alpha_0, \alpha_1, \cdots, \alpha_N)$,
- $\boldsymbol{D} = diag(0, d_1, \cdots, d_N) =$ $diag(0, \sigma(\beta w_1)(1-\sigma(\beta w_1))\beta^2, \cdots, \sigma(\beta w_N)(1-\sigma(\beta w_N))\beta^2)$
- $\boldsymbol{k} = [0, \beta(1-\sigma(\beta w_1)), \cdots, \beta(1-\sigma(\beta w_N))]^T$ is the $N+1$ vector, aiming to ensure that weights w_i are non-negative.

Efficient PCVM by sequentially maximize model evidence

Model evidence $L(\boldsymbol{\alpha}) = P(\boldsymbol{t}|\boldsymbol{\alpha})$ can be written as

$$L(\boldsymbol{\alpha}) = L(\boldsymbol{\alpha}_{-i}) + l(\alpha_i),$$

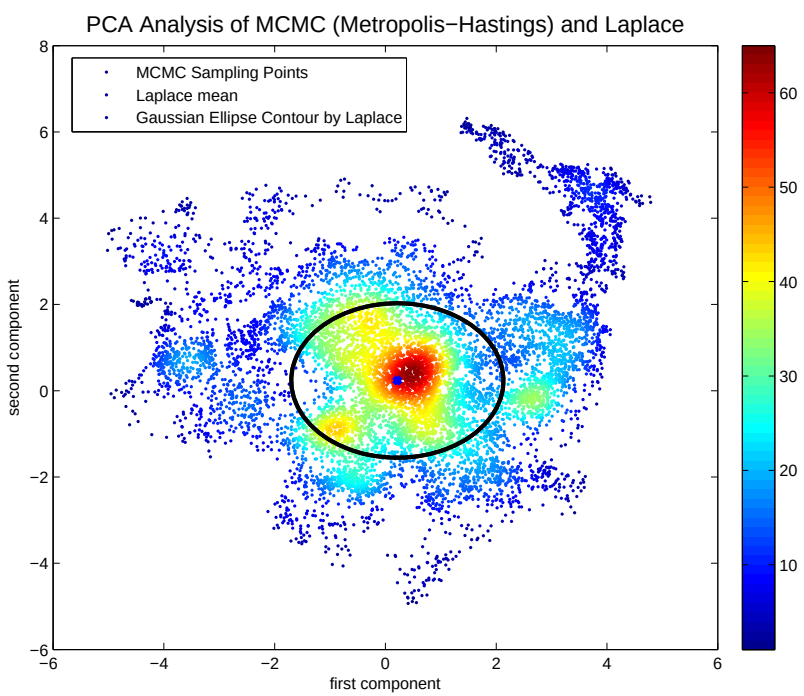
where

- $L(\boldsymbol{\alpha}_{-i})$: the model evidence with basis function ϕ_i deleted.
- $l(\alpha_i)$: the contribution of α_i to evidence when include ϕ_i.

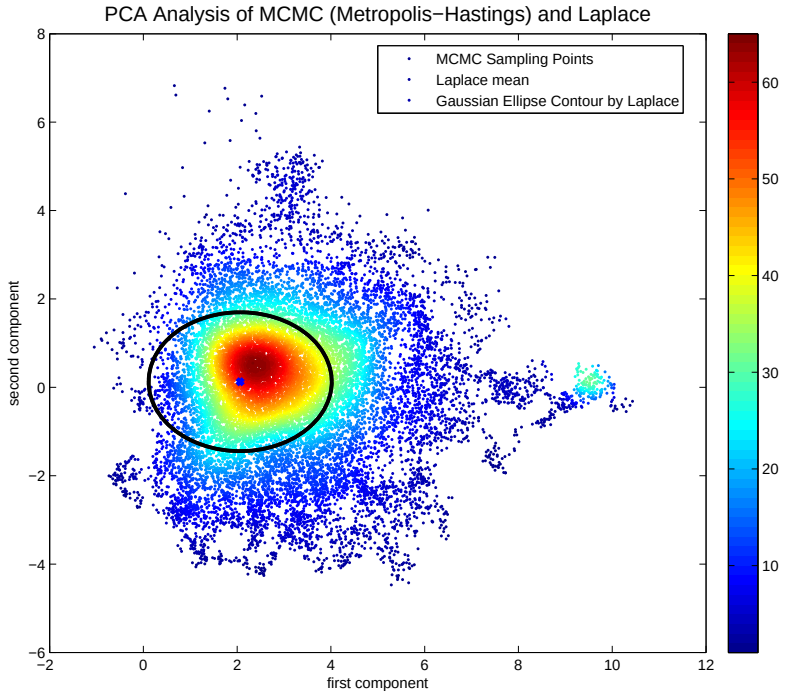
Analyzing each $l(\alpha_i) \rightarrow$ sequentially maximize model evidence $\rightarrow$ incremental PCVM.

