Bayesian Support Vector Machines for Feature Ranking and Selection

written by Chu, Keerthi, Ong, Ghahramani

Patrick Pletscher
pat@student.ethz.ch

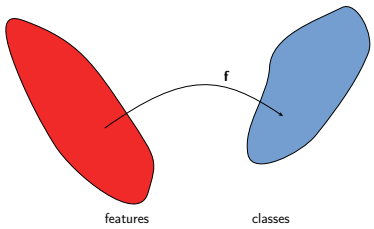
ETH Zurich, Switzerland

12th January 2006

Overview

Our Task: Feature Selection

- Want to learn the *mapping function* between given data (feature vectors) and target values, to *predict* the target value for unseen data.



- As a sub-problem we want to *find the relevant features* first, to then only do the learning on a smaller data set.
- Advantages: decrease computational load and/or increase accuracy.

Before Christmas: Embedded Methods

Guided Search

Embedded methods: Search is guided by a *learning process*. Two advantages compared to wrappers:

- less computationally expensive
- less prone to overfitting

Feature Ranking/Selection with BSVMs

Is as well an embedded method. Ranking done with a BSVMs classifier, for selection any learning algorithm can be used.

Support Vector Machines (1/3)

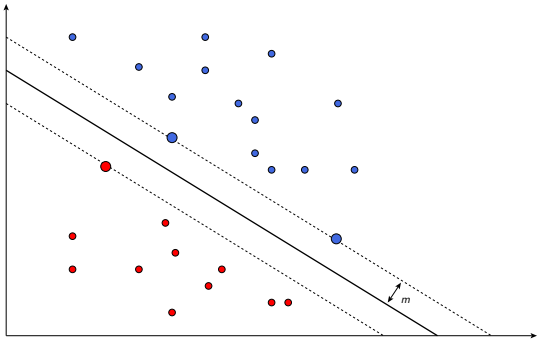


Figure: Schematic illustration of a simple (linear) support vector machine

Support Vector Machines (2/3)

What you should already know

- Sparseness: only support vectors enter the classification rule. Usually much less support vectors than data points.
- The great power of the support vector machines comes from the *kernel trick*: we map our features to some high dimensional feature space and compute the discrimination function there

$$f_{\text{SVM}}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) + \beta_0.$$

Support Vector Machines (3/3)

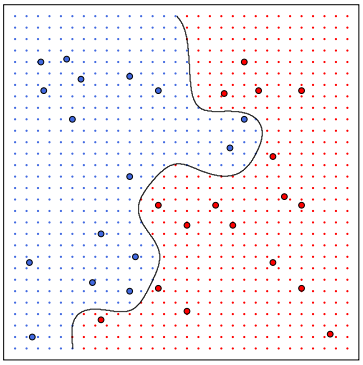


Figure: Radial basis support vector machine

Reproducing Kernel Hilbert Space (1/2)

Kernel trick: Some questions ...

- How does this high dimensional space look like?
- Could one define regularization coefficients in this high-dimensional space?

Answered by the theory of *Reproducing Kernel Hilbert Space* (RKHS), denoted by $\mathcal{H}$ throughout the talk.

RKHS: a smooth Hilbert space

- We don't want to have non-smooth discrimination functions $\mathbf{f}$.
- Hilbert space provides us with all the properties we want, except for the smoothness.
- Idea: restrict the space of possible functions.

Reproducing Kernel Hilbert Space (2/2)

Given some kernel $k(\mathbf{x}, \mathbf{x}')$, we construct a Hilbert space, such that k is a dot product in that space. Define *Gram matrix*. Given data $\mathbf{x}_1, \ldots, \mathbf{x}_m$, define:

$$\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$$

Positive definite $\mathbf{K}$

We get a "good" Hilbert space, if the matrix representation of our kernel is *positive definite*. Theorem of Mercer provides us with a recipe how to construct such kernels.

