COMPSCI 589
Lecture 11: KOLS and Gaussian Process Regression

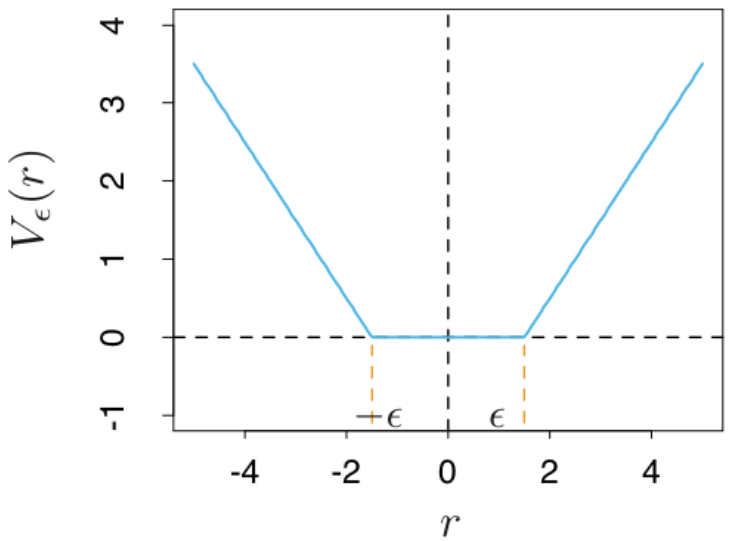
Benjamin M. Marlin

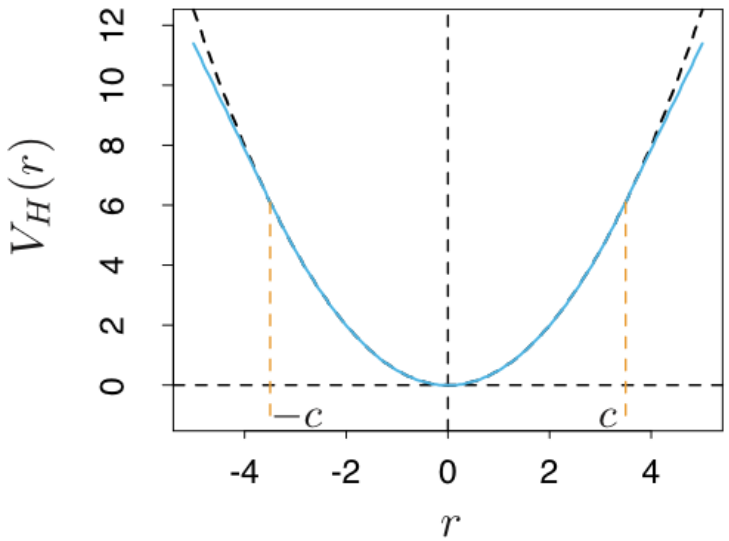
College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).
Created with support from National Science Foundation Award# IIS-1350522.

Support Vector Regression

- Support Vector Regression (SVR) is the generalization of SVMs to the case of regression.
- As with SVMs, SVR is a linear model trained using a different objective function. In this case, the *epsilon insensitive loss*.





Kernelization

Using the same representer theorem used in classification, it can be shown that

$$f_{Lin}(\mathbf{x}) = \mathbf{x}\mathbf{w}^* + b* = \sum_{i=1}^{N} \alpha_i < \mathbf{x}, \mathbf{x}_i > + \sum_{i=1}^{N} \alpha_i < 1, \mathbf{x}_i >$$

This can again be generalized using kernels to allow for non-linear models:

$$f_{Lin}(\mathbf{x}) = \mathbf{x}\mathbf{w}^* + b* = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

Let's have a look at how this model works in more detail.

What About Probabilities?

- An additional drawback of SVR is that it isn't probabilistic.
- For many applications like weather prediction and stock market forecasting, we might like the model to predict both a mean value and a confidence interval or standard deviation.
- **Question:** How can we define a flexible regression model like SVR, but with probabilistic outputs?

