Introduction to Kernels and Regularization

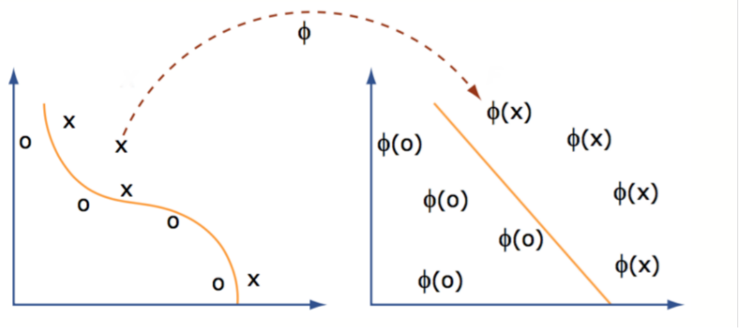
M. Vazirgiannis (presented by Johannes Lutzeyer)

DaSciM, LIX, École Polytechnique

October, 2024

Outline

Mapping

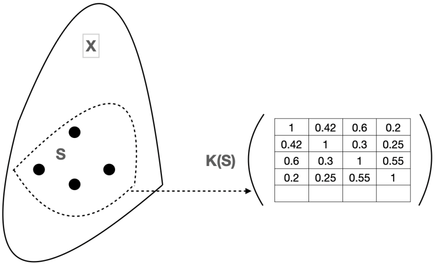


- Map data points into an inner product space *H* with some function $\phi : \phi : x \rightarrow \phi(x) \in H$
- The map ϕ aims to convert the nonlinear relations into linear ones.

Constructing Features

- Problems
  - Need to be an expert in the domain
  - Features may not be adequate
  - Extracting features can sometimes be computationally expensive
    - Example: second order features in 1000 dimensions.
- Solutions
  - Calculate a similarity measure in the feature space instead of the coordinates of the vectors there,
  - apply algorithms that only need the value of this measure

Kernels as distance matrices



1	0.42	0.6	0.2
0.42	1	0.3	0.25
0.6	0.3	1	0.55
0.2	0.25	0.55	1

- Define a “similarity/distance function”: $K : X \times X \to R$
- Represent a set of n data points $S = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}$ by the $n \times n$ matrix: $\mathbf{K}[ij] := K(x_i, x_j)$ where $K(x_i, x_j)$ is the distance/similarity as depicted in K(S)

Kernels

- A kernel is a function $k : X \times X \rightarrow \mathbb{R}$ for which the following property holds

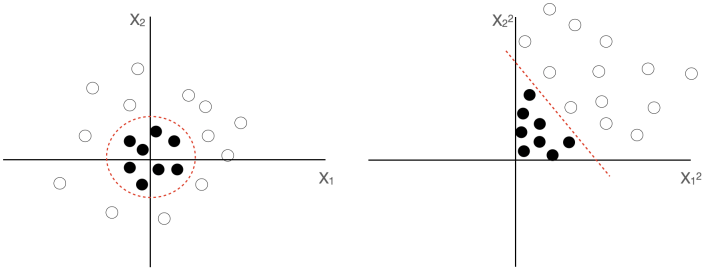
$$k\left(x, x'\right) = < \phi(x), \phi\left(x'\right) >$$

where ϕ is a mapping from X to a Hilbert (inner product) space H

$$\phi : x \rightarrow \phi(x) \in H$$

Kernel Example

- Quadratic Features in $\mathbb{R}^2$



$$\phi : x = (x_1, x_2) \to \phi(x) = \left(x_1^2, x_2^2, \sqrt{2}x_1 x_2\right)$$

- Inner product in the feature space

$$\langle \phi(x), \phi(z) \rangle = \left\langle \left(x_1^2, x_2^2, \sqrt{2}x_1 x_2\right), \left(z_1^2, z_2^2, \sqrt{2}z_1 z_2\right) \right\rangle$$
$$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 x_2 z_1 z_2 = \langle x, z \rangle^2$$

Kernel trick

- enable operation in a high-dimensional, implicit feature space
- without computing the coordinates of the data in that space - $\phi(x)$
- simply computing inner products between the images of all pairs of data in the feature space
- need a function: $k(x, x') = < \phi(x), \phi(x') >$
  - computationally cheaper than the explicit computation of the coordinates.
  - introduced for sequence data, graphs, text, images, as well as vectors.

Properties of a Kernel Matrix

- Symmetric
  - K due to the symmetry of the dot product:

$$K_{ij} = K_{ji} \text{ as } \langle \phi(x), \phi(x') \rangle = \langle \phi(x'), \phi(x) \rangle$$

- K is Positive Semidefinite
  - $a^T K a \geq 0$ for all $a \in \mathbb{R}^n$ and all kernel matrices $K \in \mathbb{R}^{n \times n}$.

Constructing Kernels from Kernels

- Assuming valid kernels $k_1(x,z)$ and $k_2(x,z)$, the following are also valid kernels:
  - $k(x,z) = ck_1(x,z)$, where $c \in \mathbb{R}^+$
  - $k(x,z) = k_1(x,z) + k_2(x,z)$
  - $k(x,z) = k_1(x,z)k_2(x,z)$
  - $k(x,z) = \exp(k_1(x,z))$
  - $k(x,x') = k_a(x_a,x'_a) + k_b(x_b,x'_b)$, where $x = (x_a,x_b)$
  - $k(x,x') = k_a(x_a,x'_a)\,k_b(x_b,x'_b)$, where $x = (x_a,x_b)$

Typical Kernels

- Linear

$$k(x, x') = \langle x, x' \rangle$$

- Laplacian RBF

$$k(x, x') = \exp\left(-\lambda \frac{\|x - x'\|}{\sigma}\right)$$

- Gaussian RBF

$$k(x, x') = \exp\left(-\lambda \frac{\|x - x'\|^2}{\sigma^2}\right)$$

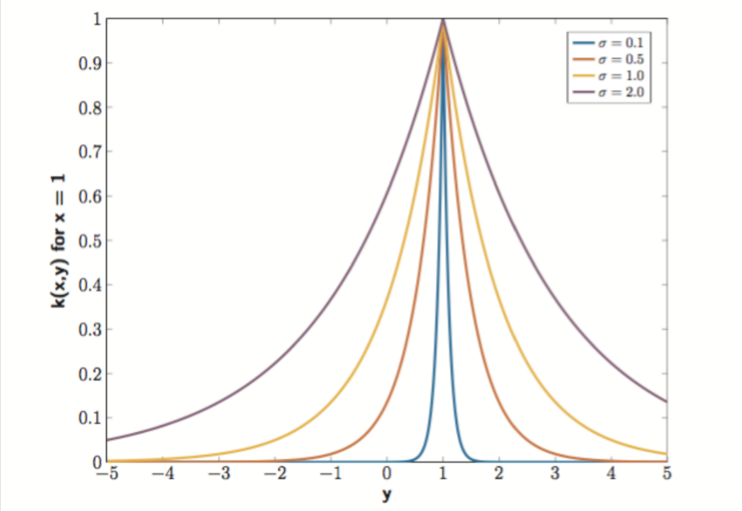
- Polynomial

$$k(x, x') = (\alpha \langle x, x' \rangle + c)^d, \alpha, c \geq 0, d \in \mathbb{N}$$