Reproducing Kernel Hilbert Space

$\mathcal{H}$ is the vector space containing all linear combinations of the functions $k(\cdot\,, \mathbf{x})$:

$$\mathbf{f}(\cdot\,) = \sum_{i=1}^{m} \alpha_i k(\cdot\,, \mathbf{x}_i).$$

Regularized risk for RKHS

Just like in standard linear SVMs we try to minimize

$$\min_{\mathbf{f}\in\mathcal{H}} \mathcal{R}(\mathbf{f}) = \underbrace{\sum_{i=1}^{m} \ell(y_i, f(\mathbf{x}_i))}_{\text{loss}} + \frac{1}{C} \underbrace{\|\mathbf{f}\|_{\mathcal{H}}^2}_{\text{regularizer}}$$

where $\|\cdot\|_{\mathcal{H}}$ is a norm in $\mathcal{H}$. For a linear SVM this $\|\cdot\|_{\mathcal{H}}$ is $\|\mathbf{w}\|_2$, the 2-norm of the parameters that determine the hyperplane. Notice: the kernel is still present implicitly in the loss and regularizer term.

Applying Baysian learning to SVMs (1/2)

Bayesian Philosophy

The *State of Nature* is modelled as a random variable.
For our problem: the mapping function $\mathbf{f}$ is the State of Nature.

Double random process

1. Function $\mathbf{f}$ is drawn from a family of functions (given by $\mathcal{H}$) according to a distribution $P(\mathbf{f})$.
2. Data drawn from $P(\mathcal{D}|\mathbf{f})$.

One problem with this approach: what is correct $P(\mathbf{f})$?

Applying Baysian learning to SVMs (2/2)

Bayes' Theorem

$$\underbrace{P(\mathbf{f}|\mathcal{D})}_{\text{posterior}} \propto \underbrace{P(\mathcal{D}|\mathbf{f})}_{\text{likelihood}} \underbrace{P(\mathbf{f})}_{\text{prior}}$$

Bayesian SVMs: combine SVMs and Bayes theory

Need to define

- prior $P(\mathbf{f})$
- likelihood $P(\mathcal{D}|\mathbf{f})$

Bayesian Philosophy

$\mathbf{f} = \{f(\mathbf{x}_i)\}_{i=1\ldots m}$ is now assumed as a RV.

Combine Bayes' theorem with regularized risk – likelihood

Empirical Risk

Given a term of the form

$$\mathcal{R}_{\text{emp}} = \sum_{i=1}^{m} \ell(y_i, f(\mathbf{x}_i)),$$

which measures the *empiricial risk* (loss on training examples).

Maximum Likelihood

Adapt the *maximum likelihood* philosophy: favour mapping functions with small error on our training sample! Introduce probabilities: small probabilities for big empirical risks, high probabilities for small empirical risks. One comes up with

$$P(\mathcal{D}|\mathbf{f}) \propto \exp\left(-\sum_{i=1}^{m} \ell(y_i, f(\mathbf{x}_i))\right).$$

Combine Bayes' theorem with regularized risk – prior

Gaussian Process

Apply a concept called *Gaussian Process*. We want to favour small values of the regularizer:

$$P(\mathbf{f}) = \exp\left(-\frac{1}{C}\|\mathbf{f}\|_{\mathcal{H}}^2\right)$$

Such a regularizer corresponds to a *weight decay*, which we already have seen in previous lectures. Such a prior is called *Automatic Relevance Determination* (ARD).

Why do we use Gaussians and exp. functions all over again?

- Well-known theory from statistical physics: compare to Boltzmann distribution for energy.
- If one uses a Gaussian for the prior it makes sense to use an exponential function as well for the likelihood, as posterior then gets easier.

Putting it all together

Prior for mapping function has the form

$$P(\mathbf{f}) \propto \exp\left(-\frac{1}{C}\|\mathbf{f}\|_{\mathcal{H}}^2\right)$$

and *likelihood* as

$$P(\mathcal{D}|\mathbf{f}) \propto \exp\left(-\sum_{i=1}^{m} \ell(y_i, f(\mathbf{x}_i))\right)$$

minimizer of regularized functional can then be interpreted as *MAP* (maximum a posteriori):

$$\min_{\mathbf{f}\in\mathcal{H}} \mathcal{R}(\mathbf{f})$$

is equivalent to

$$\max_{\mathbf{f}\in\mathcal{H}} P(\mathcal{D}|\mathbf{f})P(\mathbf{f})$$

Regularizers for Bayesian SVMs (1/3)

2-norm regularization

For the linear SVM we used $\mathbf{w}$ as the regularizer (see Lecture 7). As a short remainder: $\mathbf{w}$ are the parameters of the hyperplane.