OLS Regression as a Probabilistic Model

Recall that the ordinary least squares solution to linear regression is equivalent to optimizing the following probabilistic model where σ^2 is the noise variance.

$$P(y|\mathbf{x}) = \mathcal{N}(y; \mathbf{x}\mathbf{w}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y - \mathbf{x}\mathbf{w})^2\right)$$

Parameter Estimates

The solution for the parameter estimates is:

$$\mathbf{w}_* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{1}$$

$$\sigma_*^2 = \frac{1}{N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n\mathbf{w})^2 \tag{2}$$

Note that learning this model estimates both the regression weights $\mathbf{w}$ and the residual noise variance σ^2.

Kernelized OLS Regression

Just as with SVR, the OLS solution can be written in terms of inner products between data cases and the representer theorem can be used to obtain a kernelized form of the model where:

$$f_{Lin}(\mathbf{x}) = \mathbf{x}\mathbf{w}_* = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^{N} (y_n - f_{Lin}(\mathbf{x}_n))^2$$

Let's have a look at how this model compares to SVR.

KOLS and Uncertainty

Question: What is the problem with the uncertainty estimate coming from KOLS?

It's the same everywhere! We should have more uncertainty far from where we have data and less uncertainty close to where we have data.

Gaussian Process Regression

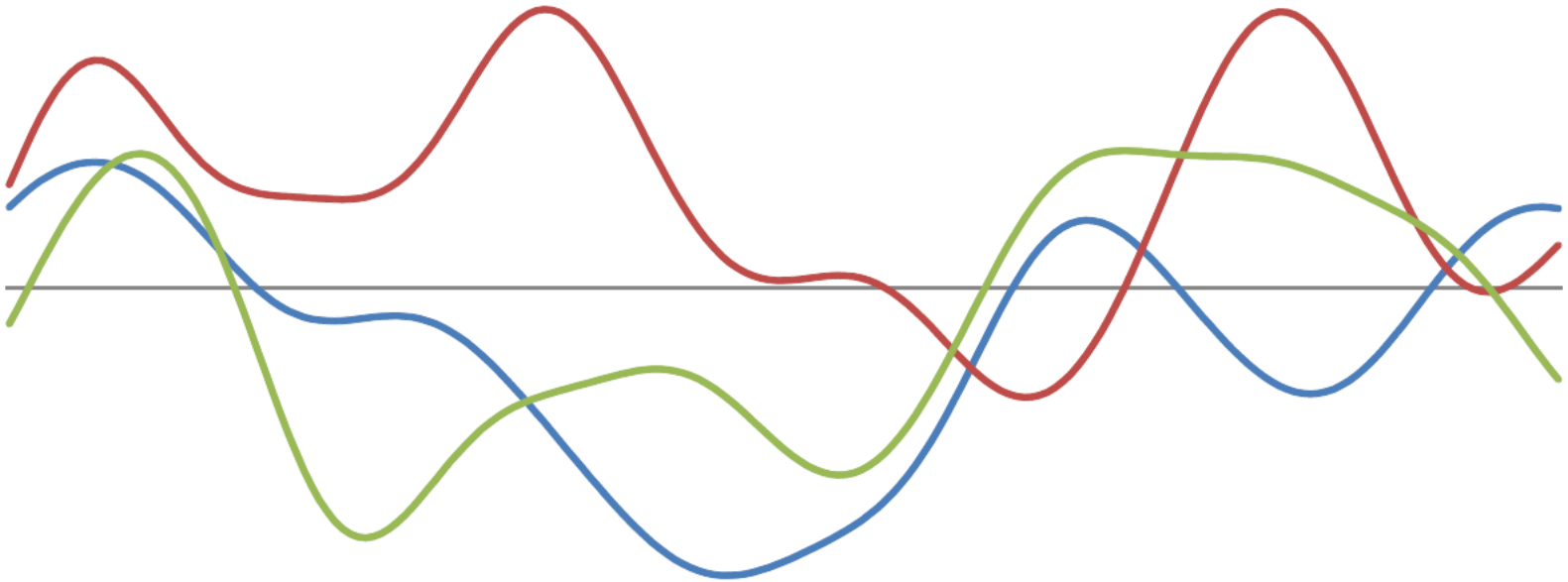
- Gaussian Process Regression is a kernelized probabilistic regression model that infers a probability distribution over latent functions (a *stochastic process*).
- A Gaussian process regression model uses a Gaussian process prior over the space of functions, and a regular normal likelihood.

