Regularization and Evaluation

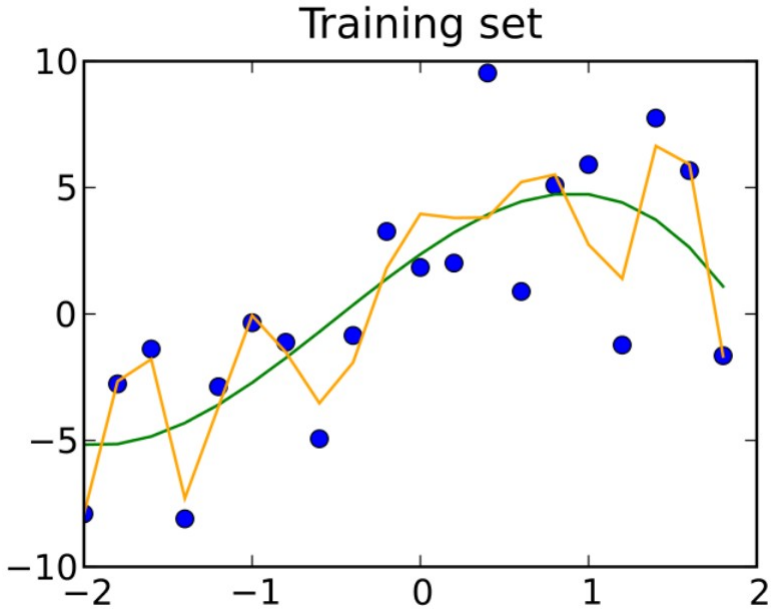
M. Vazirgiannis

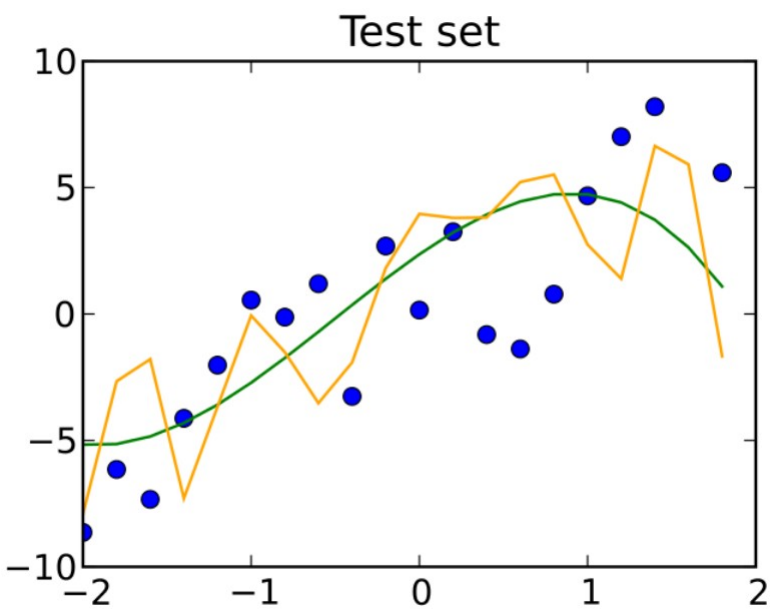
November 2018

Outline

- Regularization
- Machine learning evaluation

Training vs. test error

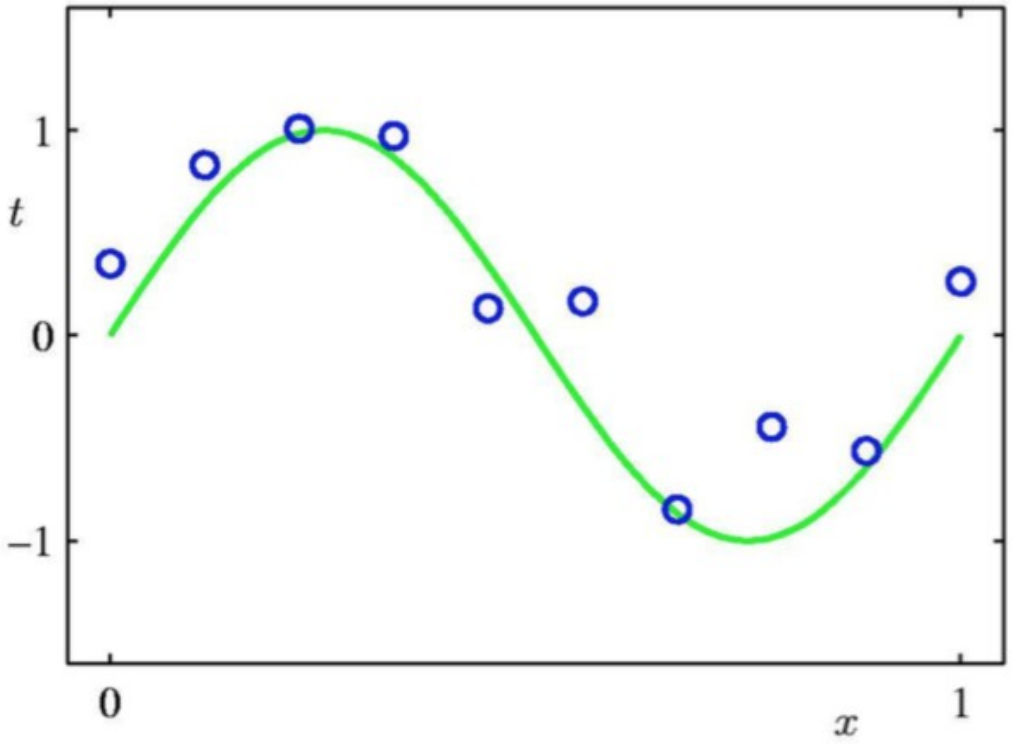




- Model complexity causes overfitting