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i y_i$$

Regularization of a function $\mathbf{f}$?

How should one define a norm that can be used for the functions in the RKHS. What we want:

- We don't want too complicated functions (one reason: Vapnik-Chernovnenkis theorem).
- The kernel used for the RKHS should influence the regularizer.

Regularizers for Bayesian SVMs (2/3)

Take a covariance matrix as the regularizer!

Compare to the linear SVM (for n features)

$$\|\mathbf{w}\|^2 = \left(\sum_{i=1}^{m} \alpha_i \mathbf{x}_{i,1} y_i\right)^2 + \ldots + \left(\sum_{i=1}^{m} \alpha_i \mathbf{x}_{i,n} y_i\right)^2$$

Feature Selection

General idea: include an ARD (automatic relevance determination) parameter (compare to the scaling parameters of Embedded Methods), and measure the covariance between the outputs corresponding to inputs $\mathbf{x}_i$ and $\mathbf{x}_j$. This is the first time features occur in our formulas (indexed by l).

Regularizers for Bayesian SVMs (3/3)

ARD Linear Kernel

$$\text{Cov}[f(\mathbf{x}_i), f(\mathbf{x}_j)] = \sum_{l=1}^{n} \kappa_{a,l} \mathbf{x}_{i,l} \mathbf{x}_{j,l} + \kappa_b$$

ARD Gaussian Kernel

$$\text{Cov}[f(\mathbf{x}_i), f(\mathbf{x}_j)] = \kappa_0 \exp\left(-\frac{1}{2}\sum_{l=1}^{n} \kappa_{a,l}(\mathbf{x}_{i,l} - \mathbf{x}_{j,l})^2\right) + \kappa_b$$

Prior for RVs $\{f(\mathbf{x}_i)\}$

collect parameters in $\boldsymbol{\theta}$ and let $\boldsymbol{\Sigma}$ denote the $m \times m$ covariance matrix

$$P(\mathbf{f}|\boldsymbol{\theta}) = \frac{1}{Z_{\mathbf{f}}} \exp\left(-\frac{1}{2}\mathbf{f}^T \boldsymbol{\Sigma}^{-1} \mathbf{f}\right).$$

Likelihood for Bayesian SVMs

Assume training data is i.i.d. and likelihood has thus the form

$$P(\mathcal{D}|\mathbf{f},\boldsymbol{\theta}) = \prod_{i=1}^{m} P(y_i|f(\mathbf{x}_i))$$

usually $-\ln P(y_i|f(\mathbf{x}_i))$ is referenced as the loss function $\ell(y_i, f(\mathbf{x}_i))$ (we want a sum instead of the product).

Trigonometric loss function

The authors introduce a new *trigonometric loss function*. Why? As one will have to solve an *optimization problem*, just like in the standard SVM setting. *Sparseness* then helps to make the problem computationally tractable.

$$\ell_t(y_i, f(\mathbf{x}_i)) = \begin{cases} +\infty & \text{if } y_i \cdot f(\mathbf{x}_i) \in (-\infty, -1] \\ 2\ln\sec(\frac{\pi}{4}(1 - y_i \cdot f(\mathbf{x}_i))) & \text{if } y_i \cdot f(\mathbf{x}_i) \in (-1, +1) \\ 0 & \text{if } y_i \cdot f(\mathbf{x}_i) \in [+1, +\infty) \end{cases}$$

Bayesian SVMs – Trigonometric loss function

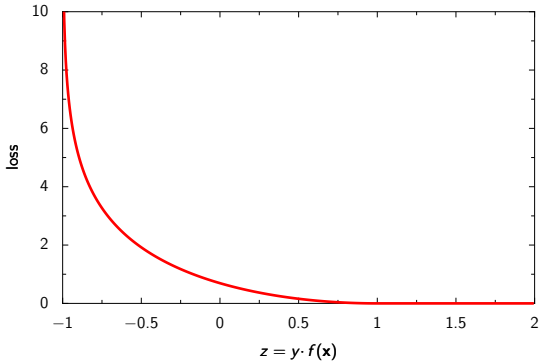


Figure: Trigonometric loss function introduced by Chu et al.

