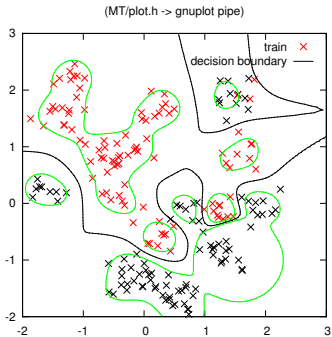
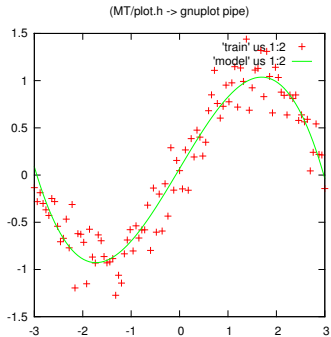


# Machine Learning

## Regression basics

*linear regression, non-linear features  
(polynomial, RBFs, piece-wise), regularization,  
cross validation, Ridge/Lasso, kernel trick*

Marc Toussaint  
U Stuttgart



- Are these linear models? Linear in *what*?
  - Input: No.
  - Parameters, features: Yes!

*Linear Modelling is more powerful than it might seem at first!*

## *Linear Modelling is more powerful than it might seem at first!*

- Linear Regression on non-linear features → very powerful (polynomials, piece-wise, spline basis, kernels)
- Regularization (Ridge, Lasso) & cross-validation for proper generalization to test data
- Gaussian Processes and SVMs are closely related (linear in kernel features, but with different optimality criteria)
- Liquid/Echo State Machines, Extreme Learning, are examples of linear modelling on many (sort of random) non-linear features
- Basic insights in model complexity (effective degrees of freedom)
- Input relevance estimation (z-score) and feature selection (Lasso)
- Linear regression → linear classification (logistic regression: outputs are likelihood ratios)

## *Linear Modelling is more powerful than it might seem at first!*

- Linear Regression on non-linear features → very powerful (polynomials, piece-wise, spline basis, kernels)
- Regularization (Ridge, Lasso) & cross-validation for proper generalization to test data
- Gaussian Processes and SVMs are closely related (linear in kernel features, but with different optimality criteria)
- Liquid/Echo State Machines, Extreme Learning, are examples of linear modelling on many (sort of random) non-linear features
- Basic insights in model complexity (effective degrees of freedom)
- Input relevance estimation (z-score) and feature selection (Lasso)
- Linear regression → linear classification (logistic regression: outputs are likelihood ratios)

⇒ Good foundation for learning about ML

(We roughly follow Hastie, Tibshirani, Friedman: *Elements of Statistical Learning*)

# Linear Regression

- Notation:
  - input vector  $x \in \mathbb{R}^d$
  - output value  $y \in \mathbb{R}$
  - parameters  $\beta = (\beta_0, \beta_1, \dots, \beta_d)^\top \in \mathbb{R}^{d+1}$
  - linear model

$$f(x) = \beta_0 + \sum_{j=1}^d \beta_j x_j$$

# Linear Regression

- Notation:
  - input vector  $x \in \mathbb{R}^d$
  - output value  $y \in \mathbb{R}$
  - parameters  $\beta = (\beta_0, \beta_1, \dots, \beta_d)^\top \in \mathbb{R}^{d+1}$
  - linear model

$$f(x) = \beta_0 + \sum_{j=1}^d \beta_j x_j$$

- Given training data  $D = \{(x_i, y_i)\}_{i=1}^n$  we define the *least squares* cost (or “loss”)

$$L^{\text{ls}}(\beta) = \sum_{i=1}^n (y_i - f(x_i))^2$$

# Optimal parameters $\beta$

- Augment input vector with a 1 in front:

$$\mathbf{x} = (1, x) = (1, x_1, \dots, x_d)^\top \in \mathbb{R}^{d+1}$$

$$\beta = (\beta_0, \beta_1, \dots, \beta_d)^\top \in \mathbb{R}^{d+1}$$

$$f(x) = \beta_0 + \sum_{j=1}^n \beta_j x_j = \mathbf{x}^\top \beta$$



# Optimal parameters $\beta$

- Augment input vector with a 1 in front:

$$\mathbf{x} = (1, x) = (1, x_1, \dots, x_d)^\top \in \mathbb{R}^{d+1}$$

$$\beta = (\beta_0, \beta_1, \dots, \beta_d)^\top \in \mathbb{R}^{d+1}$$

$$f(x) = \beta_0 + \sum_{j=1}^n \beta_j x_j = \mathbf{x}^\top \beta$$

- Rewrite sum of squares as:

$$L^{\text{ls}}(\beta) = \sum_{i=1}^n (y_i - \mathbf{x}_i^\top \beta)^2 = \|\mathbf{y} - \mathbf{X}\beta\|^2$$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ \vdots & & & & \vdots \\ 1 & x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