- Orange model: $MSE_{test}/MSE_{train} = 4$
- Green model: $MSE_{test}/MSE_{train} = 2$

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

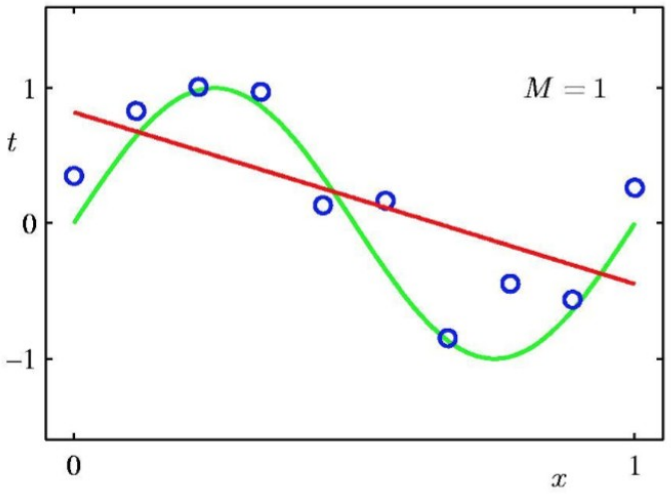
Polynomial Curve fitting

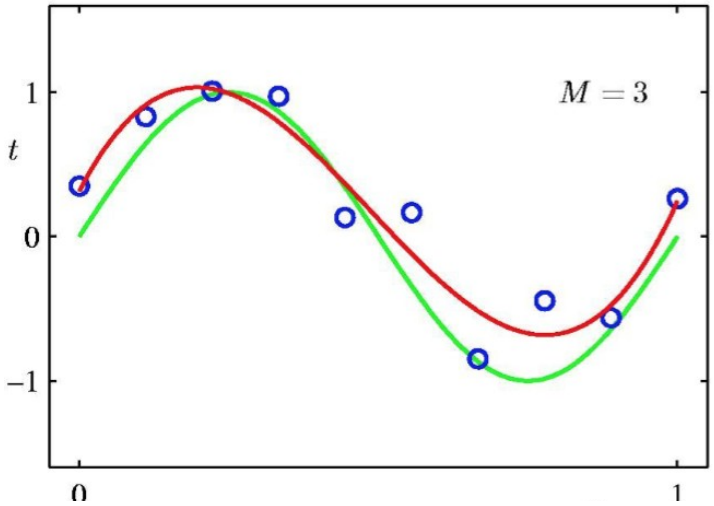


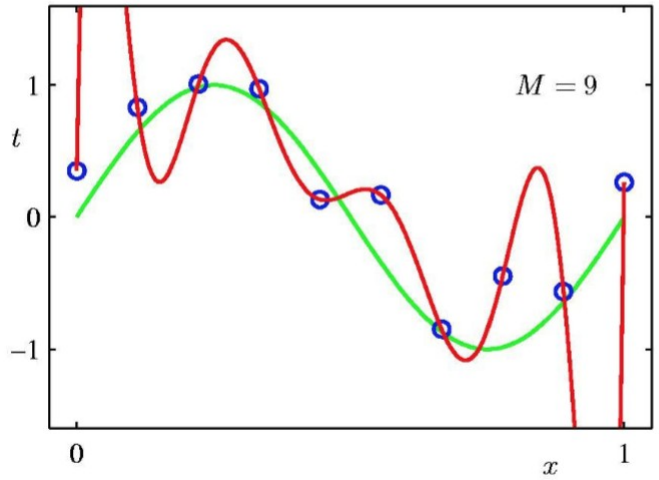
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

