Kernelized SVM

Part II

Radial Basis Function (RBF) Kernels

$$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(1)})^2}{2\sigma^2}\right)$$

If $x \approx l^{(1)}$:

If x if far from $l^{(1)}$:

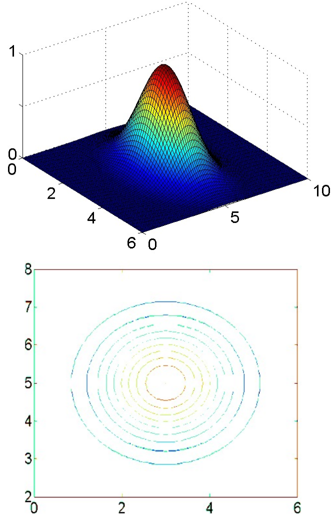
Note: Do perform feature scaling before using the Gaussian kernel.

Radial Basis Function (RBF) Kernels, Cont'd

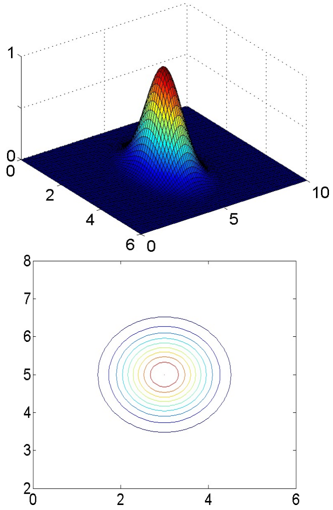
Example:

$$l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \quad f_1 = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$$

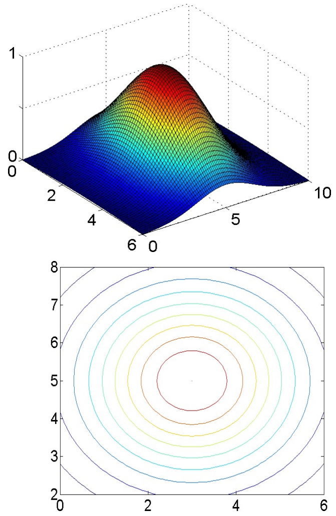
$\sigma^2 = 1$



$\sigma^2 = 0.5$



$\sigma^2 = 3$



SVM With Kernels

Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)}),$
choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \ldots, l^{(m)} = x^{(m)}.$

Given example x :

$$f_1 = \text{similarity}(x, l^{(1)})$$
$$f_2 = \text{similarity}(x, l^{(2)})$$
$$\ldots$$

Hypothesis: Given x, compute features $f \in \mathbb{R}^{m+1}$

Predict “y=1” if $\theta^T f \geq 0$

Training:

$$\min_\theta C \sum_{i=1}^{m} y^{(i)} cost_1(\theta^T f^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

Parameters for RBF Kernels: Part I

Large C: Lower bias, higher variance.