$$p(f) = \mathcal{GP}(f; m, \mathcal{C}) \qquad p(y|\mathbf{x}, f) = \mathcal{N}(y; f(x), \sigma^2)$$

- A Gaussian process is characterized by a mean function $m(\mathbf{x})$ and a covariance function (kernel function) $\mathcal{C}(\mathbf{x}, \mathbf{x}')$ such that the joint distribution of the process over any finite collection of input $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is multivariate Gaussian.

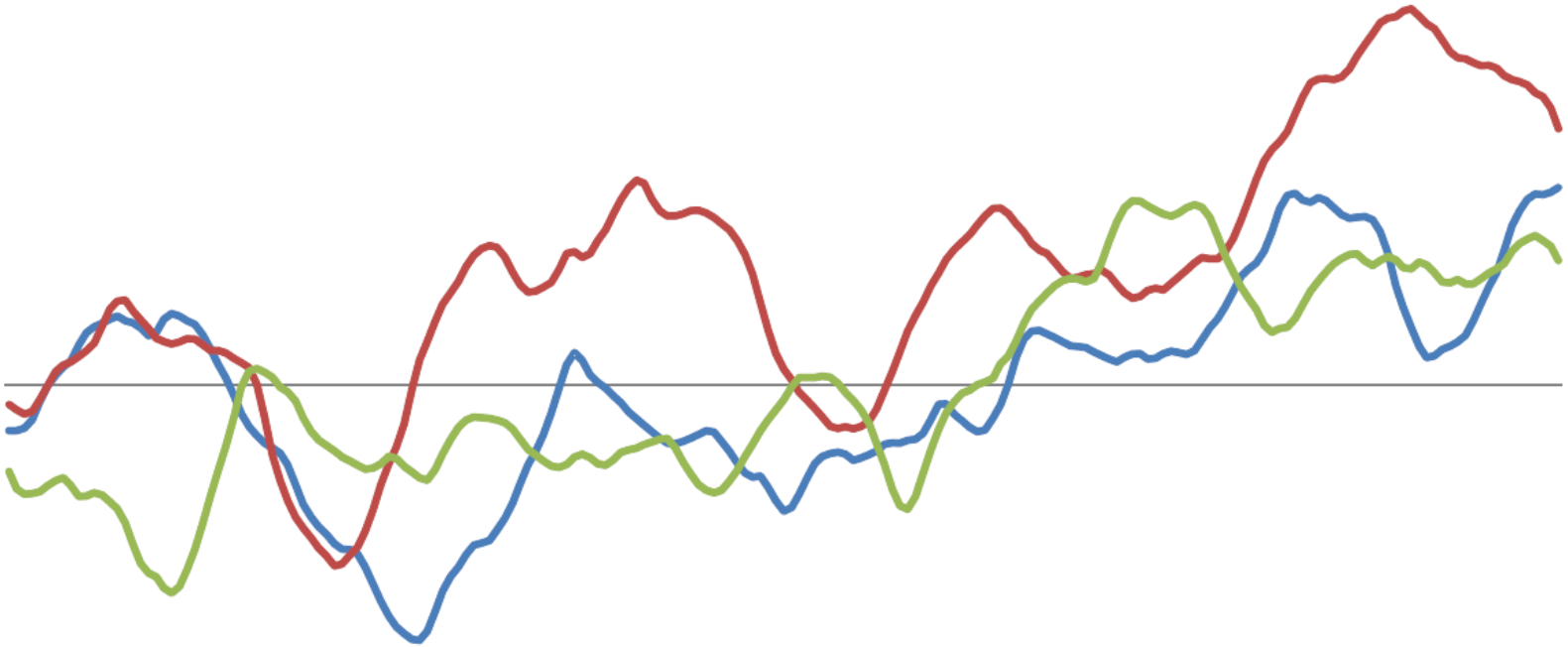
Gaussian Process Regression

The covariance function $\mathcal{C}$ defines how "smooth" typical functions f from the process look. Different covariance functions and different hyper-parameters lead to favoring different types of functions.