Outline

MCMC vs. Laplace Approximation

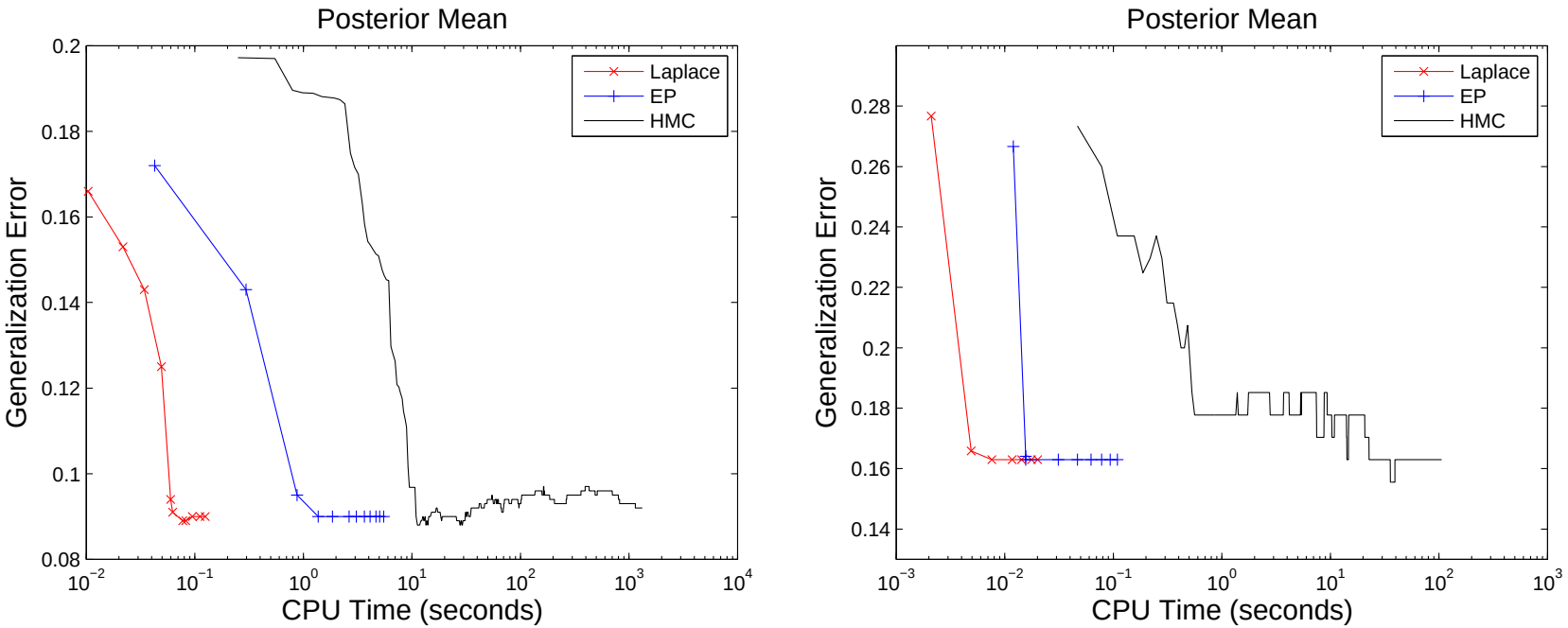


(m) Synth



(n) Heart

Figure : The posteriors of combination weights calculated by MCMC (40000 sampling points) and Laplace Approximation.