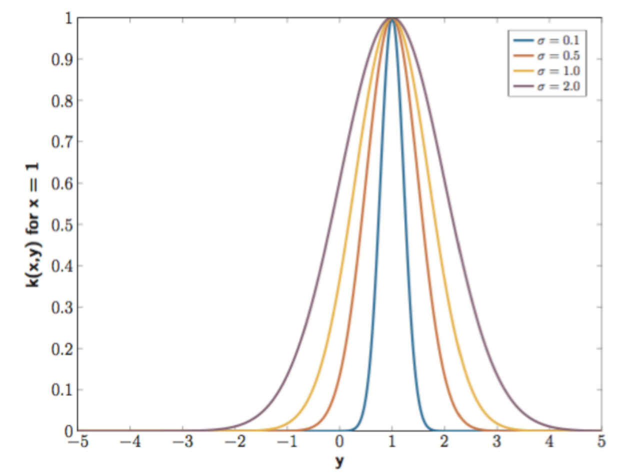
- Sigmoid

$$k(x, x') = \tanh(\alpha \langle x, x' \rangle + b)$$

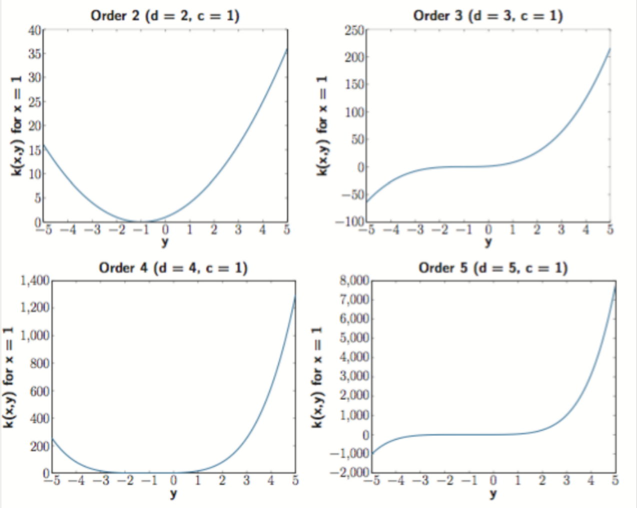
Laplacian Kernel



Gaussian Kernel



Polynomial Kernel



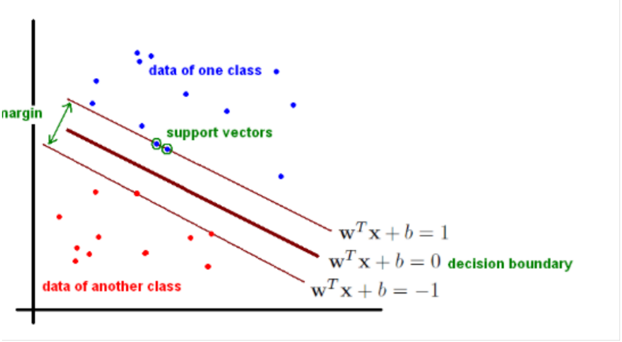
$$k(x, x') = (< x, x' > + c)^d, c \geq 0, d \in \mathbb{N}$$

Outline

SVMs

- Issues that motivated SVMs:
  - bias variance tradeoff
  - Over-fitting
- For a given learning task, a finite amount of training data, the best generalization performance is achieved by jointly optimizing
  - accuracy attained on a training set,
  - “capacity”: ability to learn from any training set without error

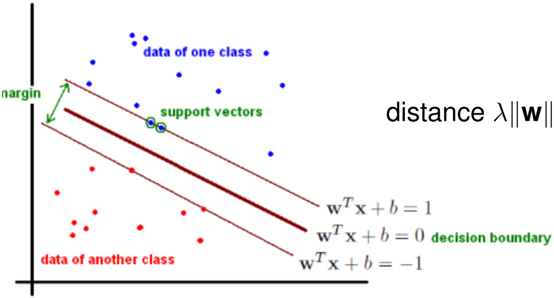
SVMs



- Goal: find a a hyperplane (i.e. decision boundary) linearly separating our classes.
- Boundary equation: $\mathbf{w}^T\mathbf{x} + b = 0$
  - if $x_i : \mathbf{w}^T\mathbf{x} + b > 0$ then $y_i = 1$
  - if $x_i : \mathbf{w}^T\mathbf{x} + b < 0$ then $y_i = -1$
    equivalent: $y(\mathbf{w}^T\mathbf{x} + b) \geq 1$

SVMs – distance between the boundaries

- lines are parallel, with same parameters $\mathbf{w}$, b
- Assume x_1 on $\mathbf{w}^T\mathbf{x} + b = 1$, the closest point of x_2 on line $\mathbf{w}^T\mathbf{x} + b = -1$ and $\lambda\mathbf{w}$ the vector connecting x_1 and x_2.
- Solving for λ : $\mathbf{w}^T\mathbf{x}_2 + b = 1$ where $\mathbf{x}_2 = \mathbf{x}_1 + \lambda\mathbf{w} => \lambda = \frac{2}{\mathbf{w}^T\mathbf{w}} = \frac{2}{\|\mathbf{w}\|^2}$



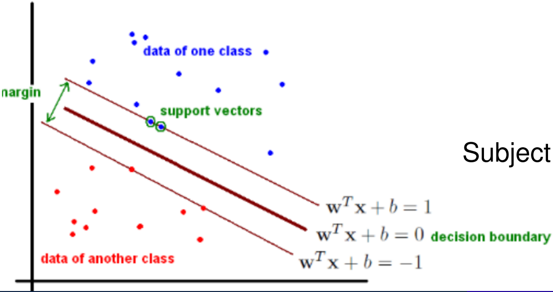
distance $\lambda\|\mathbf{w}\|$ is $\frac{2}{\|\mathbf{w}\|^2}\|\mathbf{w}\| = \frac{2}{\|\mathbf{w}\|} = \frac{2}{\sqrt{\mathbf{w}^{\mathrm{T}}\mathbf{w}}}$