Trigonometric loss – Comparison to other losses

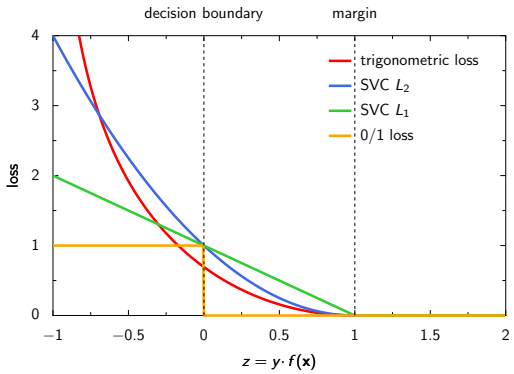


Figure: Comparison of the trigonometric loss function introduced by Chu et al. to other losses proposed in the literature.

Trigonometric likelihood function $P_t(y_i|f(\mathbf{x}_i))$

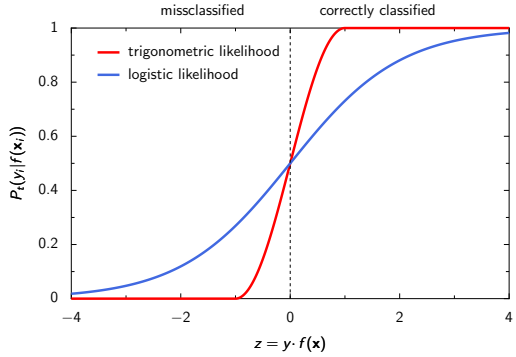


Figure: Comparison of the likelihood implied by the trigonometric loss to the likelihood implied by logistic loss.

Posterior probability and MAP

Based on Bayes' theorem, the posterior probability of $\mathbf{f}$ can be written as

$$P(\mathbf{f}|\mathcal{D},\boldsymbol{\theta}) = \frac{1}{Z_{\mathcal{R}}} \exp\left(-\mathcal{R}(\mathbf{f})\right)$$

where $\mathcal{R}(\mathbf{f}) = \frac{1}{2}\mathbf{f}^T\boldsymbol{\Sigma}^{-1}\mathbf{f} + \sum_{i=1}^{m} \ell(y_i, f(\mathbf{x}_i))$.

MAP estimate on values of $\mathbf{f}$

MAP is minimizer of the following optimization problem

$$\min_{\mathbf{f}} \mathcal{R}(\mathbf{f}) = \frac{1}{2}\mathbf{f}^T\boldsymbol{\Sigma}^{-1}\mathbf{f} + \sum_{i=1}^{m} \ell(y_i, f(\mathbf{x}_i))$$

Solving this optimization problem is similar to the one that arises in standard SVMs.

Hyperparameter Inference

Use again Bayes' theorem

$$P(\boldsymbol{\theta}|\mathcal{D}) = P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})/P(\mathcal{D})$$

The technical details are rather complicated, as it involves solving high dimensional integrals and some gradient descent-like optimization methods. But: one can approximately solve them rather efficient.

What's important: depends on $\mathbf{f}$!

$\mathcal{R}(\mathbf{f})$ from previous slide enters the computation of $P(\mathcal{D}|\boldsymbol{\theta})$, so we really use the information provided by our BSVM classifier.

What have we gained?

We get a parameter vector $\boldsymbol{\theta}$ which contains $\kappa_{a,l}$ the relevance of feature l for $l = 1 \ldots, n$.