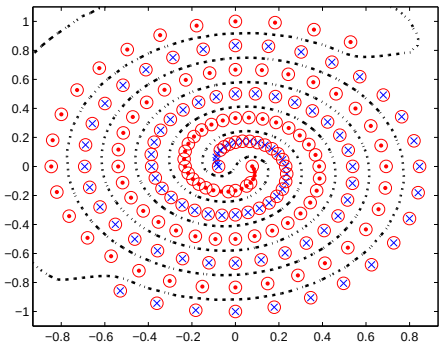
(a) Synth (b) Heart

Figure : The comparisons of Laplace approximation, expectation propagation and hybrid monte carlo (200,0000 sampling points) in terms of generalization error and CPU time.

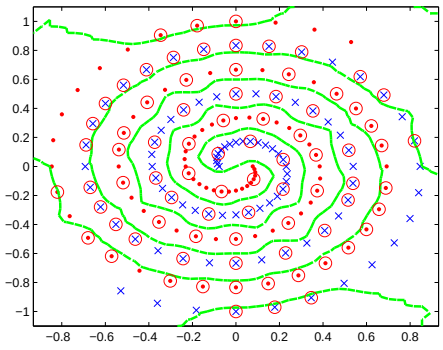
Table : Comparisons of MCMC, EP and Laplace approximation on four data sets.

Methods	Cancer				Diabetics			
	error	AUC	#vec	CPUTime	error	AUC	#vec	CPUTime
MCMC	26.61	71.94	12	669.1s	23.17	82.86	23	764.1s
EP	26.65	72.53	9	3.2s	23.18	82.89	17	357.2s
Laplace	26.71	72.03	16	0.2s	23.11	83.12	22	1.1s
Methods	Heart				Thyroid			
	error	AUC	#vec	CPUTime	error	AUC	#vec	CPUTime
MCMC	16.37	90.67	16	707.4s	4.94	98.71	22	913.1s
EP	16.65	90.91	13	254.7s	5.16	98.63	10	61.2s
Laplace	16.65	90.83	15	0.3s	5.02	98.87	21	0.2s

