# Kernel Methods


Héctor Corrada Bravo

University of Maryland, College Park, USA


Fannie Mae: 2017-08-04


CBCB Center for Bioinformatics & Computational Biology

University of Maryland

# Kernel Methods

We saw predictive models that depend only on inner products

and that this can be generalized using the "kernel trick"

# Kernel Methods

We saw predictive models that depend only on inner products

and that this can be generalized using the "kernel trick"

1) a similar "trick" can be used for methods other than SVMs,

2) kernels are capable of representing datatypes that are more complex than the tabular representation we have been using so far.

# The SVM as a regularized estimation method

Consider the linear SVM formulation again.

$$\min_{\beta_0,\beta,\xi} C\sum_{i=1}^{N}\xi_i + \frac{1}{2}\|\beta\|^2$$
$$\text{s. t } y_i(\beta_0 + \beta' x_i) \geq 1 - \xi_i \,\forall i$$
$$\xi_i \geq 0 \,\forall i$$

# The SVM as a regularized estimation method

This is equivalent to the following optimization problem with $\lambda = 1/C$

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} (1 - y_i f_i)_+ + \frac{\lambda}{2} \|\beta\|^2$$

- $f_i = \beta_0 + \beta' x_i$
- $(1 - y_i f_i)_+$ denoting the positive part of $1 - y_i f_i$.

# The SVM as a regularized estimation method

$$\min_{\beta_0,\beta} \sum_{i=1}^{N}(1-y_i f_i)_+ + \frac{\lambda}{2}\|\beta\|^2$$

If observation $x_i$ is on the proper side of the margin,

then $y_i f_i > 1$ and thus $(1-y_i fi)_+ = 0$.

# The SVM as a regularized estimation method

$$\min_{\beta_0,\beta} \sum_{i=1}^{N}(1-y_i f_i)_+ + \frac{\lambda}{2}\|\beta\|^2$$

If observation $x_i$ is on the proper side of the margin,

then $y_i f_i > 1$ and thus $(1-y_i fi)_+ = 0$.

Otherwise, $(1-y_i f_i)_+$ equals the signed distance to the margin for observation $x_i$.

# The SVM as a regularized estimation method

$$\min_{\beta_0,\beta} \sum_{i=1}^{N}(1-y_i f_i)_+ + \frac{\lambda}{2}\|\beta\|^2$$

This formulation makes the connection to SVMs as regularized estimation procedure much clearer.

# The SVM as a regularized estimation method

$$\min_{\beta_0,\beta} \sum_{i=1}^{N}(1-y_i f_i)_+ + \frac{\lambda}{2}\|\beta\|^2$$

This formulation makes the connection to SVMs as regularized estimation procedure much clearer.

The first term corresponds to a "loss function"

# The SVM as a regularized estimation method

$$\min_{\beta_0,\beta} \sum_{i=1}^{N}(1-y_i f_i)_+ + \frac{\lambda}{2}\|\beta\|^2$$

This formulation makes the connection to SVMs as regularized estimation procedure much clearer.

The first term corresponds to a "loss function"

The second term a regularization term that controls model complexity.

# The SVM as a regularized estimation method

For the non-linear SVM, recall the discriminant function has form

$$f(x) = \beta_0 + \sum_{i=1}^{N} \alpha_i k(x_i, x)$$

# The SVM as a regularized estimation method

The optimization problem takes a similar form:

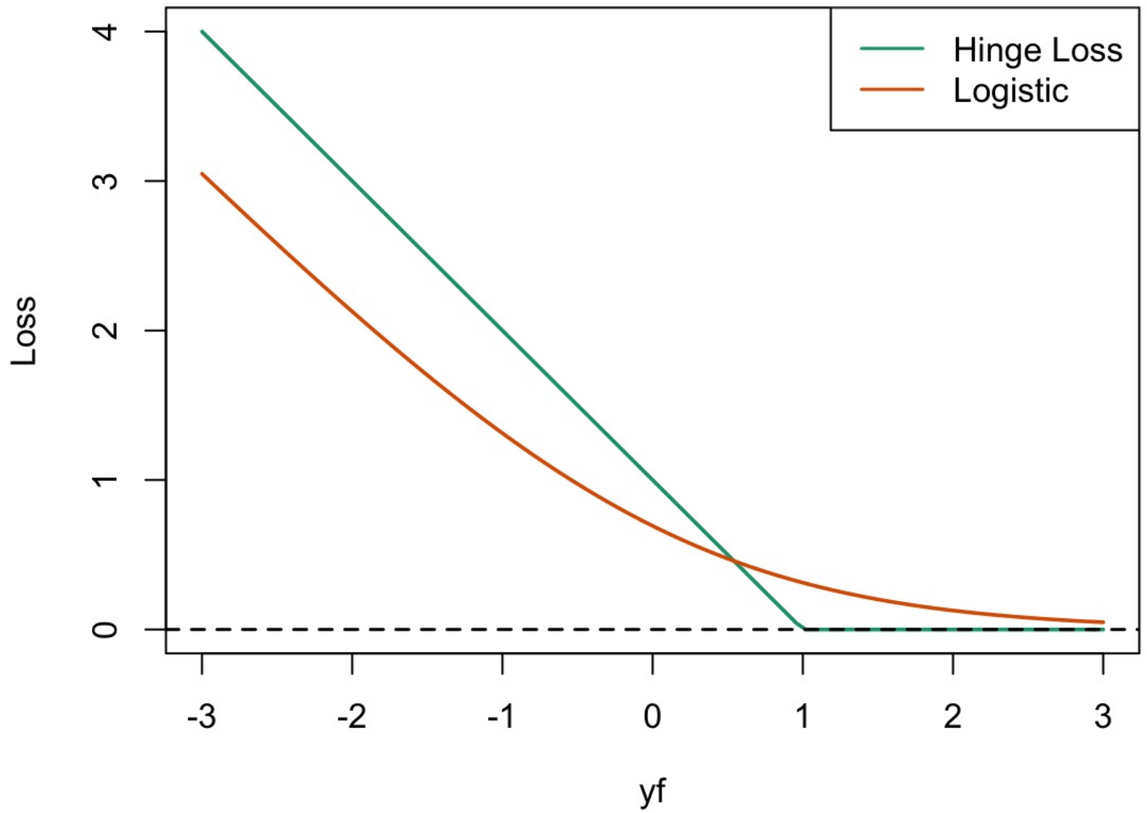
$$\min_{\alpha_0,\alpha} \sum_{i=1}^{N}(1-y_i f_i)_+ + \frac{\lambda}{2}\alpha' K \alpha$$

where $K$ is the $N \times N$ matrix with $K_{ij} = k(x_i, x_j)$.

# Kernelized logistic regression

The loss function in the first term of the formulation is called "Hinge-loss".

We can compare it with the likelihood function for logistic regression.



# Kernelized Logistic Regression

We can therefore use the same "loss + penalty" formulation to obtain a kernelized version of logistic regression:

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \log(1+e^{-y_i f_i}) + \frac{\lambda}{2}\alpha' K \alpha$$

# Kernelized Logistic Regression

As before, function $f$ has a linear expansion in terms of the kernel function:

$$f(x) = \beta_0 + \sum_{i=1}^{N} \alpha_i k(x, x_i)$$

# Kernelized Logistic Regression

As before, function $f$ has a linear expansion in terms of the kernel function:

$$f(x) = \beta_0 + \sum_{i=1}^{N} \alpha_i k(x, x_i)$$

Unlike the SVM, the logistic regression loss function does not tend to set $\alpha_i = 0$ for correctly classified observations.

# Kernelized Logistic Regression

As before, function $f$ has a linear expansion in terms of the kernel function:

$$f(x) = \beta_0 + \sum_{i=1}^{N} \alpha_i k(x, x_i)$$

Nonetheless, function $f$ retains the interpretation in logistic regression

$$f(x) = \frac{Pr(Y = +1|X = x)}{Pr(Y = -1|X = x)}$$

# Kernelized Regression

In similar fashion, we could build non-linear regression models using the "kernel trick" by using least squares as the loss function when predicting continuous outcomes.

$$\min \alpha_0, \alpha \sum_{i=1}^{N} (y_i - f_i)^2 + \frac{\lambda}{2} \alpha' K \alpha$$

# Kernelized Regression

Again, function $f$ has a linear expansion in terms of the kernel function

$$f(x) = \beta_0 + \sum_{i=1}^{N} \alpha_i k(x_i, x)$$

and retains the interpretation as conditional expection

$$f(x) = E[Y|X = x]$$

# Kernelized Regression

This does not lead to sparse representations over a subset of observations like SVMS.

