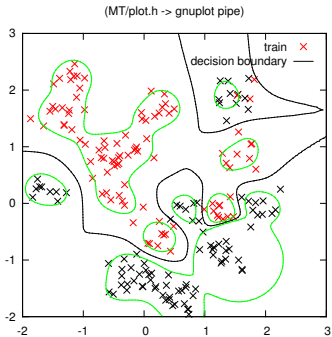
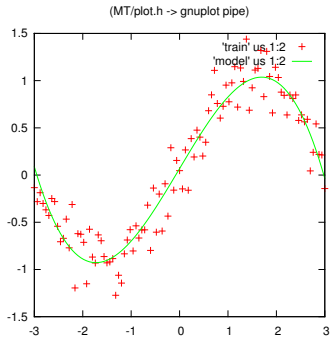


# Machine Learning

## Regression basics

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

Marc Toussaint  
University of Stuttgart  
Summer 2015



- 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)

⇒ 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:

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

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

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

- Rewrite sum of squares as:

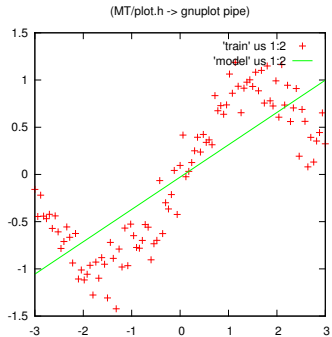
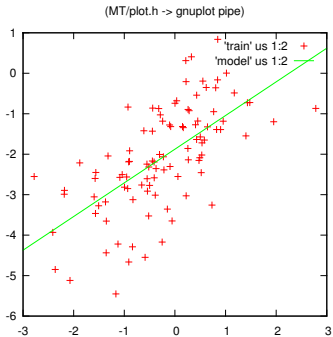
$$L^{\text{ls}}(\beta) = \sum_{i=1}^n (y_i - \mathbf{x}_i^\top \beta)^2 = \|\mathbf{y} - \mathbf{X}\beta\|^2$$

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

- Optimum:

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

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



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

# 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

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

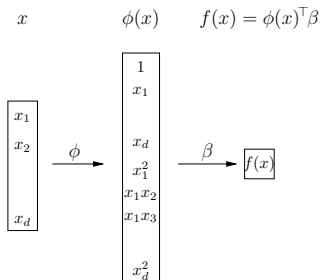
- What are “features”?

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

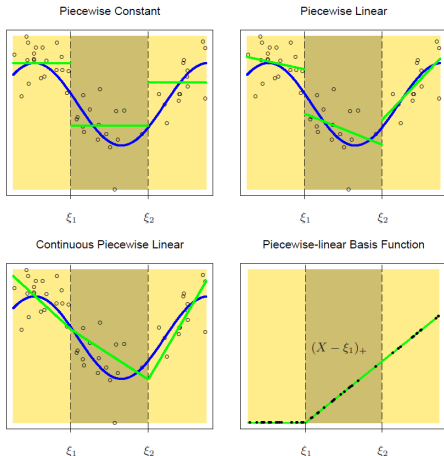
- Linear:  $\phi(x) = (1, x_1, \dots, x_d) \in \mathbb{R}^{1+d}$
- Quadratic:  $\phi(x) = (1, x_1, \dots, x_d, x_1^2, x_1x_2, x_1x_3, \dots, x_d^2) \in \mathbb{R}^{1+d+\frac{d(d+1)}{2}}$
- Cubic:  $\phi(x) = (\dots, x_1^3, x_1^2x_2, x_1^2x_3, \dots, x_d^3) \in \mathbb{R}^{1+d+\frac{d(d+1)}{2}+\frac{d(d+1)(d+2)}{6}}$



```
./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.

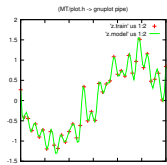
# Features

- 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 -rbfWidth .1  
-sigma .5 -lambda 1e-10
```