SVMs – optimization formulation

- maximize the distance between the two boundaries defining the classes – to avoid mis-classifications: maximal margin
- Objective :

$$\max \frac{2}{\sqrt{\mathbf{w}^T\mathbf{w}}} \approx \min \frac{\sqrt{\mathbf{w}^T\mathbf{w}}}{2} \approx \min \frac{\mathbf{w}^T\mathbf{w}}{2}$$

- Quadratic formulation problem:



$$\min_{\mathbf{w},b} \frac{\mathbf{w}^T\mathbf{w}}{2}$$

Subject to : $y_i\left(\mathbf{w}^T x_i + b\right) \geq 1, \forall x_i$

Soft Margin extension

- We allow for some misclassification: some data points on the other side of the boundary (slack variables: $\epsilon_i > 0$ for each point x_i).
- The problem becomes:

$$\min_{\mathbf{w},b} \frac{\mathbf{w}^T\mathbf{w}}{2} + C\sum_{I} \epsilon_i$$

$$\text{Subject to: } y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \epsilon_i \quad \& \quad \epsilon_i \geq 0 \qquad \forall \mathbf{x}_i.$$

SVMs – Non Linear Decision Boundary

- If data are not linearly separable we consider a mapping to a higher dimensional space via a function $\phi(\mathbf{x})$. Then the optimization becomes:

$$\min_{\mathbf{w},b} \frac{\mathbf{w}^T\mathbf{w}}{2} + C\sum_{i} \epsilon_i$$

$$\text{Subject to: } y_i\left(\mathbf{w}^T\phi(\mathbf{x}_i) + b\right) \geq 1 - \epsilon_i \quad \& \quad \epsilon_i \geq 0 \qquad \forall \mathbf{x}_i.$$

SVMs – reformulation as a Lagrangian

- Introduce Lagrangian multipliers to represent the condition
- $y_i \left(\mathbf{w}^T \phi\left(\mathbf{x}_i\right) + b\right)$ should be as close to 1 as possible :
- This condition is captured by: $\max_{\alpha_i \geq 0} \alpha_i \left[1 - y_i \left(\mathbf{w}^T \phi\left(\mathbf{x}_i\right) + b\right)\right]$
  - When $y_i \left(\mathbf{w}^T \phi\left(\mathbf{x}_i\right) + b\right) \geq 1$ the expressions is maximal when $\alpha_i = 0$
  - Otherwise $y_i \left(\mathbf{w}^T \phi\left(\mathbf{x}_i\right) + b\right) < 1$, so $1 - y_i \left(\mathbf{w}^T \phi\left(\mathbf{x}_i\right) + b\right)$ is a positive value and the expression is maximal when $\alpha_i \to \infty$
- This results in penalizing (large α_i) misclassified data points, while 0 penalty to properly classified ones
- Thus we have the following formulation:

$$\min_{\mathbf{w}, b} \left[\frac{\mathbf{w}^T \mathbf{w}}{2} + \sum_i \max_{\alpha_i \geq 0} \alpha_i \left[1 - y_i \left(\mathbf{w}^T \phi\left(\mathbf{x}_i\right) + b\right)\right]\right]$$

SVMs – reformulation as a Lagrangian

$\min_{\mathbf{w},b}\left[\frac{\mathbf{w}^T\mathbf{w}}{2}+\sum_i \max_{\alpha_i\geq 0}\alpha_i\left[1-y_i\left(\mathbf{w}^T\phi\left(\mathbf{x}_i\right)+b\right)\right]\right]$

- Preventing α variables to ∞ impose constraints $0\leq\alpha_i\leq C$
- We define the dual problem interchanging the max, min:

$$\max_{\alpha\geq 0}\left[\min_{\mathbf{w},b} J(\mathbf{w},b;\alpha)\right]$$

$$\text{where } J(\mathbf{w},b;\alpha)=\frac{\mathbf{w}^T\mathbf{w}}{2}+\sum_i\alpha_i\left[1-y_i\left(\mathbf{w}^T\phi\left(\mathbf{x_i}\right)+b\right)\right]$$

- To solve the optimization problem: $\frac{\partial J}{\partial \mathbf{w}}=0$ hence $\mathbf{w}=\sum_i\alpha_i y_i\phi\left(x_i\right)$, $\frac{\partial J}{\partial b}=0$ hence $\sum_i\alpha_i y_i=0$. Substitute and simplify:

$$\min_{\mathbf{w},b} J(\mathbf{w},b;\alpha)=\sum_i\alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j\phi\left(\mathbf{x}_i\right)^T\phi\left(\mathbf{x_j}\right)$$

The dual problem is:

$$\max_{\alpha\geq 0}\left[\sum_i\alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j\phi\left(\mathbf{x}_i\right)^T\phi\left(\mathbf{x_j}\right)\right]$$
$$\text{Subject to: } \sum_i\alpha_i y_i=0 \text{ and } 0\leq\alpha_i\leq C$$

SVM - Kernel trick

- As dimensionality may be infinite computation of $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ may be intractable
  Kernel Trick:
  $$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$
- Thus our computation is simplified with rewriting the dual in terms of the kernel:
  $$\max_{\alpha \geq 0} \left[\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right]$$

SVM – Decision function

- To classify a novel instance **x**, having learned the optimal α_i parameters:

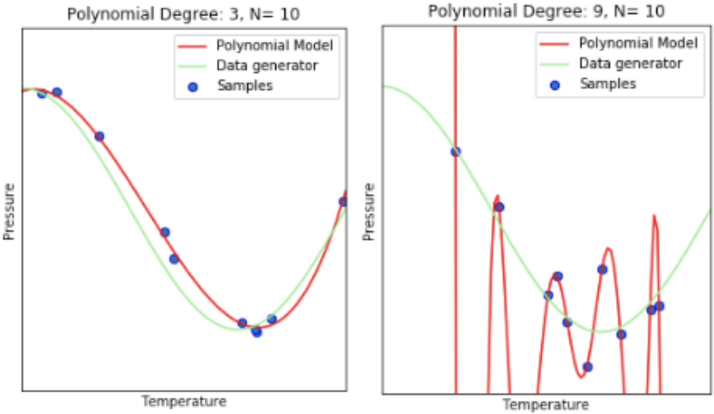
$$f(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\phi(\mathbf{x}) + b\right) = \text{sign}\left(\sum_i \alpha_i y_i K\left(\mathbf{x}_i, \mathbf{x}\right) + b\right),$$

  by setting $\mathbf{w} = \sum_i \alpha_i y_i \phi\left(\mathbf{x}_i\right)$ and using the kernel trick