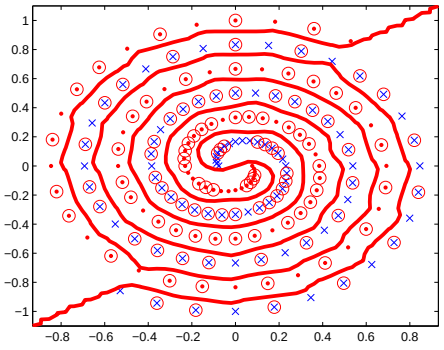
Synthetic Data Sets



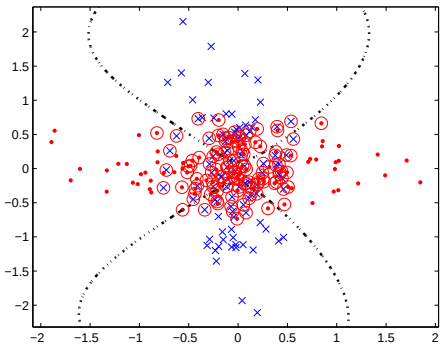
(a) Spiral: SVM



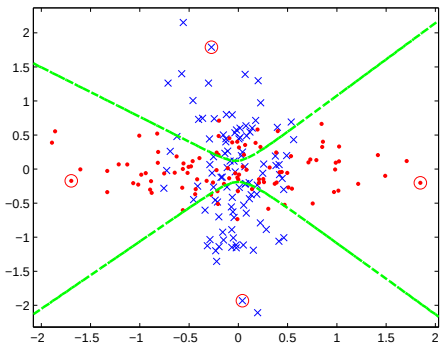
(b) Spiral: RVM



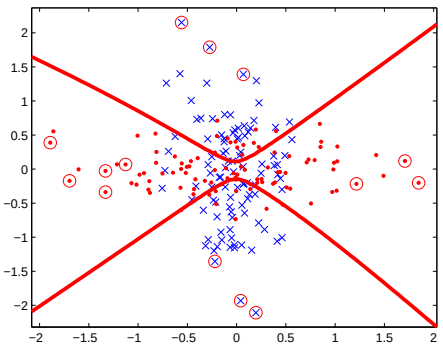
(c) Spiral: PCVM



(d) Bumpy: SVM

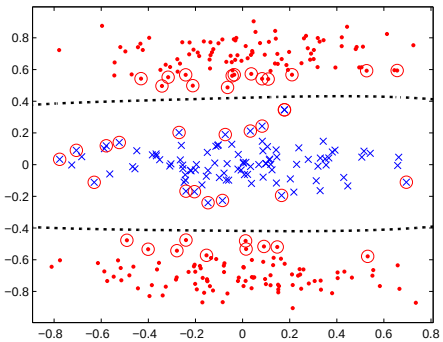


(e) Bumpy: RVM

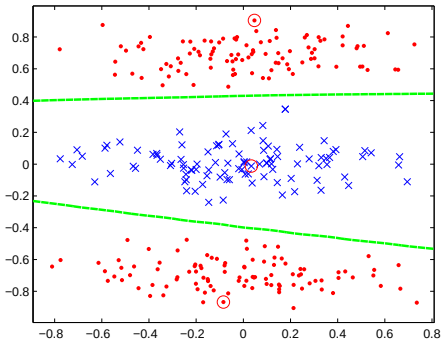


(f) Bumpy: PCVM

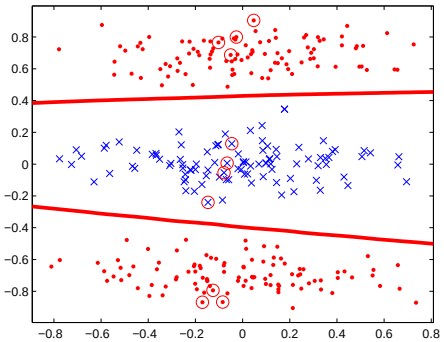
Synthetic Data Sets



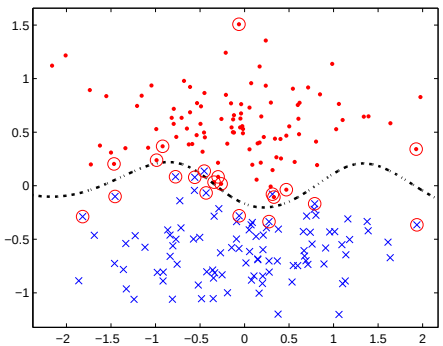
(g) Relevance: SVM



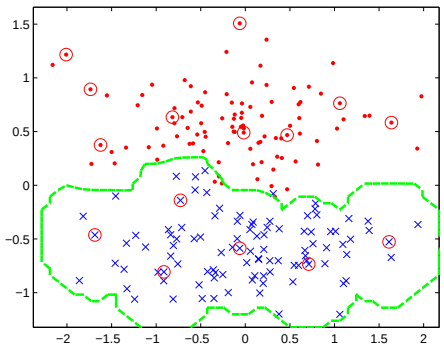
(h) Relevance: RVM



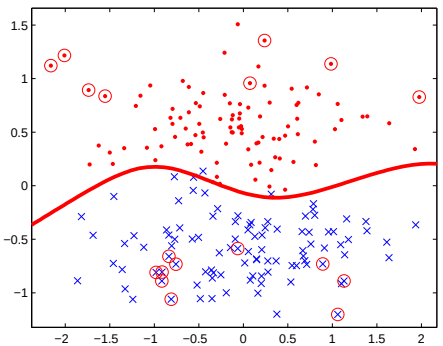
(i) Relevance: PCVM



(j) Overlap: SVM



(k) Overlap: RVM



(l) Overlap: PCVM

PCVM can handle predominating linear data.