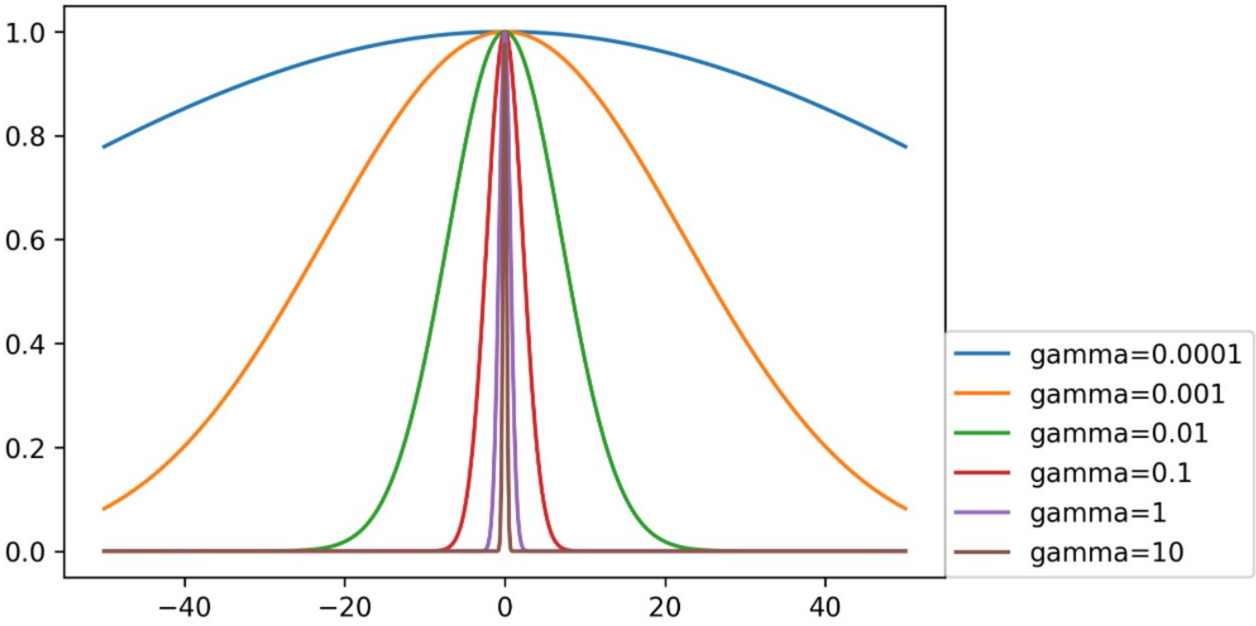
Small C: Higher bias, lower variance.

Large σ^2 : Features f_i vary more smoothly.
Higher bias, lower variance.

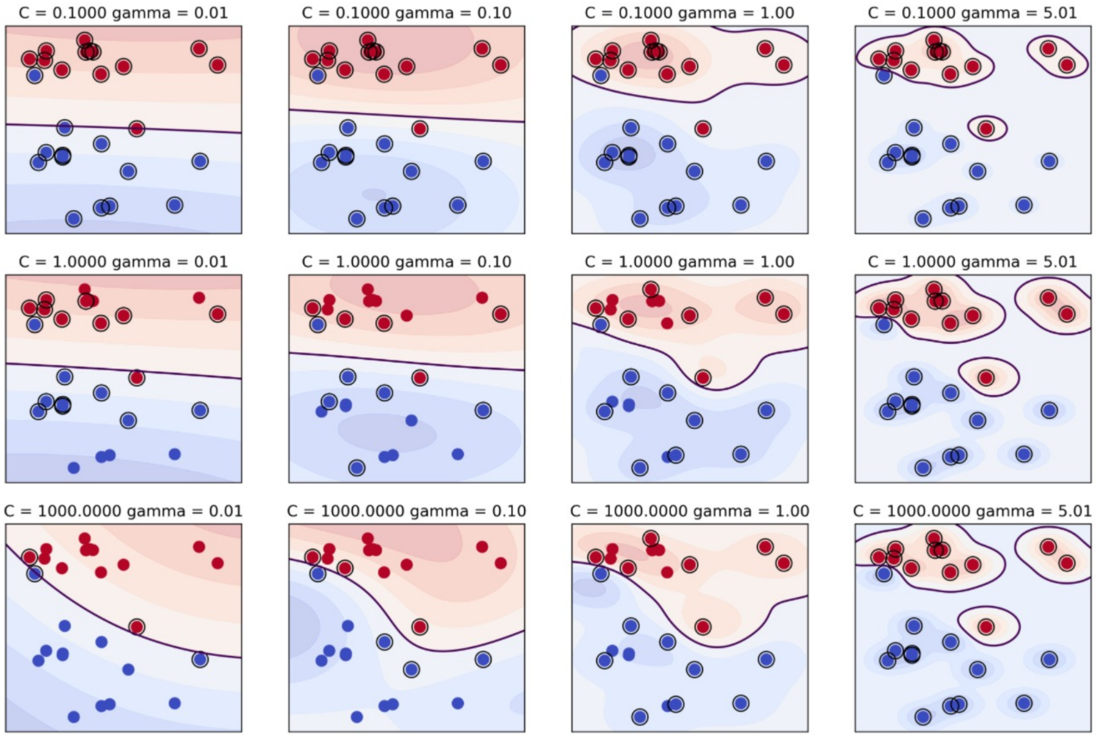
Small σ^2 : Features f_i vary less smoothly.
Lower bias, higher variance.