- α_i are non zero for $\phi(x_i)$ on or close to the boundary – support vectors

Outline

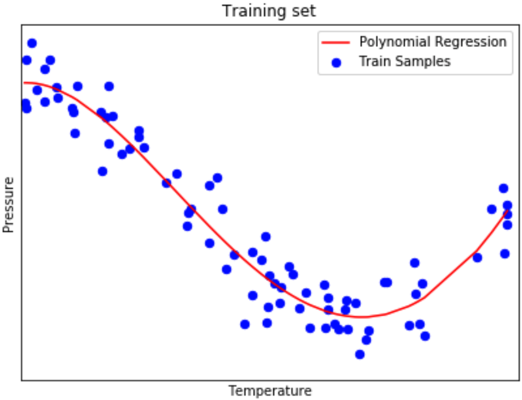
Training vs. Test error



- Model complexity causes overfitting : $\left(\frac{\text{MSE}_{\text{test}}}{\text{MSE}_{\text{train}}}\right)_{degree=3} \leq \left(\frac{\text{MSE}_{\text{test}}}{\text{MSE}_{\text{train}}}\right)_{degree=9}$

https://en.wikipedia.org/wiki/Test_set t

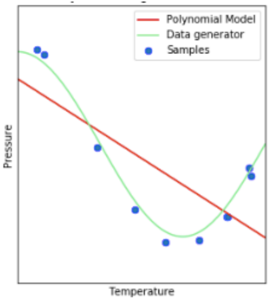
Polynomial Curve fitting

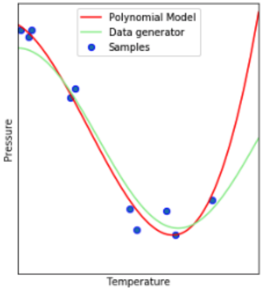


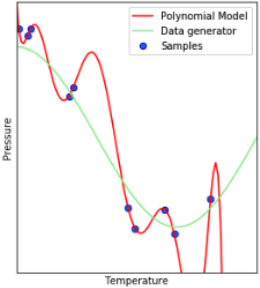
$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

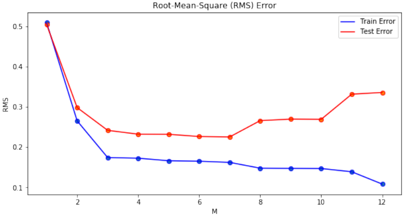
$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \boldsymbol{w}) - t_n\}^2$$

Model complexity vs. Overfitting





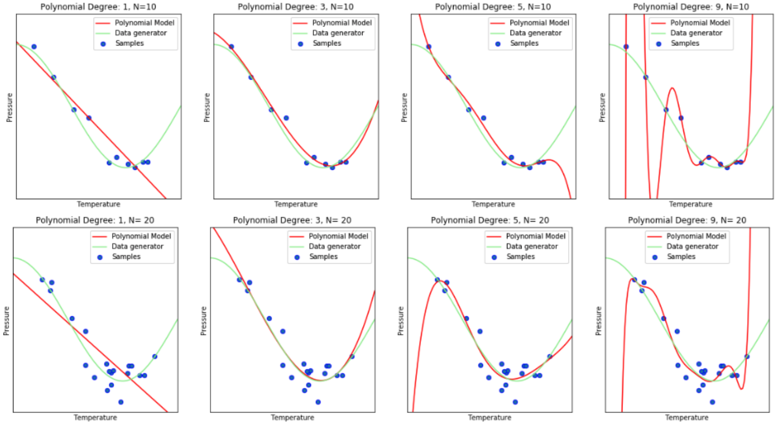




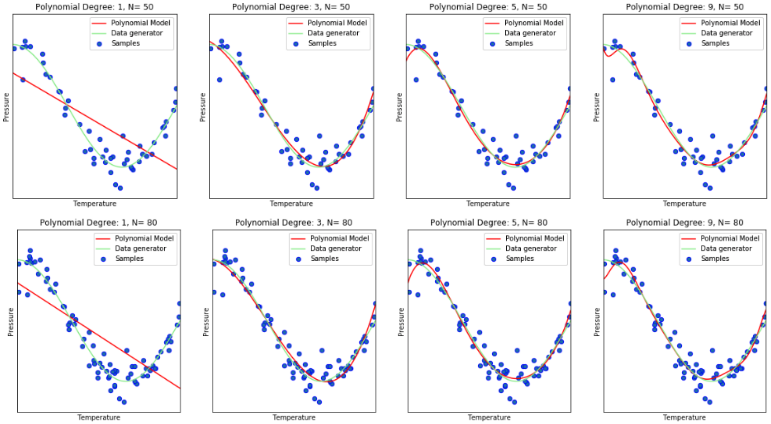
Polynomial coefficients

	M1	M3	M5	...	M9
w_0	-3.16	-1.54	0.57		-50.04
w_1		-12.13	-16.01		1618.22
w_2		13.81	-16.05		-21585.65
w_3			88.54		141927.36
w_4			-61.42		-515139.44
w_5					1083133.64
w_6					-1313771.37
w_7					852910.97
w_8					-229423.36

Dataset size



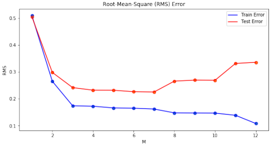
Dataset size



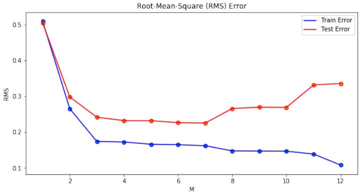
Regularization

- Complex models tend to overfit
- Need to penalize the complexity of the model

$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2 + \frac{\lambda}{2}||\boldsymbol{w}||^2$$



Regularization: Error vs. λ



	M1	M3	M5	...	M9
w_0	-3.16	-1.54	0.57		-50.04
w_1		-12.13	-16.01		1618.22
w_2		13.81	-16.05		-21585.65
w_3			88.54		141927.36
w_4			-61.42		-515139.44
w_5					1083133.64
w_6					-1313771.37
w_7					852910.97
w_8					-229423.36

- A learning algorithm is stable if a small change of the input does not change the output of the algorithm much

Different risk/error types

- Empirical risk: $L_{(S)} = \frac{1}{n}\sum_{i=1}^{N} \delta(f(\boldsymbol{x}), \boldsymbol{y})$
- Expected risk: $L_{(D,f)} = \int_{\mathbb{X},\mathbb{Y}} \delta(f(\boldsymbol{x}), \boldsymbol{y}) p(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y}$