# Optimal parameters $\beta$

- Augment input vector with a 1 in front:

$$\mathbf{x} = (1, \mathbf{x}) = (1, x_1, \dots, x_d)^\top \in \mathbb{R}^{d+1}$$

$$\beta = (\beta_0, \beta_1, \dots, \beta_d)^\top \in \mathbb{R}^{d+1}$$

$$f(\mathbf{x}) = \beta_0 + \sum_{j=1}^n \beta_j x_j = \mathbf{x}^\top \beta$$

- Rewrite sum of squares as:

$$L^{\text{ls}}(\beta) = \sum_{i=1}^n (y_i - \mathbf{x}_i^\top \beta)^2 = \|\mathbf{y} - \mathbf{X}\beta\|^2$$

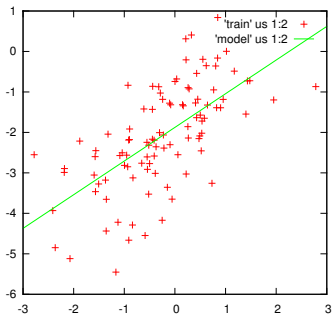
$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ \vdots & & & & \vdots \\ 1 & x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

- Optimum:

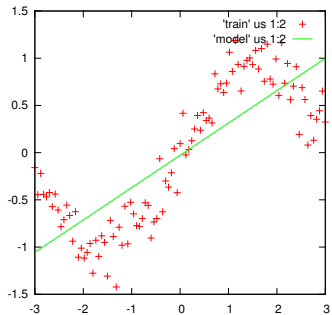
$$\mathbf{0}_d^\top = \frac{\partial L^{\text{ls}}(\beta)}{\partial \beta} = -2(\mathbf{y} - \mathbf{X}\beta)^\top \mathbf{X} \iff \mathbf{0}_d = \mathbf{X}^\top \mathbf{X} \beta - \mathbf{X}^\top \mathbf{y}$$

$$\hat{\beta}^{\text{ls}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

(MT/plot.h -> gnuplot pipe)



(MT/plot.h -> gnuplot pipe)



# Non-linear features

- Replace the inputs  $x_i \in \mathbb{R}^d$  by some non-linear features  $\phi(x_i) \in \mathbb{R}^k$

$$f(x) = \sum_{j=1}^k \phi_j(x) \beta_j = \phi(x)^\top \beta$$

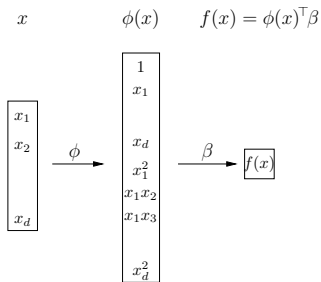
- The optimal  $\beta$  is the same

$$\hat{\beta}^{\text{ls}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \quad \text{but with} \quad \mathbf{X} = \begin{pmatrix} \phi(x_1)^\top \\ \vdots \\ \phi(x_n)^\top \end{pmatrix} \in \mathbb{R}^{n \times k}$$

- What are “features”?
  - a) Features are an arbitrary set of basis functions
  - b) Any function *linear in  $\beta$*  can be written as  $f(x) = \phi(x)^\top \beta$  for some  $\phi$  – which we denote as “features”