Setup for Benchmark Tests

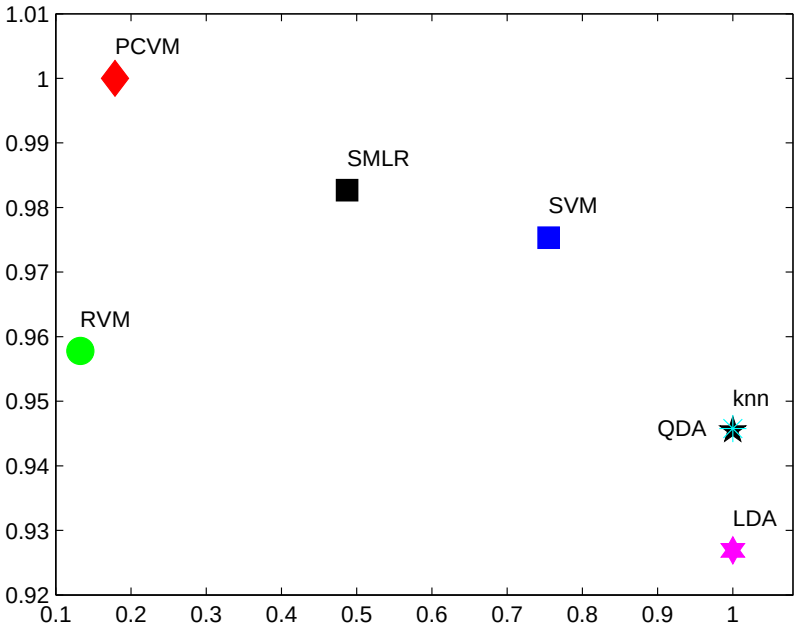
- Compared algorithms: PCVM, SVM, relevance vector machine (RVM) and sparse multinomial logistic regression (SMLR).
- Baseline algorithms: linear/quadratic discriminant analysis (LDA/QDA) and k Nearest Neighbor (kNN).
- Parameter optimization by cross validations, including kernel parameters in SVM, RVM, EPCVM, SMLR.

SMLR stands for sparse multinomial logistic regression (Krishnapuram05: Sparse Multinomial Logistic Regression: Fast Algorithms and Generalization Bounds, IEEE TPAMI, 27(6), 2005)