Generalization

- Empirical risk: $L_{(S)} = \frac{1}{n}\sum_{i=1}^{N}\delta(f(\boldsymbol{x}),\boldsymbol{y})$

- Expected risk: $L_{(D,f)} = \int_{\mathbb{X},\mathbb{Y}}\delta(f(\boldsymbol{x}),\boldsymbol{y})p(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{x}d\boldsymbol{y}$

- Generalization error: $G = L_{(D,f)} - L_{(S,f)}$
  - Difference between the *training set* and the *underlying joint probability distribution* error
  - An algorithm generalizes well if

$$\lim_{n\to\infty} G = L_{(D,f)} - L_{(S,f)} = 0$$

$p(\boldsymbol{x},\boldsymbol{y})$ unknown probability distribution $\Rightarrow$ impossible to compute

Regularization

- Regularized Loss Minimization (RLM)
- Minimize the sum of the empirical risk $+$ regularization function
  - measures the complexity of hypotheses/models
  - an interpretation of the regularization function is the structural risk minimization paradigm
- Regularization: stabilizer of the learning algorithm
  - stability: a slight change of its input does not change its output much

Regularized Loss Minimization (RLM)

- Learning rule jointly minimizing the empirical risk $L_s(\boldsymbol{w})$ and penalizing the model complexity via the regularization function $R(\boldsymbol{w})$

$$\arg\min_{\boldsymbol{w}}(L_s(\boldsymbol{w}) + R(\boldsymbol{w}))$$

- The simplest one is $R(\boldsymbol{w}) = \lambda||\boldsymbol{w}||^2$, λ scalar value and $||\boldsymbol{w}|| = \sqrt{\sum w_i^2}$

- Therefore: $A(S) = \arg\min_{\boldsymbol{w}}(L_s(\boldsymbol{w}) + \lambda||\boldsymbol{w}||^2)$

Squared Loss – Ridge regression

- RLM rule with Tikhonov regularization to linear regression with the squared loss, we obtain the following learning rule

$$\arg\min_{\boldsymbol{w}\in\mathbb{R}^d}(\lambda||\boldsymbol{w}||^2 + \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}(<\boldsymbol{w},\boldsymbol{x_i}> - y_i)^2)$$

- Solving for gradient $= 0$ we get:

$$\boldsymbol{w}^{Ridge} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

- $\lambda \geq 0$ is a parameter that controls the amount of weights' shrinkage:
  - the larger the value of λ, the greater the amount of shrinkage. The coefficients are shrunk toward zero.

Squared Loss – Ridge regression

- Equivalent formulation:

$$\arg\min_{\boldsymbol{w}\in\mathbb{R}^d}(\lambda||\boldsymbol{w}||^2 + \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}(<\boldsymbol{w},\boldsymbol{x_i}> - y_i)^2)$$

- subject to $\sum_{i=1}^{m} w_i^2 \leq t$
- We assume that mean value of $y =$ intercept

Regularization of the intercept

- Regression $\boldsymbol{y} = \boldsymbol{w}\boldsymbol{X} \Rightarrow \boldsymbol{y} = w_0 + w_1\boldsymbol{x}_1 + \ldots + w_m\boldsymbol{x}_m$
- $\Rightarrow$ for $\boldsymbol{X} = 0$: $|\boldsymbol{y}| = w_0$
- Regularization based on the idea that overfitting on $\boldsymbol{y}$ is caused by being ”overly specific”,
  - usually resulting in large values of $\boldsymbol{w}$ elements.
- w_0 offsets the relationship: less important to the problem.
  - in case a large offset needed, regularizing it will prevent finding the correct relationship.
- $\boldsymbol{y} = \boldsymbol{w}_{1\ldots M}\boldsymbol{X} + w_0$,
  - $\boldsymbol{w}_{1\ldots M}$ is multiplied with the explaining variables, w_0 added to it.
- Regularizing w_0 may increase bias. . .

Ridge regularization computation

- $\boldsymbol{X}_{N\times(M+1)}$: matrix of input features (N rows, $M+1$ columns, M weights and the intercept)
- $\boldsymbol{y}_N$: the actual outcome variable
- $\hat{\boldsymbol{y}}_N$: predicted values of $\boldsymbol{y}$
- $\boldsymbol{w}_{M+1}$: weights or the coefficients
- For each data point the prediction is $\hat{y}_i = \sum_{j=0}^{M} w_j x_{ij}$
- Cost function to be minimized: RSS (Residual Sum of Squares)

$$\text{Cost}(\boldsymbol{w}) = \text{RSS}(\boldsymbol{w}) = \sum_{i=1}^{N} \{y_i - \hat{y}_i\}^2 = \sum_{i=1}^{N} \left\{ y_i - \sum_{j=0}^{M} w_j x_{ij} \right\}^2$$

Ridge regularization computation

Regression algorithm – with gradient descent:

$\boldsymbol{w} = 0$; η: learning rate;
while *no convergence* **do**
  for *each feature j in* $\{0, 1, \ldots, M\}$ **do**
    determine the gradient;
    $\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} - \eta * \text{gradient}$;
  end
end

Gradient for the j^{th} weight:

$$\frac{\partial}{\partial w_j} = -2 \sum_{i=1}^{N} x_{ij} \left\{ y_i - \sum_{k=0}^{M} w_k x_{ik} \right\}$$

Therefore:

$$w_j^{t+1} = w_j^{t} + 2\eta \sum_{i=1}^{N} x_{ij} \left\{ y_i - \sum_{k=0}^{M} w_k x_{ik} \right\}$$

Ridge regularization computation

- L2 regularization loss function: $\sum_{i=1}^{N}\left\{y_i - \sum_{j=0}^{M} w_j x_{ij}\right\}^2 + \lambda \sum_{j=0}^{M} w_j^2$
- The gradient:

$$\frac{\partial}{\partial w_j} Cost(\boldsymbol{w}) = -2\sum_{i=1}^{N} x_{ij}\left\{y_i - \sum_{k=0}^{M} w_k x_{ik}\right\} + 2\lambda w_j$$

$$w_j^{t+1} = w_j^t - \eta\left[-2\sum_{i=1}^{N} x_{ij}\left\{y_i - \sum_{k=0}^{M} w_k x_{ik}\right\} + 2\lambda w_j^t\right]$$

$$w_j^{t+1} = (1 - 2\lambda\eta) w_j^t + 2\eta \sum_{i=1}^{N} x_{ij}\left\{y_i - \sum_{k=0}^{M} w_k x_{ik}\right\}$$

Ridge regularization computation

- Simple regression update rule:

$$w_j^{t+1} = w_j^t + 2\eta \sum_{i=1}^{N} x_{ij} \left\{ y_i - \sum_{k=0}^{M} w_k x_{ik} \right\}$$