# Example: Polynomial features

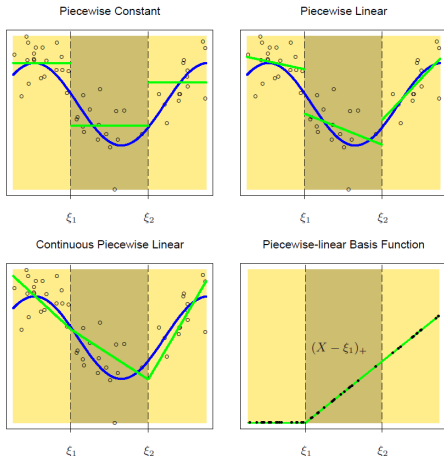
- Linear:  $\phi(x) = (1, x_1, \dots, x_d) \in \mathbb{R}^{1+d}$
- Quadratic:  $\phi(x) = (1, x_1, \dots, x_d, x_1^2, x_1x_2, x_1x_3, \dots, x_d^2) \in \mathbb{R}^{1+d+\frac{d(d+1)}{2}}$
- Cubic:  $\phi(x) = (\dots, x_1^3, x_1^2x_2, x_1^2x_3, \dots, x_d^3) \in \mathbb{R}^{1+d+\frac{d(d+1)}{2}+\frac{d(d+1)(d+2)}{6}}$



```
./x.exe -mode 1 -dataFeatureType 1 -modelFeatureType 1
```

# Example: Piece-wise features

- Piece-wise constant:  $\phi_j(x) = I(\xi_1 < x < \xi_2)$
- Piece-wise linear:  $\phi_j(x) = xI(\xi_1 < x < \xi_2)$
- Continuous piece-wise linear:  $\phi_j(x) = (x - \xi_1)_+$



## Example: Radial Basis Functions (RBF)

- Given a set of centers  $\{c_j\}_{j=1}^k$ , define

$$\phi_j(x) = b(x, c_j) = e^{-\frac{1}{2}\|x-c_j\|^2} \in [0, 1]$$

Each  $\phi_j(x)$  measures similarity with the center  $c_j$

- Special case:

*use all training inputs  $\{x_i\}_{i=1}^n$  as centers*

$$\phi(x) = \begin{pmatrix} 1 \\ b(x, x_1) \\ \vdots \\ b(x, x_n) \end{pmatrix} \quad (n + 1 \text{ dim})$$

This is related to “kernel methods” and GPs, but not quite the same – we’ll discuss this later.

# Features

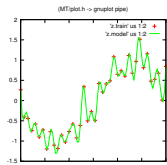
- Polynomial
- Piece-wise
- Radial basis functions (RBF)
- Splines (see Hastie Ch. 5)
  
- Linear regression on top of rich features is extremely powerful!



# The need for regularization

Noisy  $\sin$  data fitted with radial basis functions

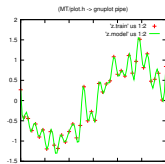
```
./x.exe -mode 1 -n 40 -modelFeatureType 4 -dataType 2 -sigma .3 -lambda  
1e-10
```



# The need for regularization

Noisy `sin` data fitted with radial basis functions

```
./x.exe -mode 1 -n 40 -modelFeatureType 4 -dataType 2 -sigma .3 -lambda  
1e-10
```



- **Overfitting & generalization:**

The model overfits to the data – and generalizes badly

- **Estimator variance:**

When you repeat the experiment (keeping the underlying function fixed), the regression always returns a different model estimate

# Estimator variance

- Assumptions:

- We computed parameters  $\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top y$
- The data was noisy with variance  $\text{Var}(y) = \sigma^2 \mathbf{I}_n$

Then

$$\text{Var}(\hat{\beta}) = (\mathbf{X}^\top \mathbf{X})^{-1} \sigma^2$$

- high data noise  $\sigma \rightarrow$  high estimator variance
  - more data  $\rightarrow$  less estimator variance  $\text{Var}(\hat{\beta}) \propto \frac{1}{n}$
- In practise we don't know  $\sigma$ , but we can estimate it based on the deviation from the learnt model:

$$\hat{\sigma}^2 = \frac{1}{n - d - 1} \sum_{i=1}^n (y_i - f(x_i))^2$$

## Estimator variance

- “Overfitting”
  - ← picking one specific data set  $y \sim \mathcal{N}(y_{mean}, \sigma^2 \mathbf{I}_n)$
  - ↔ picking one specific  $\hat{b} \sim \mathcal{N}(\beta_{mean}, (\mathbf{X}^\top \mathbf{X})^{-1} \sigma^2)$
- If we could reduce the variance of the estimator, we could reduce overfitting – and increase generalization.

Hastie's section on shrinkage methods is great! Describes several ideas on reducing estimator variance — by reducing model complexity. We focus on regularization.