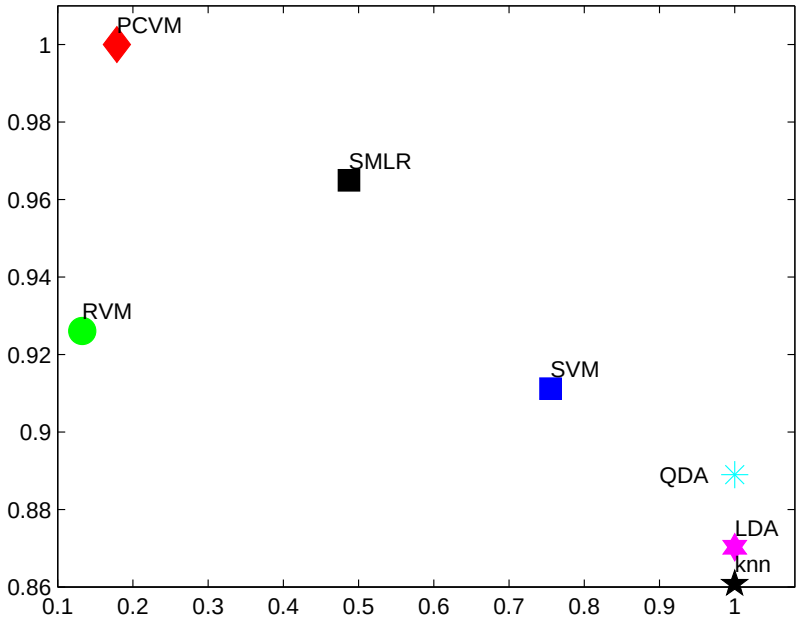
Summary of Benchmark Data Sets

Data	No. Train	No. Test	Positive %	Negative %	Dim
Abalone	2089	2088	50.18%	49.82%	8
Banana	2650	2650	44.83%	55.17%	2
Cancer	132	131	29.28%	70.72%	9
Diabetics	384	384	34.90%	65.10%	8
German	500	500	30.00%	70.00%	20
Heart	135	135	44.44%	55.56%	13
Image	1043	1043	56.95%	43.05%	18
Ringnorm	3700	3700	49.51%	50.49%	20
Splice	1496	1495	44.93%	55.07%	60
Thyroid	108	107	30.23%	69.77%	5
Titanic	1101	1100	58.33%	41.67%	3
Twonorm	3700	3700	50.04%	49.96%	20
Waveform	2500	2500	32.94%	67.06%	21

Benchmark Results



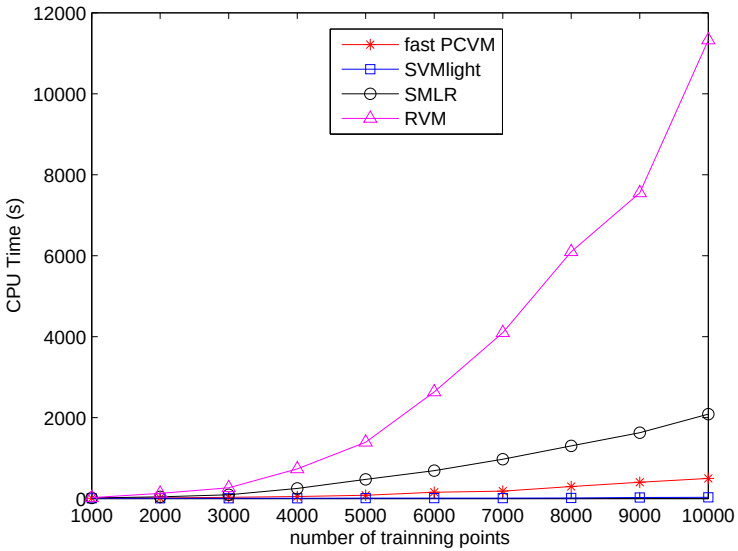
(m) Error Rate



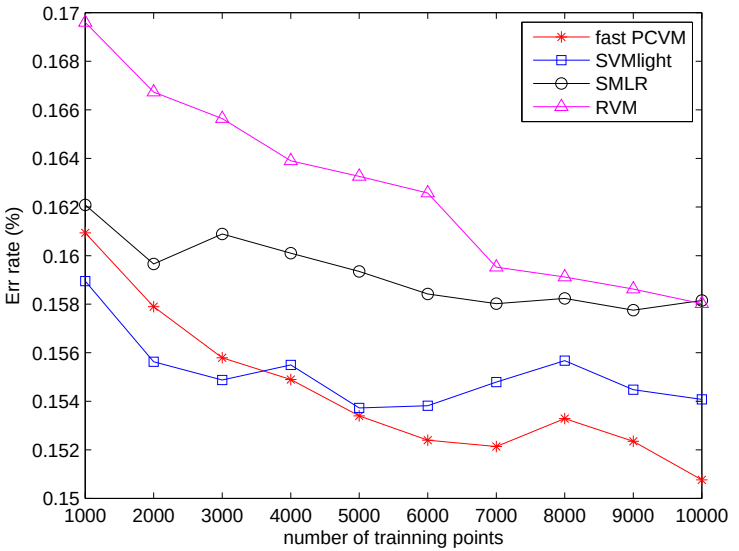
(n) AUC

- x-axis: sparsity degree, i.e. % data points used in prediction
- y-axis: normalized performance across 13 data sets.
- PCVM is less sparse than RVM.
- PCVM achieves the best performance with error rate and AUC.

Scalability



(o) CPU time on Adult Data Set



(p) Error Rate on Adult Data Set

Figure : Comparison of CPU time and the err rate of fast PCVM, SVM, SMLR and RVM on Adult data set.

Analysis

- PCVM scales well with the number of training points without compromise the performance.
- RVM and SMLR do not scale well with increased data points.
- SVMLight is the fastest algorithm as it was optimised by sequential minimal optimization algorithm (SMO) and the optimization for large problems have been implemented.