Feature Ranking

Extract correctly normalized relevance variables $\{r^i\}_{i=1}^n$ from the ARD parameters

$$r^i = \frac{\kappa_{a,i}}{\sum_{j=1}^n \kappa_{a,j}}$$

Severe problem for inference of $\boldsymbol{\theta}$: local minimas

Gradient-descent like methods can get stucked in local minimas!
Solution: Repeat the minimization several times for random start points and assume underlying distribution of $P(\boldsymbol{\theta}|\mathcal{D})$ is a superposition of Gaussians centered at the different minimas we get.

Posterior $P(\boldsymbol{\theta}|\mathcal{D})$

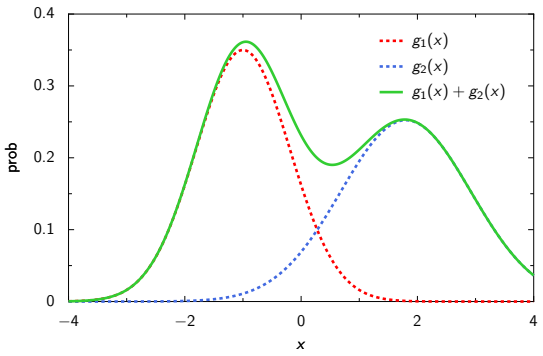


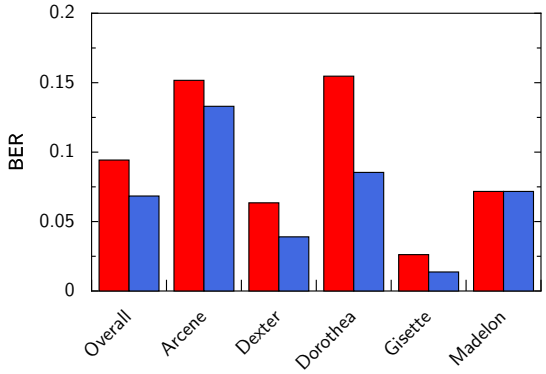
Figure: Reconstructed Posterior

Feature Ranking and Selection

Given some learning algorithm $\mathcal{L}$.

1 initialize validation error to infinity, and $k = 0$;
2 **repeat**
3 | k=k+1;
4 | use the top k features as input vector to $\mathcal{L}$;
5 | carry out cross validation via grid searching on model parameters;
6 | pick up best validation error;
7 **until** *validation error is increasing significantly* ;
8 **return** *the top* $k-1$ *features as the minimal subset*

NIPS 2003 results (1/2)



NIPS 2003 results (2/2)

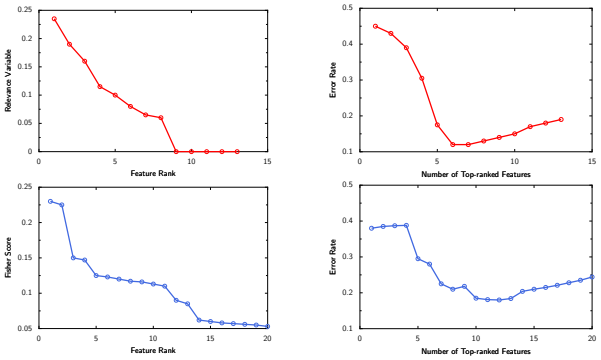


Figure: Results on Madelon data set for BSVM feature ranking (top, red) and fisher score (bottom, blue). Values of relevance variables (left) and validation error rate (right).

Conclusion and take-home message

Conclusion

Some approaches seem interesting, but I as well have some doubts about the method

- Nice: You get posterior probabilites rather than just class labels.
- Seems quite slow (only a vague impression).
- A lot of buzz words: Bayesian Inference, Support Vector Machines, Gaussian process. World formula of feature selection?

Take-home message

One can create an *embedded method* for Support Vector Machines to do feature selection, but at the expense of quite a bit of complexity.

Appendix: Notation & Acknowledgment

Notation

- mapping function $\mathbf{f} := [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m)]^T$.
- N denotes the number of support vectors.
- m denotes the number of data points.
- n denotes the number of features.

Acknowledgment

I want to thank Isabelle Guyon for the immense support while preparing the slides of this talk.