- Ridge regression update rule:

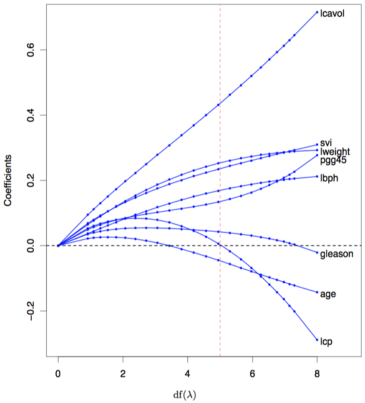
$$w_j^{t+1} = (1 - 2\lambda\eta) w_j^t + 2\eta \sum_{i=1}^{N} x_{ij} \left\{ y_i - \sum_{k=0}^{M} w_k x_{ik} \right\}$$

- Ridge regression:
  - reduces the weights by a factor of $(1 - 2\lambda\eta)$
  - then applies the same update rule as simple linear regression
- Explains why the coefficients reduce to small numbers but never zero

$$A(S) = \arg\min_{\mathbf{w}} (L_S(\mathbf{w}) + \lambda ||\mathbf{w}||^2) \qquad \mathbf{w}^{Ridge} = (\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

Solution is affected by the parameter λ (shrinkage parameter)

- λ controls size of coefficients therefore the amount of **regularization**
- for each λ value – different solution
- $\lambda \to 0$: obtain the un-regularized solution
- $\lambda \to \infty$: $\mathbf{w} = 0$
  - infinite weights lead coefficients to 0



$\lambda \to \infty$ $\qquad\qquad$ $\lambda \to 0$

Lasso regularization

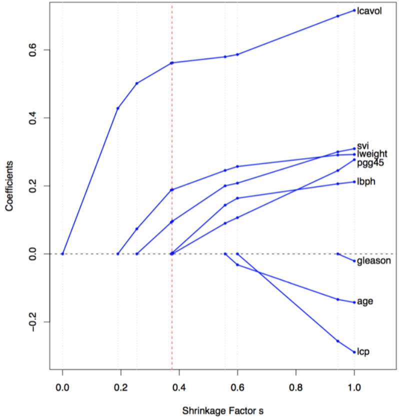
- The Lasso regression [Tibshirani, 1996] is penalizing the sum of absolute values of the weights

$$\boldsymbol{w}^{Lasso} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} (y_i - w_0 - \sum_{j=1}^{m} x_{ij} w_j)^2$$

- Subject to the constraint $\sum_{i=1}^{m} |w_i| \leq t$.
- The constraint makes the solutions nonlinear in the y_i: no closed form expression.
- Computing the lasso solution is a quadratic programming problem.

Lasso regression

- Assume $t_0 = \sum_{i=1}^{m} |w_i|$, w_i the weights produced by the least square solution (i. e. non regularized results)
- for values of $t \leq t_0$ ($t >= 0$), solutions are shrunken versions of the least squares estimates
  - often, some coefficients w_j are zero
- "t" defines the number of predictors to use in a regression model
- t value to minimize expected prediction error – via cross validation



$$s = \frac{t}{\sum_{i=1}^{m} |w_i|}$$

Computation of the Lasso solutions

- A quadratic programming problem; can be tackled by numerical analysis algorithms.

Incremental Forward Stepwise (FS)[1] Regression:

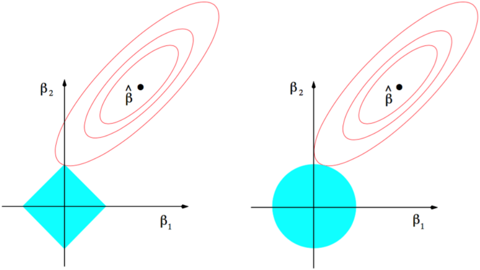
Start with residual $\boldsymbol{r} = \boldsymbol{y} - \bar{\boldsymbol{y}}$ and all $w_j = 0$;
repeat
    Find predictor $\boldsymbol{x}_j$ most correlated with $\boldsymbol{r}$;
    Update $w_j \leftarrow w_j + \delta_j$, where $\delta_j = \epsilon * \text{sign}[\text{corr}(\boldsymbol{r}, \boldsymbol{x}_j)]$;
    Update $\boldsymbol{r} \leftarrow \boldsymbol{r} - \delta_j \boldsymbol{x}_j$
until *no predictor has any correlation with* $\boldsymbol{r}$;

[1] Forward stagewise regression and the monotone lasso, Trevor Hastie, Jonathan Taylor, Robert Tibshirani, Guenther Walther, *Electronic Journal of Statistics*, Vol. 1 (2007) 1–29 ISSN: 1935-7524. https://arxiv.org/pdf/0705.0269.pdf

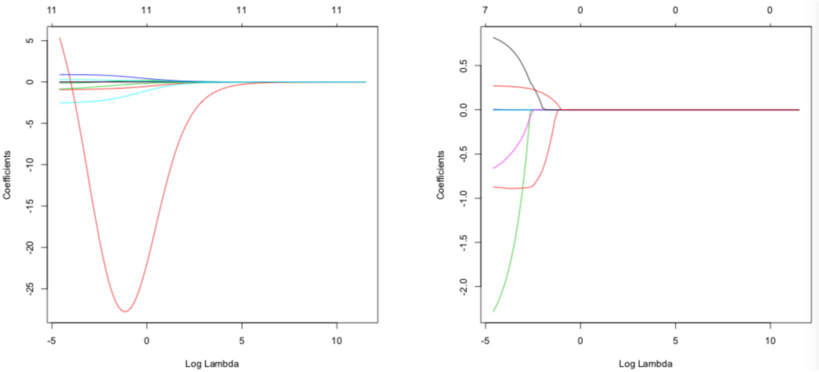
Comparing ridge and lasso

Contours of the error and constraint functions:

- solid blue areas – constraint regions: $|\beta_1| + |\beta_2| \leq t$ and $\beta_1^2 + \beta_2^2 \leq t^2$.
- red ellipses: contours of the least squares error function.