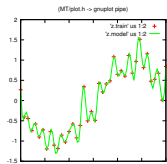




# The need for regularization

Noisy `sin` data fitted with radial basis functions

```
./x.exe -mode 1 -n 40 -modelFeatureType 4 -dataType 2 -rbfWidth .1  
-sigma .5 -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  $\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$
- 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  $y \sim \mathcal{N}(y_{\text{mean}}, \sigma^2 \mathbf{I}_n)$
  - ↔ picking one specific  $\hat{b} \sim \mathcal{N}(\beta_{\text{mean}}, (\mathbf{X}^\top \mathbf{X})^{-1} \sigma^2)$
- 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:

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

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

- The objective is now composed of two “potentials”: 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 $\lambda$ : generalization error & cross validation

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



# Choosing $\lambda$ : generalization error & cross validation

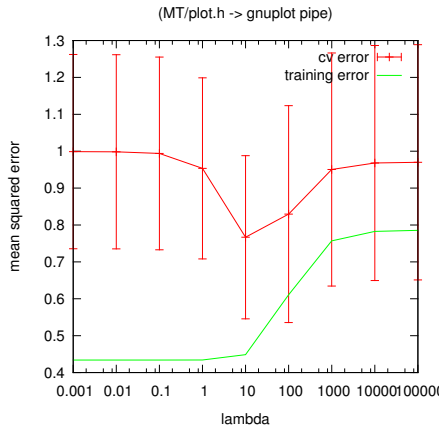
- $\lambda = 0$  will always have a lower *training* data error  
We need to estimate the *generalization* error on test data
- *k-fold cross-validation*:



- 
- 1: Partition data  $D$  in  $k$  equal sized subsets  $D = \{D_1, \dots, D_k\}$
  - 2: **for**  $i = 1, \dots, k$  **do**
  - 3:     compute  $\hat{\beta}_i$  on the training data  $D \setminus D_i$  leaving out  $D_i$
  - 4:     compute the error  $\ell_i = L^{\text{ls}}(\hat{\beta}_i, D_i)$  on the validation data  $D_i$
  - 5: **end for**
  - 6: 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
```

# 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

Representation	Objective		Solver
	error	regularization	
<ul style="list-style-type: none"> <li>• <math>f(x) = \phi(x)^\top \beta</math> (also kernel, SVR, GP, splines)</li> <li>• Neural Networks</li> <li>• Regression Trees (~ tree-features)</li> <li>• lazy learning, local smoothing, <math>k</math>NN</li> </ul>	<ul style="list-style-type: none"> <li>• sqr. error</li> <li>• median sqr. error</li> <li>• robust error functions, Huber, Forsyth</li> <li>• SVM regression loss</li> </ul>	<ul style="list-style-type: none"> <li>• Ridge, <math>L_2, \lambda \ \beta\ ^2</math></li> <li>• Lasso, <math>L_1, \lambda \ \beta\ </math></li> </ul>	<ul style="list-style-type: none"> <li>• analytic <math>\beta^* = (\mathbf{X}^\top \mathbf{X} + \lambda I)^{-1} \mathbf{X}^\top \mathbf{y}</math></li> </ul>

## Appendix: Lasso: $L_1$ -regularization

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

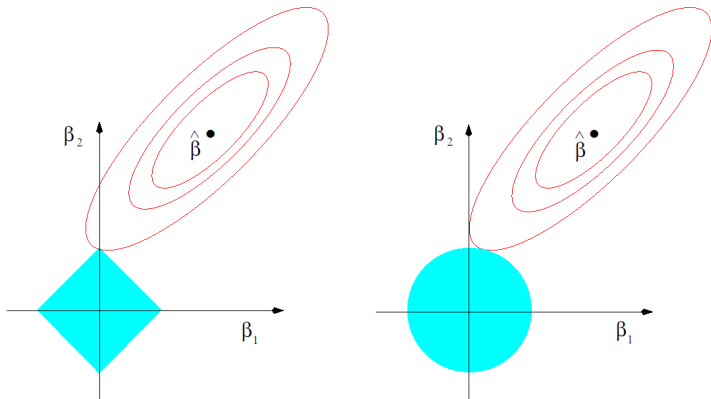
$$L^{\text{lasso}}(\beta) = \sum_{i=1}^n (y_i - \phi(x_i)^\top \beta)^2 + \lambda \sum_{j=1}^k |\beta_j|$$

NOTE:  $\beta_0$  is usually not regularized!

- Has no closed form expression for optimum

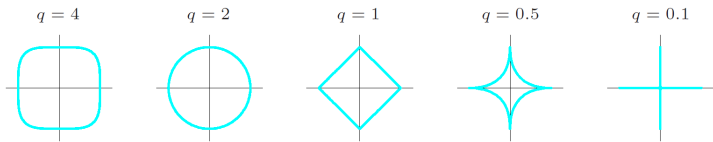
(Optimum can be found by solving a quadratic program; see appendix.)

## 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  $\beta_j \neq 0$ )

## Appendix: Dual 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

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

subject to  $\sum_{j=1}^k \beta_j^2 \leq t$

$\lambda$  is the Lagrange multiplier for the inequality constraint



## Appendix: Dual 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  $\sum_{j=1}^k |\beta_j| \leq t$

- Decreasing  $t$  is called “shrinkage”: The space of allowed  $\beta$  shrinks. Some  $\beta$  will become zero  $\rightarrow$  feature selection