Outline

Rademacher Complexity Bound

Rademacher complexity measures "richness" of a class of real-valued functions.

(Meir03) Consider arbitrary scalars $g > 0$, $r > 0$. Then, for $\delta \in (0, 1)$ with probability at least $(1 - \delta)$ over draws of training sets, the following bound holds:

$$P(yf(\boldsymbol{x}, q) < 0) \leq R_{emp}[f, D] + \frac{2}{s}\sqrt{\frac{2\tilde{g}(q)}{N}} + \sqrt{\frac{\ln \log_r \frac{r\tilde{g}(q)}{g} + \frac{1}{2}\ln\frac{1}{\delta}}{N}},$$

where R_{emp} is the empirical loss,

$$R_{emp}[f, D] = \frac{1}{N}\sum_{n=1}^{N} l_s(y_n f(\boldsymbol{x}_n, q)), \text{ and } \tilde{g}(q) = r \cdot \max(KL(q||p), g),$$

where $KL(q||p)$ is the Kullback-Leibler divergence from the posterior q to the prior p over parameters $\boldsymbol{w}$.

KL Divergence between Prior and Posterior

- The KL divergence is a non-symmetric measure of the difference between two probability distributions.
- The bound is related to $R_{emp}[f, D]$ and $KL(q||p)$. Given the same $R_{emp}[f, D]$, the bound is tight with small $KL(q||p)$.
- The KL divergence from normalized truncated posterior $p(\boldsymbol{w}|\boldsymbol{t})$ to truncated Gaussian prior $p(\boldsymbol{w}|\boldsymbol{\alpha})$ is

$$KL(q||p) = \frac{1}{A_0} \int_0^\infty \tilde{p}(\boldsymbol{w}|\boldsymbol{t}) \ln \frac{\tilde{p}(\boldsymbol{w}|\boldsymbol{t})}{p(\boldsymbol{w}|\boldsymbol{\alpha})} d\boldsymbol{w} - \ln A_0.$$

  where $\tilde{p}$ stands for un-normalized posterior/prior.
- A_0: the cumulative distribution function of posterior $p(\boldsymbol{w}|\boldsymbol{t})$ when weights are non-negative.

Kullback-Leibler Divergence Between Prior and Posterior

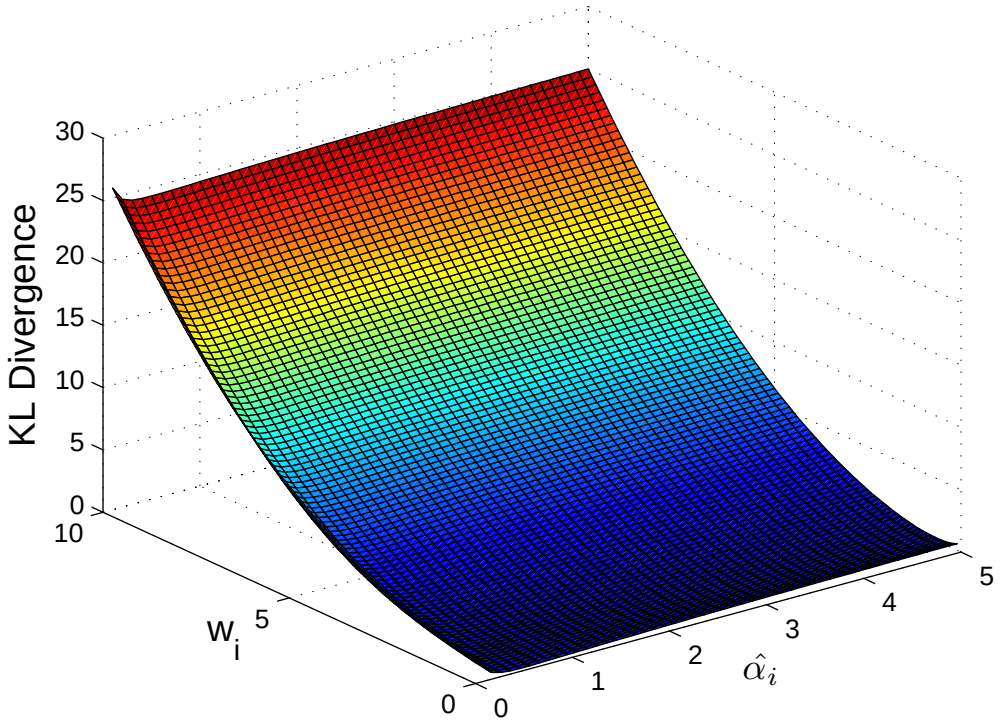
Adopt the independence assumption on weight vector, then