Comparing ridge and lasso



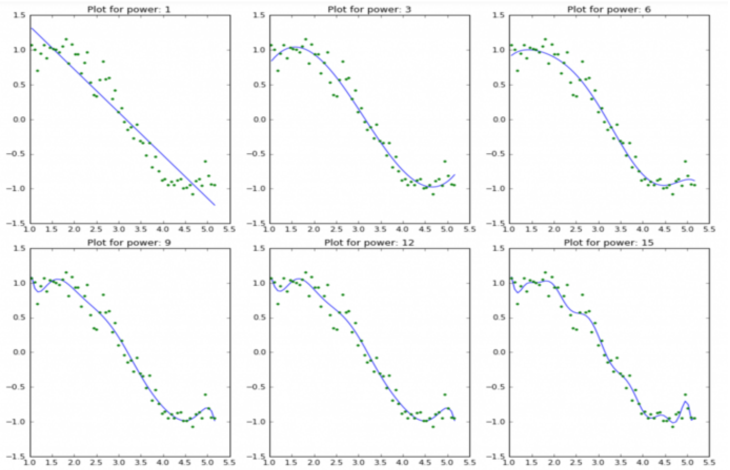
[2]

- variable coefficient estimates: Ridge regression (*left*), LASSO (*right*) for the red wine data plotted versus *log*λ.
- Upper part of the plot shows the number of non-zero coefficients in the regression model for a given *log*(λ).

[2] L.E. Melkumova et al. / Procedia Engineering 201 (2017) 746–755

Ridge regularization effect on regression weights

- Regression: RSS vs. model complexity (polynomial degree)



https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/

Ridge regularization effect on regression weights

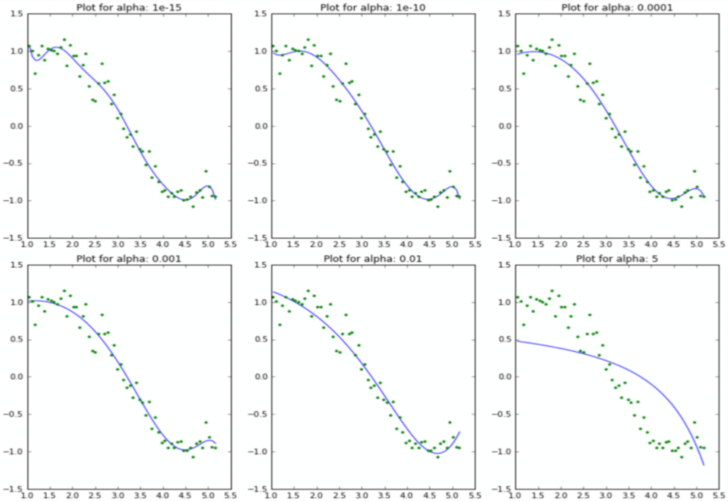
- Regression: coefficient values vs. model complexity (polynomial degree)

	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11	c
model_pow_1	3.3	2	-0.62	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_2	3.3	1.9	-0.58	-0.006	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_3	1.1	-1.1	3	-1.3	0.14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_4	1.1	-0.27	1.7	-0.53	-0.036	0.014	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_5	1	3	-5.1	4.7	-1.9	0.33	-0.021	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_6	0.99	-2.8	9.5	-9.7	5.2	-1.6	0.23	-0.014	NaN	NaN	NaN	NaN	NaN	N
model_pow_7	0.93	19	-56	69	-45	17	-3.5	0.4	-0.019	NaN	NaN	NaN	NaN	N
model_pow_8	0.92	43	-1.4e+02	1.8e+02	-1.3e+02	58	-15	2.4	-0.21	0.0077	NaN	NaN	NaN	N
model_pow_9	0.87	1.7e+02	-6.1e+02	9.6e+02	-8.5e+02	4.6e+02	-1.6e+02	37	-5.2	0.42	-0.015	NaN	NaN	N
model_pow_10	0.87	1.4e+02	-4.9e+02	7.3e+02	-6e+02	2.9e+02	-87	15	-0.81	-0.14	0.026	-0.0013	NaN	N
model_pow_11	0.87	-75	5.1e+02	-1.3e+03	1.9e+03	-1.6e+03	9.1e+02	-3.5e+02	91	-16	1.8	-0.12	0.0034	N
model_pow_12	0.87	-3.4e+02	1.9e+03	-4.4e+03	6e+03	-5.2e+03	3.1e+03	-1.3e+03	3.8e+02	-80	12	-1.1	0.062	-
model_pow_13	0.86	3.2e+03	-1.8e+04	4.5e+04	-6.7e+04	6.6e+04	-4.6e+04	2.3e+04	-8.5e+03	2.3e+03	-4.5e+02	62	-5.7	0
model_pow_14	0.79	2.4e+04	-1.4e+05	3.8e+05	-6.1e+05	6.6e+05	-5e+05	2.8e+05	-1.2e+05	3.7e+04	-8.5e+03	1.5e+03	-1.8e+02	1
model_pow_15	0.7	-3.6e+04	2.4e+05	-7.5e+05	1.4e+06	-1.7e+06	1.5e+06	-1e+06	5e+05	-1.9e+05	5.4e+04	-1.2e+04	1.9e+03	-

https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/

Ridge regularization effect on regression weights

- **Ridge** Regression: RSS vs. λ parameter – shrinkage



https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/

Ridge regularization effect on regression weights

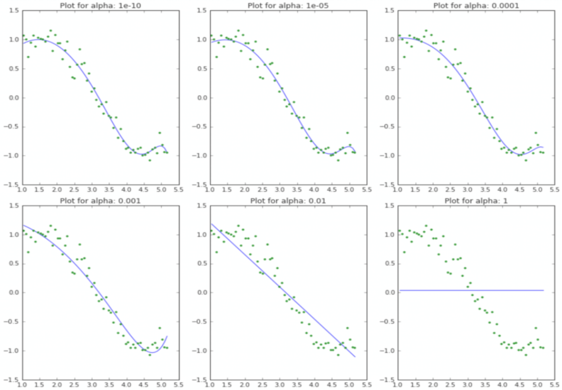
- **Ridge** Regression: coefficient values (regression weights) vs. λ (α in table) parameter shrinkage