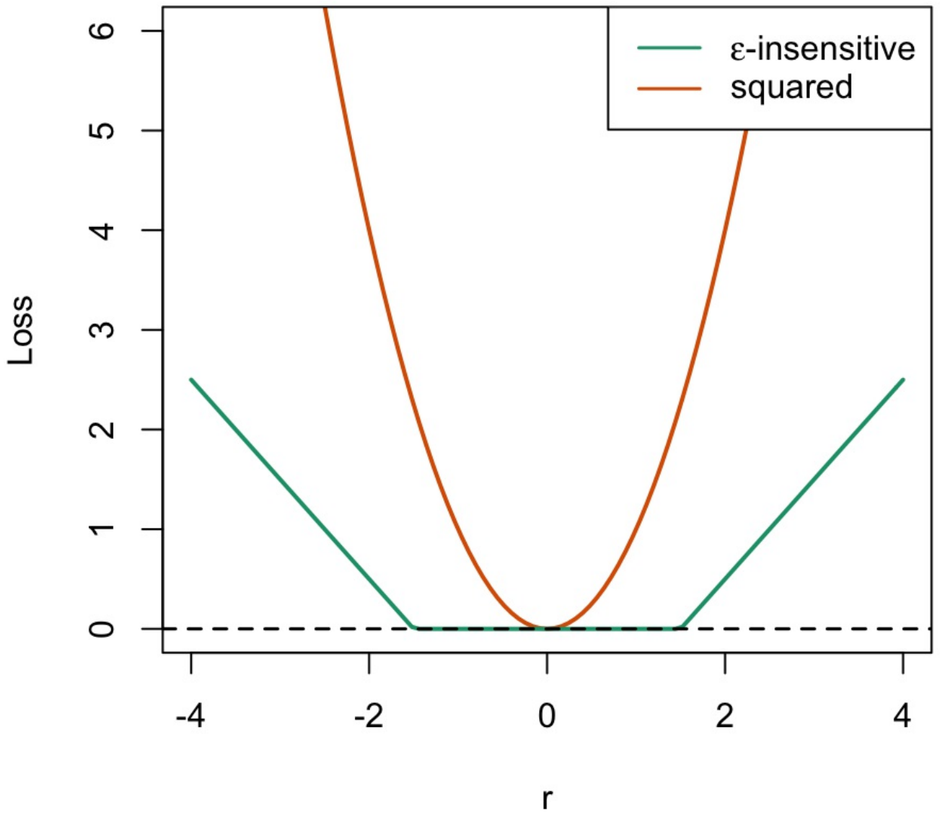
However, a different choice of loss function, similar to hinge loss, can lead to sparse representations.

# Support Vector Regression

Support Vector Regression refers to the "loss + penalty" formulation when $\epsilon$-insensitive loss is used:

$$V_\epsilon(r) = \begin{cases} 0 & \text{if} |r| < \epsilon \\ |r| - \epsilon & \text{otherwise} \end{cases}$$

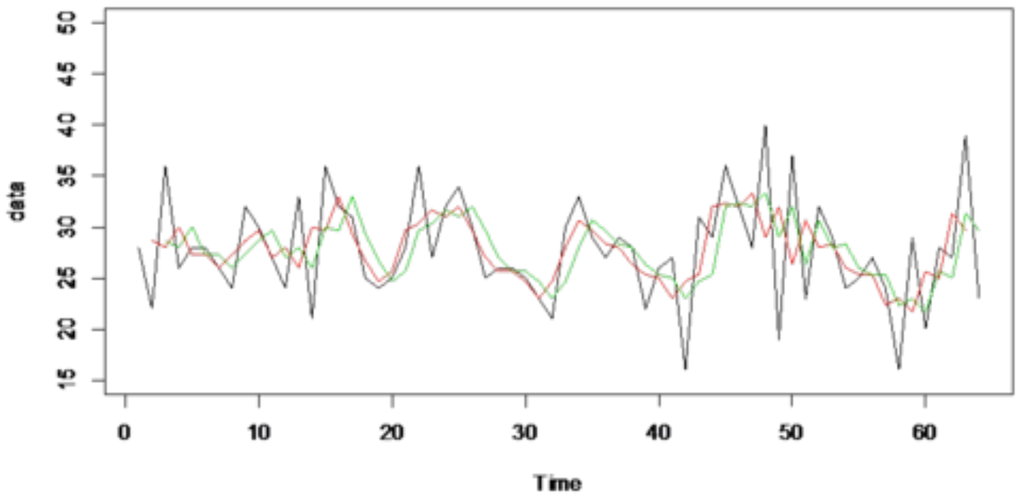
# Support Vector Regression




We can compare `(\epsilon)`-insensitive loss to squared loss

# Modeling labeled data sequences

Consider the case where predictors for observations are structured as sequences.