$$KL(q||p) = \sum_{i, w_i \neq 0} \left\{ \begin{array}{c} \frac{1}{2}\left[\frac{\alpha_i}{\hat{\alpha}_i} - 1 + \ln\left(\frac{\hat{\alpha}_i}{\alpha_i}\right) + \alpha_i w_i^2\right] \\ + \frac{(2\pi\hat{\alpha}_i)^{-1/2}(\alpha_i + \hat{\alpha}_i) w_i}{erfcx\left(-w_i\sqrt{\hat{\alpha}_i/2}\right)} \\ - \ln\left(erfc\left(-\frac{w_i\hat{\alpha}_i}{2}\right)\right) \end{array} \right\},$$

where

- $erfcx(a) = e^{a^2} erfc(a)$.
- α_i are the initial hyperparameters.
- $\hat{\alpha}_i$ are the optimised hyperparameters.
- Fix the initial hyperparameter to $\alpha_i = 0.5$ (the value used in the paper), then we obtain

KL divergence between Truncated Posterior and Gaussian prior



- $KL(q||p)$ is much more sensitive to weights w_i than the optimized posterior hyperparameters $\hat{\alpha}_i$.
- Sparseness helps to minimize the $KL(q||p)$ divergence.

Sparseness and the Bound

$$\tilde{g}(q) = r \cdot \max(KL(q||p), g),$$

- Minimizing KL does not lead to minimal $\tilde{g}$: KL that is lower than g does not help to further reduce $\tilde{g}$.
- The generalization bound is to minimize empirical loss term (needs sufficient (i.e. not too sparse) parameters of the model) and the sparsity (represented by $KL(q||p)$ and g).
- More sparseness may not be better, e.g. RVM is more sparse than SVM and PCVM (mean of truncated normal distribution is not zero)
- Adequate sparsity is preferred in sparse Bayesian learning.

Outline

Conclusion

- EPCVM makes Bayesian classification more stable with respect to kernel parameters by pointing at the weakness of standard Gaussian prior (used for decades).
- The solution of EPCVM is fully Bayesian by using the Laplace approximation and expectation propagation.
- EPCVM can incrementally choose basis functions into the model by maximizing model evidence, which makes EPCVM computationally more efficient.
- Theoretical analysis for EPCVM and comprehensive empirical analysis.

For Further Reading 1

- (Chen09) H. Chen, P. Tino, and X. Yao, Probabilistic classification vector machines, IEEE Transactions on Neural Networks, vol. 20, pp. 901– 914, 2009.
- (Chen13) H. Chen, P. Tino, and X. Yao, Efficient Probabilistic Classification Vector Machine with Incremental Basis Function Selection, IEEE Transactions on Neural Networks, 2013. Accepted.
- (Tipping01) M. E. Tipping, Sparse bayesian learning and the relevance vector machine, Journal of Machine Learning Research, vol. 1, pp. 211–244, 2001.

For Further Reading 2

- (Tipping03) M. E. Tipping and A. Faul, Fast marginal likelihood maximisation for sparse bayesian models, in Proceedings of the Ninth international workshop on artificial intelligence and statistics, vol.1, no.3, 2003.
- (Meir03) R. Meir and T. Zhang, Generalization error bounds for bayesian mixture algorithms, Journal of Machine Learning Research, vol. 4, pp. 839–860, 2003

Many thanks for your attention!