## Ridge regression: $L_2$ -regularization

- We add a *regularization* to the cost:

$$L^{\text{ridge}}(\beta) = \sum_{i=1}^n (y_i - \phi(x_i)^\top \beta)^2 + \lambda \sum_{j=1}^k \beta_j^2$$

NOTE:  $\beta_0$  is usually *not* regularized!

## Ridge regression: $L_2$ -regularization

- We add a *regularization* to the cost:

$$L^{\text{ridge}}(\beta) = \sum_{i=1}^n (y_i - \phi(x_i)^\top \beta)^2 + \lambda \sum_{j=1}^k \beta_j^2$$

NOTE:  $\beta_0$  is usually *not* regularized!

- Optimum:

$$\hat{\beta}^{\text{ridge}} = (\mathbf{X}^\top \mathbf{X} + \lambda I)^{-1} \mathbf{X}^\top \mathbf{y}$$

(where  $I = \mathbf{I}_k$ , or with  $I_{1,1} = 0$  if  $\beta_0$  is not regularized)

- The objective is now composed of two “potential”: The loss, which depends on the data and jumps around (introduces variance), and the regularization penalty (sitting steadily at zero). Both are “pulling” at the optimal  $\beta \rightarrow$  the regularization reduces variance.
- The estimator variance becomes less:  $\text{Var}(\hat{\beta}) = (\mathbf{X}^\top \mathbf{X} + \lambda I)^{-1} \sigma^2$
- The ridge effectively reduces the complexity of the model:

$$\text{df}(\lambda) = \sum_{j=1}^d \frac{d_j^2}{d_j^2 + \lambda}$$

where  $d_j^2$  is the eigenvalue of  $\mathbf{X}^\top \mathbf{X} = \mathbf{V} \mathbf{D}^2 \mathbf{V}^\top$   
 (details: Hastie 3.4.1)



## Choosing $\lambda$ & cross validation

- $\lambda = 0$  will always have a lower *training* data error  
We need to estimate the *generalization* error on test data

## Choosing $\lambda$ & cross validation

- $\lambda = 0$  will always have a lower *training* data error  
We need to estimate the *generalization* error on test data

- *k*-fold cross-validation:



Partition data  $D$  in  $k$  equal sized subsets  $D = \{D_1, \dots, D_k\}$

**for**  $i = 1, \dots, k$  **do**

    compute  $\hat{\beta}_i$  on the training data  $D \setminus D_i$  leaving out  $D_i$

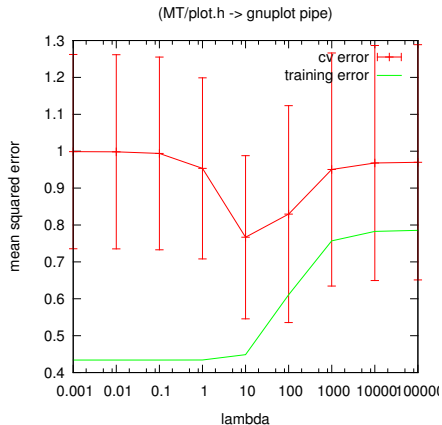
    compute the error  $\ell_i = L^{\text{ls}}(\beta_i, D_i)$  on the validation data  $D_i$

**end for**

report mean error  $\hat{\ell} = 1/k \sum_i \ell_i$  and variance  $(1/k \sum_i \ell_i^2) - \hat{\ell}^2$

- Choose  $\lambda$  for which  $\hat{\ell}$  is smallest

## quadratic features on sinus data:



```
./x.exe -mode 4 -n 10 -modelFeatureType 2 -dataType 2 -sigma .1
```

```
./x.exe -mode 1 -n 10 -modelFeatureType 2 -dataType 2 -sigma .1
```

## Lasso: $L_1$ -regularization

- We add a  $L_1$  regularization to the cost:

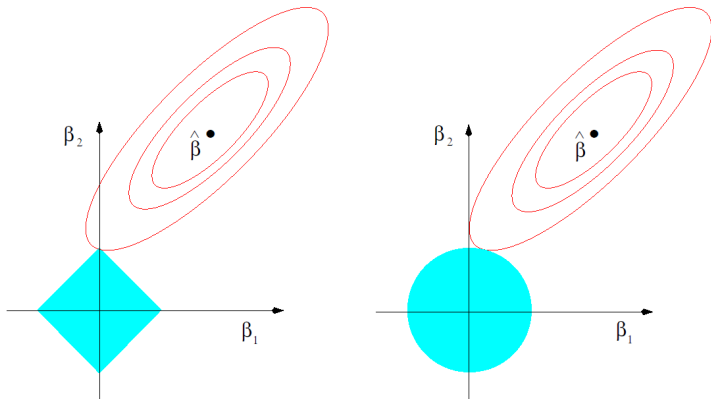
$$L^{\text{lasso}}(\beta) = \sum_{i=1}^n (y_i - \phi(x_i)^\top \beta)^2 + \lambda \sum_{j=1}^k |\beta_j|$$

NOTE:  $\beta_0$  is usually not regularized!

- Has no closed form expression for optimum

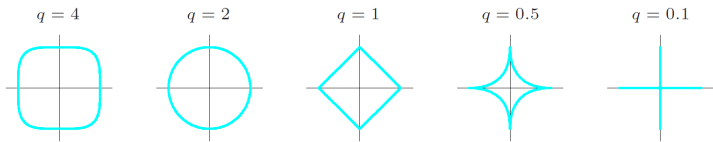
(Optimum can be found by solving a quadratic program; see appendix.)

## Lasso vs. Ridge:



- Lasso  $\rightarrow$  sparsity! feature selection!

$$L^q(\beta) = \sum_{i=1}^n (y_i - \phi(x_i)^\top \beta)^2 + \lambda \sum_{j=1}^k |\beta_j|^q$$



- **Subset selection:**  $q = 0$  (counting the number of  $\beta_j \neq 0$ )

# Summary

- **Linear models** on non-linear features – extremely powerful

linear	Ridge	regression
polynomial	Lasso	classification
RBF		
kernel		

- Generalization  $\leftrightarrow$  **Regularization**  $\leftrightarrow$  complexity/DoF penalty
- **Cross validation** to estimate generalization empirically  $\rightarrow$  use to choose regularization parameters

# Kernel Ridge Regression – the “Kernel Trick”

- Reconsider solution of Ridge regression:

$$\hat{\beta}^{\text{ridge}} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_k)^{-1} \mathbf{X}^T \mathbf{y} = \mathbf{X}^T (\mathbf{X} \mathbf{X}^T + \lambda \mathbf{I}_n)^{-1} \mathbf{y}$$



# Kernel Ridge Regression – the “Kernel Trick”

- Reconsider solution of Ridge regression:

$$\hat{\beta}^{\text{ridge}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_k)^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top (\mathbf{X} \mathbf{X}^\top + \lambda \mathbf{I}_n)^{-1} \mathbf{y}$$

- Recall  $\mathbf{X}^\top = (\phi(x_1), \dots, \phi(x_n)) \in \mathbb{R}^{k \times n}$ , then:

$$f^{\text{ridge}}(x) = \phi(x)^\top \beta^{\text{ridge}} = \underbrace{\phi(x)^\top \mathbf{X}^\top}_{\kappa(x)} (\underbrace{\mathbf{X} \mathbf{X}^\top}_K + \lambda I)^{-1} \mathbf{y}$$

$K$  is called *kernel matrix* and has elements

$$K_{ij} = k(x_i, x_j) := \phi(x_i)^\top \phi(x_j)$$

$\kappa$  is the kernel vector:  $\kappa(x) = k(x, x_{1:n}) = \phi(x)^\top \mathbf{X}^\top$ .

*The kernel function  $k(x, x')$  calculates the scalar product in feature space.*

# The Kernel Trick

- We can rewrite kernel ridge regression as:

$$f^{\text{ridge}}(x) = \kappa(x)(\mathbf{K} + \lambda I)^{-1}\mathbf{y}$$

$$\text{with } \mathbf{K}_{ij} = k(x_i, x_j)$$

$$\kappa_i(x) = k(x, x_i)$$

- at no place we actually need to compute the parameters  $\hat{\beta}$
  - at no place we actually need to compute the features  $\phi(x_i)$
  - we only need to be able to compute  $k(x, x')$  for any  $x, x'$
- This rewriting is called *kernel trick*.
  - It has great implications:
    - Instead of inventing funny non-linear features, we may directly invent funny kernels
    - Inventing a kernel is intuitive:  $k(x, x')$  expresses how correlated  $y$  and  $y'$  should be: it is a measure of similarity, it compares  $x$  and  $x'$ . Specifying how 'comparable'  $x$  and  $x'$  are is often more intuitive than defining "features that might work".