Parameters for RBF Kernels: Part II

$$k_{\mathrm{rbf}}(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$$



Parameters for RBF Kernels: Part III



Choice of Kernels

Note: Not all similarity functions $\mathrm{similarity}(x, l)$ make valid kernels.
(Need to satisfy technical condition called "Mercer's Theorem" to make sure SVM packages' optimizations run correctly, and do not diverge).

Common kernel functions include:

- **Polynomial Kernel** (kernel='poly') (quadratic kernels d = 2 are commonly used in NLP.

$$K(x_1, x_2) = (x_1^\top x_2 + 1)^d$$

where d is a hyperparameter.

- **Radial Basis Function Kernel** (kernel='rbf')

$$K(x_1, x_2) = \exp\left\{-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right\}$$

where σ is a hyperparameter.

- **Sigmoid Kernel** (kernel='sigmoid') (hyperbolic tangent function, a rescaling of the sigmoid function)

$$K(x_1, x_2) = \tanh(\kappa x_1^\top x_2 + \theta)$$

where κ and θ are hyperparameters.

Kernelized SVM: Pros and Cons

Pros

- Can perform well on a range of datasets
- Versatile: different kernel functions can be specified, or custom kernels can be defined for specific data types.
- Works well for both low-and high-dimensional data.

Cons

- Efficiency (runtime speed and memory usage) decreases as training set size increases (e.g., over 50000 samples).
- Needs careful normalization of input data and parameter tuning.
- Does not provide direct probability estimates.
- Difficult to interpret why a prediction was made.

Logistic Regression Versus SVM

$n =$ number of features ($x \in \mathbb{R}^{n+1}$), $m =$ number of training examples

If n is large (relative to m):

Use logistic regression, or SVM without a kernel ("linear kernel")

If n is small, m is intermediate:

  Use SVM with Gaussian kernel

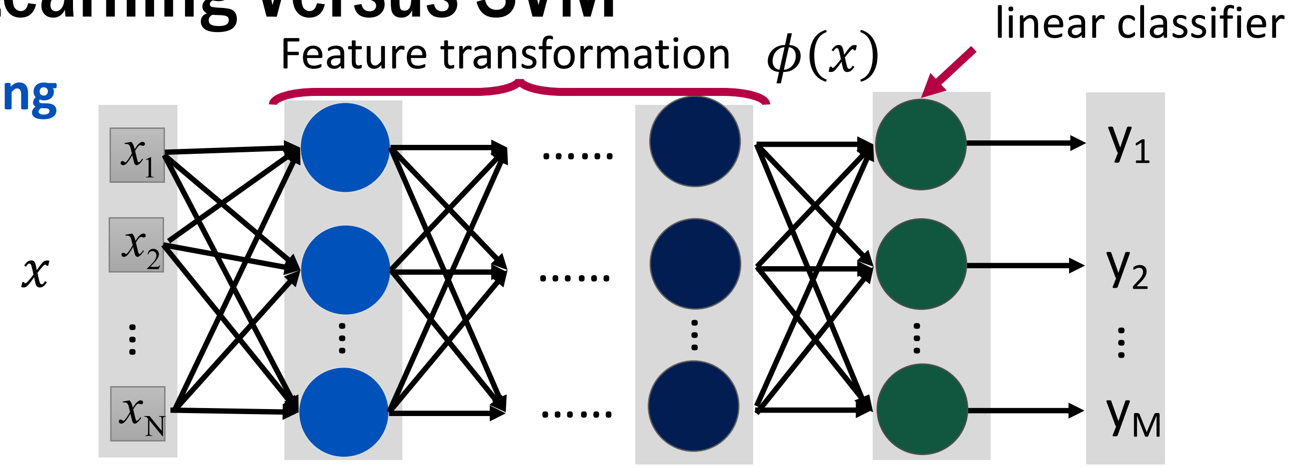
If n is small, m is large:

  Create/add more features, then use logistic regression or SVM without a kernel

Neural network likely to work well for most of these settings but may be slower to train.

Deep Learning Versus SVM

Deep Learning



SVM Based on kernel function $\phi(x)$