Gaussian Process Regression

The covariance function $\mathcal{C}$ defines how "smooth" typical functions f from the process look. Different covariance functions and different hyper-parameters lead to favoring different types of functions.



Gaussian Process Regression

- Intuitively, the model defines the distribution $p(y|\mathbf{x})$ by integrating over the space of all possible functions:

$$p(y|\mathbf{x}) = \int \mathcal{N}(y; f(x), \sigma^2)\mathcal{GP}(f; m, \mathcal{C})df$$

- If we have a data set $\mathcal{D} = \{(y_i, \mathbf{x}_i\}_{i=1:N}$, we can (conceptually) invert this model using Bayes rule to infer $p(f|\mathcal{D})$:

$$p(f|\mathcal{D}) = \mathcal{GP}(f; \mu_N, \mathcal{C}_N)\frac{\prod_{i=1}^N \mathcal{N}(y_i; f(\mathbf{x}_i), \sigma^2)\mathcal{GP}(f; m, \mathcal{C})}{\int \prod_{i=1}^N \mathcal{N}(y_i; f(\mathbf{x}_i), \sigma^2)\mathcal{GP}(f; m, \mathcal{C})df}$$

- We can then make predictions for new points using the equation:

$$p(y|\mathbf{x}, \mathcal{D}) = \int \mathcal{N}(y; f(x), \sigma^2)\mathcal{GP}(f; \mathcal{D})df$$

Gaussian Process Regression

- It may look like the prediction computation $p(y|\mathbf{x}, \mathcal{D})$ is impossible to actually carry out here, but it turns out to be computable in closed form:
- Let $\mathbf{y} = [y_1, ..., y_N]^T$
- Let $\mathbf{m} = [m(\mathbf{x}_1), ..., m(\mathbf{x}_N)]^T$ and $m_* = m(\mathbf{x})$
- Let $\Sigma_{ij} = \mathcal{C}(\mathbf{x}_i, \mathbf{x}_j)$, $(\Sigma_*)_{i,1} = \mathcal{C}(\mathbf{x}_i, \mathbf{x})$ and $(\Sigma_{**}) = \mathcal{C}(\mathbf{x}, \mathbf{x})$.
- We can compute the predictive distribution as follows:

$$
\begin{aligned}
p(y|\mathbf{x}, \mathcal{D}) &= \mathcal{N}(y; \mu_*, \sigma_*^2) \\
\mu_* &= m_* + \Sigma_*^T (\Sigma + \sigma^2 I)^{-1} (\mathbf{y} - \mathbf{m}) \\
\sigma_*^2 &= \Sigma_{**} - \Sigma_*^T (\Sigma + \sigma^2 I)^{-1} \Sigma_*
\end{aligned}
$$

- Let's see what this model can do.

Learning

- Since as a non-parametric Bayesian model, GPR has no explicit model parameters to learn like SVR.
- The covariance function that defines a GP has the same hyper-parameters as the corresponding kernel when used in SVR.
- The hyper-parameters can be set via cross validation. They can also be learned on training data using maximum marginal likelihood optimization.

Trade-Offs

- The computational complexity of GPR scales with cube of the number of input points due to inversion of the covariance matrix. This can be sped up using a number of approximations.
- Unlike SVR and KOLS, GPR provides a more sensible estimate of uncertainty that reflects the amount of data locally.
- The likelihood function is still Gaussian; however, so the model is sensitive to outliers like OLS, and unlike SVR.
- Unlike SVR, the model does not have a support vector property. This means naive GPR implementations must retain access to all data similar to KNN. regression.