- Every choice of features implies a kernel. But,  
*Does every choice of kernel correspond to specific choice of feature?*

# Reproducing Kernel Hilbert Space

- Let's define a vector space  $\mathcal{H}_k$ , spanned by infinitely many basis elements

$$\{h_x = k(\cdot, x) : x \in \mathbb{R}^d\}$$

Vectors in this space are linear combinations of such basis elements, e.g.,

$$f = \sum_i \alpha_i h_{x_i}, \quad f(x) = \sum_i \alpha_i k(x, x_i)$$

- Let's define a scalar product in this space, by first defining the scalar product for every basis element,

$$\langle h_x, h_y \rangle := k(x, y)$$

This is positive definite. Note, it follows

$$\langle h_x, f \rangle = \sum_i \alpha_i \langle h_x, h_{x_i} \rangle = \sum_i \alpha_i k(x, x_i) = f(x)$$

- The  $\phi(x) = h_x = k(\cdot, x)$  is the 'feature' we associate with  $x$ . Note that this is a function and infinite dimensional.

Choosing  $\alpha = (\mathbf{K} + \lambda I)^{-1} \mathbf{y}$  represents  $f^{\text{ridge}}(x) = \sum_{i=1}^n \alpha_i k(x, x_i)$ , and shows that ridge regression has a finite-dimensional solution in the basis elements  $\{h_{x_i}\}$ . A much more general version of this insight is called **representer theorem**.

# Example Kernels

- Kernel functions need to be positive definite:  $\forall_{z:|z|>0} : k(z, z') > 0$   
→  $\mathbf{K}$  is a positive definite matrix

- Examples:

- Polynomial:  $k(x, x') = (x^\top x')^d$

$$d = 2, x \in \mathbb{R}^2, \phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^\top$$

$$k(x, x') = ((x_1, x_2) \begin{pmatrix} x_1' \\ x_2' \end{pmatrix})^2$$

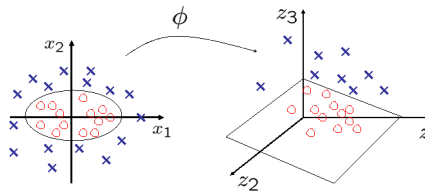
$$= (x_1x_1' + x_2x_2')^2$$

$$= x_1^2x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2x_2'^2$$

$$= (x_1^2, \sqrt{2}x_1x_2, x_2^2)(x_1'^2, \sqrt{2}x_1'x_2', x_2'^2)^\top$$

$$= \phi(x)^\top \phi(x')$$

- Gaussian (radial basis function):  $k(x, x') = \exp(-\gamma |x - x'|^2)$



## Appendix: Alternative formulation of Ridge

- The standard way to write the Ridge regularization:

$$L^{\text{ridge}}(\beta) = \sum_{i=1}^n (y_i - \phi(x_i)^\top \beta)^2 + \lambda \sum_{j=1}^k \beta_j^2$$

- Finding  $\hat{\beta}^{\text{ridge}} = \operatorname{argmin}_{\beta} L^{\text{ridge}}(\beta)$  is equivalent to solving

$$\begin{aligned} \hat{\beta}^{\text{ridge}} &= \operatorname{argmin}_{\beta} \sum_{i=1}^n (y_i - \phi(x_i)^\top \beta)^2 \\ &\text{subject to } \sum_{j=1}^k \beta_j^2 \leq t \end{aligned}$$

$\lambda$  is the Karush-Kuhn-Tucker multiplier for the inequality constraint (generalization of Lagrange multiplier)

## Appendix: Alternative formulation of Lasso

- The standard way to write the Lasso regularization:

$$L^{\text{lasso}}(\beta) = \sum_{i=1}^n (y_i - \phi(x_i)^\top \beta)^2 + \lambda \sum_{j=1}^k |\beta_j|$$

- Equivalent formulation (via KKT):

$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n (y_i - \phi(x_i)^\top \beta)^2$$

subject to  $\sum_{j=1}^k |\beta_j| \leq t$