	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11	c
alpha_1e-15	0.87	95	-3e+02	3.8e+02	-2.4e+02	66	0.96	-4.8	0.64	0.15	-0.026	-0.0054	0.00086	0
alpha_1e-10	0.92	11	-29	31	-15	2.9	0.17	-0.091	-0.011	0.002	0.00064	2.4e-05	-2e-05	-
alpha_1e-08	0.95	1.3	-1.5	1.7	-0.68	0.039	0.016	0.00016	-0.00036	-5.4e-05	-2.9e-07	1.1e-06	1.9e-07	2
alpha_0.0001	0.96	0.56	0.55	-0.13	-0.026	-0.0028	-0.00011	4.1e-05	1.5e-05	3.7e-06	7.4e-07	1.3e-07	1.9e-08	1
alpha_0.001	1	0.82	0.31	-0.087	-0.02	-0.0028	-0.00022	1.8e-05	1.2e-05	3.4e-06	7.3e-07	1.3e-07	1.9e-08	1
alpha_0.01	1.4	1.3	-0.088	-0.052	-0.01	-0.0014	-0.00013	7.2e-07	4.1e-06	1.3e-06	3e-07	5.6e-08	9e-09	1
alpha_1	5.6	0.97	-0.14	-0.019	-0.003	-0.00047	-7e-05	-9.9e-06	-1.3e-06	-1.4e-07	-9.3e-09	1.3e-09	7.8e-10	2
alpha_5	14	0.55	-0.059	-0.0085	-0.0014	-0.00024	-4.1e-05	-6.9e-06	-1.1e-06	-1.9e-07	-3.1e-08	-5.1e-09	-8.2e-10	-
alpha_10	18	0.4	-0.037	-0.0055	-0.00095	-0.00017	-3e-05	-5.2e-06	-9.2e-07	-1.6e-07	-2.9e-08	-5.1e-09	-9.1e-10	-
alpha_20	23	0.28	-0.022	-0.0034	-0.0006	-0.00011	-2e-05	-3.6e-06	-6.6e-07	-1.2e-07	-2.2e-08	-4e-09	-7.5e-10	-

1. The RSS increases with λ, while model complexity reduces
2. Even a **small** λ (e^{-15}) results in significant reduction coefficients magnitude. Compare the coefficients first row to the last row of simple linear regression table.
3. High λ values lead to significant under-fitting
   - Note rapid increase in RSS for $\lambda > 1$
   - The coefficients are **very very small**, they are **NOT zero**

https://www.analyticsvidhya.com/blog/2016/01/

Lasso regression effect on regression weights



Model complexity decreases with increase in the values of λ.

- Notice straight line at $\lambda = 1$

https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/

Regularization and sparsity

- For the same values of λ (α in table) , lasso coefficients $\ll$ ridge coefficients.
- For the same λ, lasso has higher RSS (poorer fit) as compared to ridge regression.
- Many of the coefficients are zero even for very small values of alpha.
- Inferences $\#1, 2$ might not generalize always but will hold for many cases. The real difference from ridge is coming out in the last inference.

	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11	coe
alpha_1e-15	0.96	0.22	1.1	-0.37	0.00089	0.0016	-0.00012	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.4
alpha_1e-10	0.96	0.22	1.1	-0.37	0.00088	0.0016	-0.00012	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.4
alpha_1e-08	0.96	0.22	1.1	-0.37	0.00077	0.0016	-0.00011	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.3
alpha_1e-05	0.96	0.5	0.6	-0.13	-0.038	-0	0	0	0	7.7e-06	1e-06	7.7e-08	0	0
alpha_0.0001	1	0.9	0.17	-0	-0.048	-0	-0	0	0	9.5e-06	5.1e-07	0	0	0
alpha_0.001	1.7	1.3	-0	-0.13	-0	-0	-0	0	0	0	0	0	1.5e-08	7.5
alpha_0.01	3.6	1.8	-0.55	-0.00056	-0	-0	-0	-0	-0	-0	-0	0	0	0
alpha_1	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
alpha_5	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
alpha_10	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0

HIGH SPARSITY

https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/

Elastic Net

– $p > n$ (#variables greater than sample size), lasso can select only n variables

– select only one from any set of highly correlated variables

– even when $n > p$, if the variables are strongly correlated, ridge regression tends to perform better

- Combines L1 and L2 regularization to balance out the pros and cons of ridge and lasso regression

$$\arg\min_{\boldsymbol{w}\in\mathbb{R}^p}\{|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}|^2 + \lambda_1|\boldsymbol{w}| + \lambda_2|\boldsymbol{w}|^2\}$$

- Subject to: $(1-\alpha)|\boldsymbol{w}| + \alpha|\boldsymbol{w}|^2 \leq t$ with $\alpha = \frac{\lambda_2}{\lambda_1+\lambda_2}$

Other regularization approaches

Group Lasso

– *Regularization based on groups of variables*
– *Lasso selects one of them – we might need all variables of a group (i. e. topic modelling, biological applications)*

Objective function:

$$\arg\min_{\boldsymbol{w}\in\mathbb{R}^p}\{|y-\sum_{j=1}^{J}\mathbf{X}_j w_j|^2+\lambda\sum_{j=1}^{J}|w_j|_{\mathbf{K}_j}\}$$

where J: the variable groups and $|w_j|_{\mathbf{K}_j}=(w^T\mathbf{K}_j w)^{\frac{1}{2}}$.

Properties of Group Lasso:

1. entire groups are eliminated
2. all variables of a group are present with the same weight
3. if groups contain 1 variable $\Rightarrow$ lasso

To weight variables within groups: Sparse Group Lasso

Regularization – Conclusion

- *Ridge:* all of the features retained in the model. Major advantage: coefficient shrinkage thus reducing model complexity.
- *Lasso:* Along with shrinking coefficients, lasso performs feature selection as some of the coefficients become exactly zero.

Typical Use Cases

- *Ridge: prevent overfitting* – includes all features, useful in case of very high number of features.
- *Lasso:* provides *sparse solutions* – suitable for cases where # features is large: great computational advantage as features with zero coefficients can be ignored (feature selection).

Presence of Highly Correlated Features

- *Ridge:* works well for highly correlated features – includes all of them in the model, coefficients will adjust.
- *Lasso: selects* feature among the highly correlated ones, reduces the coefficients of the rest to zero.
  For highly correlated variables even small λ values give significant sparsity.

References I

Christopher M. Bishop
Pattern Recognition and Machine Learning
2006

Trevor Hastie, Robert Tibshirani, Jerome Friedman
The Elements of Statistical Learning: Data Mining, Inference, and Prediction
Second Edition. 2009

Shai Shalev-Shwartz, Shai Ben-David
Understanding Machine Learning: theory and algorithms
Cambridge University Press. 2014

Robert Tibshirani
Regression shrinkage and selection via the Lasso
Journal of the Royal Statistics Society. 58(1), 267-288. 1996
http://statweb.stanford.edu/~tibs/lasso/lasso.pdf

http://people.inf.elte.hu/kiss/13dwhdm/roc.pdf

References II

Hui Zou, Trevor Hastie
Regularization and Variable Selection via the Elastic Net
Journal of the Royal Statistical Society. Series B (Statistical Methodology). Wiley.67(2): 301–20. 2005

Jerome Friedman, Trevor Hastie, Robert Tibshirani
A note on the group lasso and a sparse group lasso
https://arxiv.org/pdf/1001.0736.pdf

Acknowledgments

Dr. Nikos Tziortziotis
for his contributions in the Kernels section