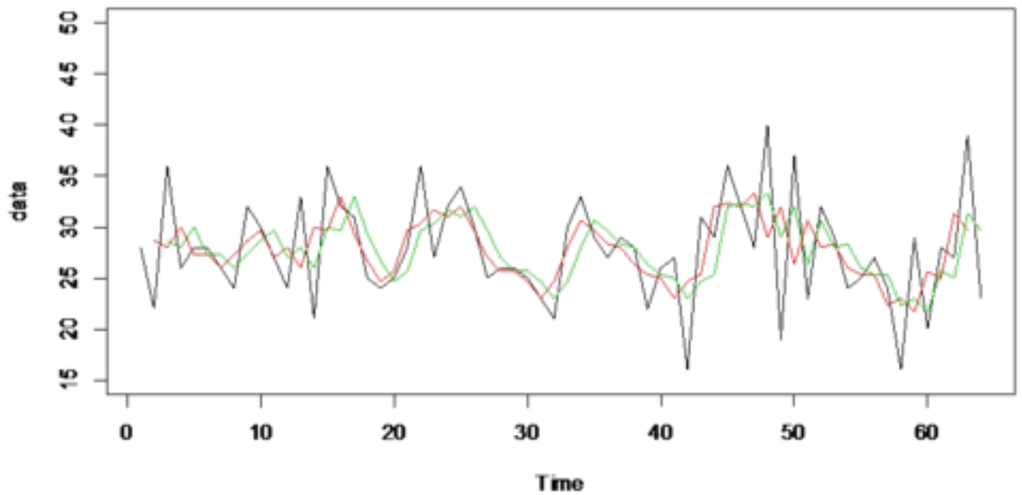



For instance, predictors correspond to some variable measured over time.

# Modeling labeled data sequences

In this case, each observation is represented by a time series

we want to discriminate between time series that belong to two different classes.



# Modeling labeled data sequences

Using the results above, we could model this using a Support Vector Machine, providing a kernel that captures similarity between time series.

Some proposals for this include:

- Autoregressive kernels: Cuturi, Doucet (2011) https://arxiv.org/abs/1101.0673.

The likelihod of a vector autoregressive model is used to create a similarity metric.

# Modeling labeled data sequences

Using the results above, we could model this using a Support Vector Machine, providing a kernel that captures similarity between time series.

Some proposals for this include:

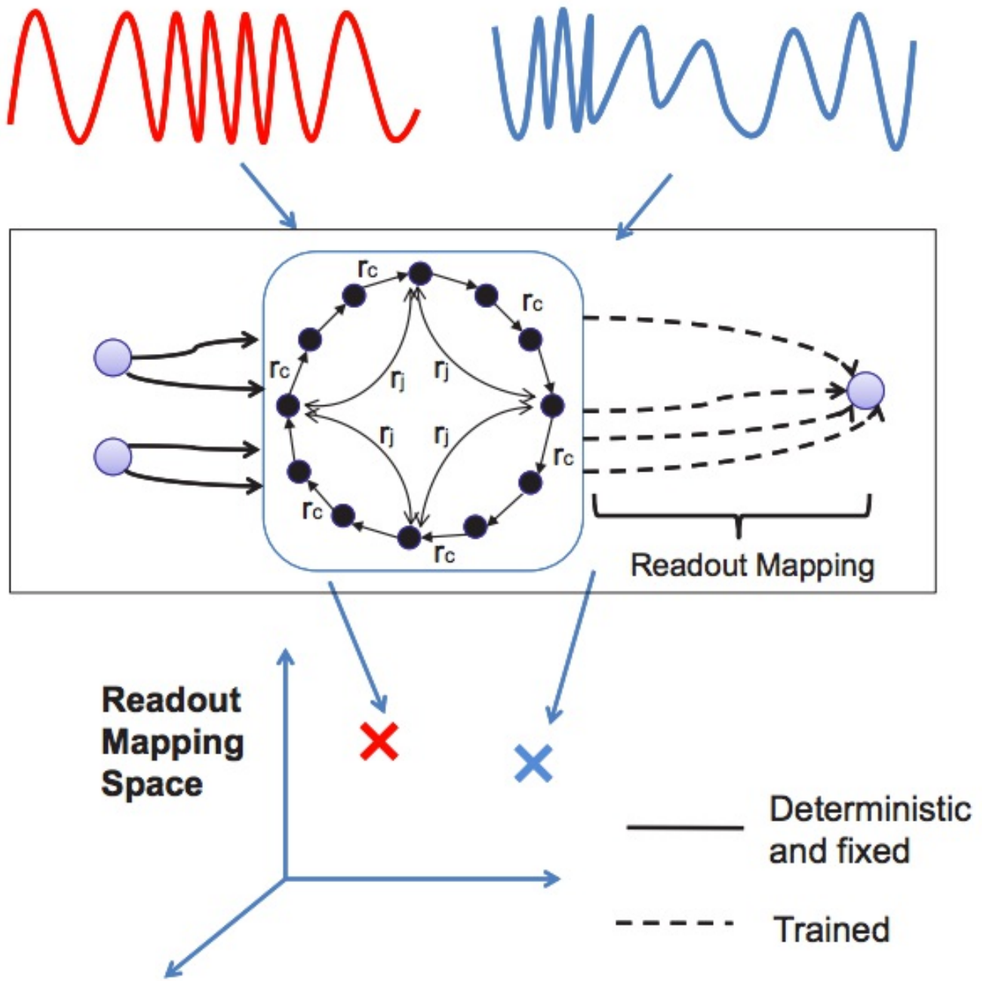
- Dynamic Time Warping Kernel: Shimodaira (2002) https://papers.nips.cc/paper/2131-dynamic-time-alignment-kernel-in-support-vector-machine.pdf.

A warping method is used to define distances between data series

# Modeling labeled data sequences

- Reservoir Computing: Chen et al. http://dl.acm.org/citation.cf id=2487700.

Reservoir state models are used to represent time series and derive kernels



# Structured Output

The "loss + penalty" representation also allows additional flexibility in the types of outcomes that are predicted.

For instance, consider the case where outcomes are numerical vectors

$\mathbf{y_i} = (y_{i1}, y_{i2}, \ldots, y_{iT})$ for each observation

along with predictors $x_i$ as before.

# Structured Output

In this case, we would use function $f$ to represent a vector as well:

$$f(x) = \begin{pmatrix} \alpha_{01} + \sum_{i=1}^{N} \alpha_{i1} k(x_i, x) \\ \\ \alpha_{02} + \sum_{i=1}^{N} \alpha_{i2} k(x_i, x) \\ \\ \vdots \\ \\ \alpha_{0T} + \sum_{i=1}^{N} \alpha_{iT} k(x_i, x) \end{pmatrix}$$

# Structured Output

We can then use a matrix norm on residuals as a loss function

$$\min_{\alpha_0,\alpha} \sum_{i=1}^{N} \|R\|_F^2 + \frac{1}{2} \sum_{j=1}^{T} \alpha_j' K \alpha_j$$

- column $i$ of residual matrix $R_i = y_i - f(x_i)$
- $\|R\|_F^2 = R'R$ is the Frobenius norm of residual matrix $R$
- $\alpha_j$ is the vector of parameters for component $j$ of the model.

# Structured Output

This regression model is over-parameterized

Is difficult to estimate

Does not capture any underlying structure in the outcome vector $Y$.

# Structured Output

Methods to address this shortcoming are generalized under the term "structured output learning".

A good starting point is https://mitpress.mit.edu/books/predicting-structured-data

# Summary

The general "loss + penalty" formulation along with kernel methods are capable of capturing a wide array of learning applications.

# Summary

The general "loss + penalty" formulation along with kernel methods are capable of capturing a wide array of learning applications.

A number of effective methods to represent similarities between data series, e.g., time series, as kernel functions allows the usage of this framework to those types of problems.

# Summary

The general "loss + penalty" formulation along with kernel methods are capable of capturing a wide array of learning applications.

A number of effective methods to represent similarities between data series, e.g., time series, as kernel functions allows the usage of this framework to those types of problems.

Structured output formulations are applicable to learn multivariate outcomes with dependency structure between the components of the outcomes.