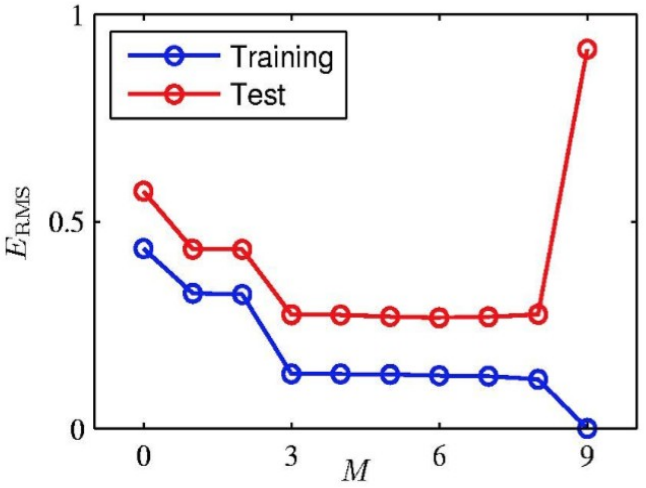
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

Model complexity vs overfitting







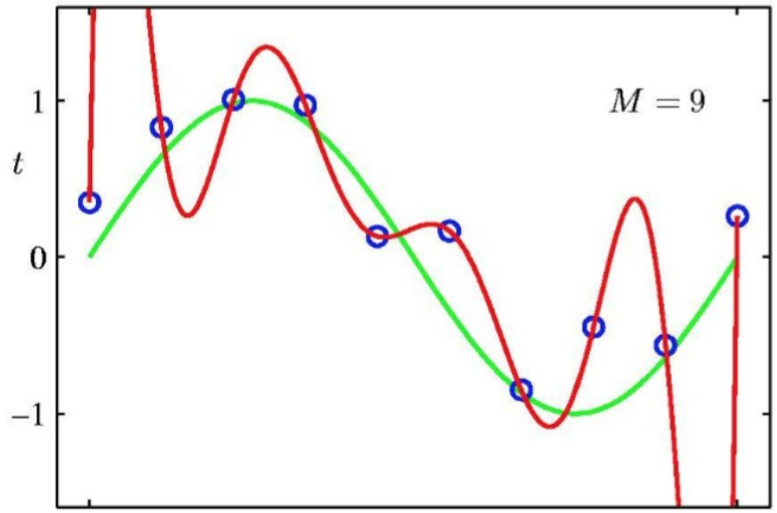


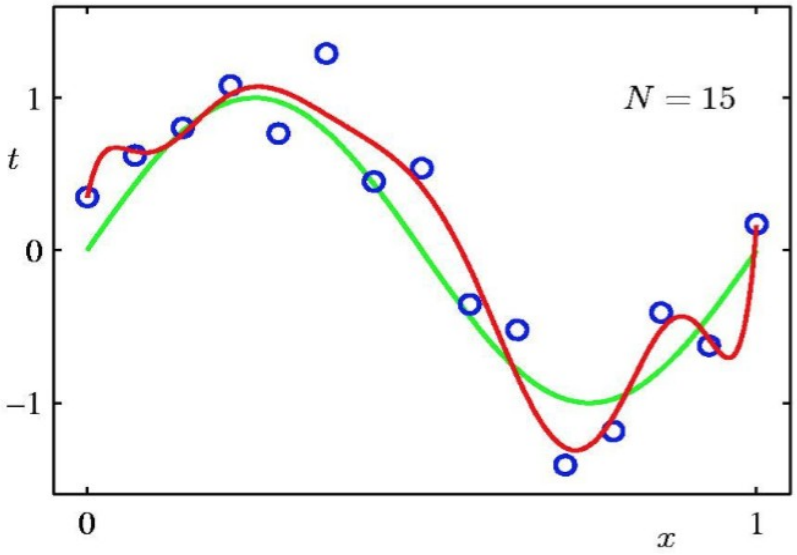
Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

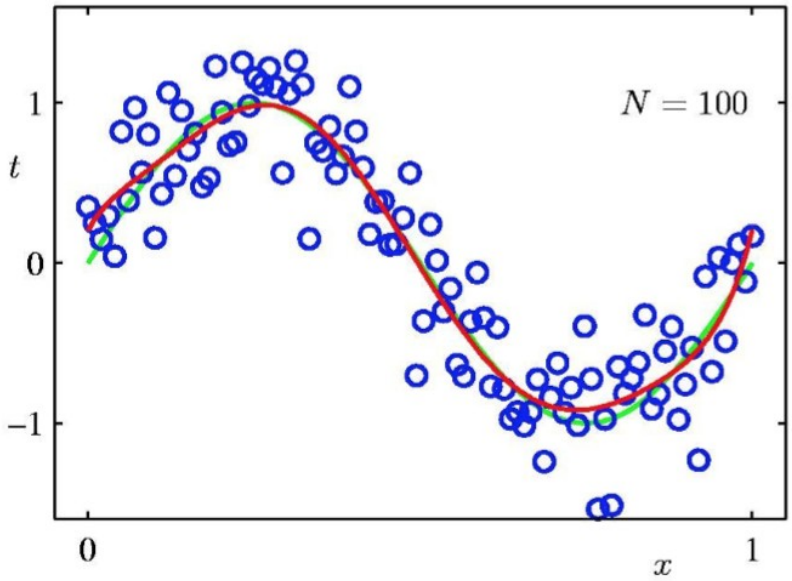
Polynomial coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

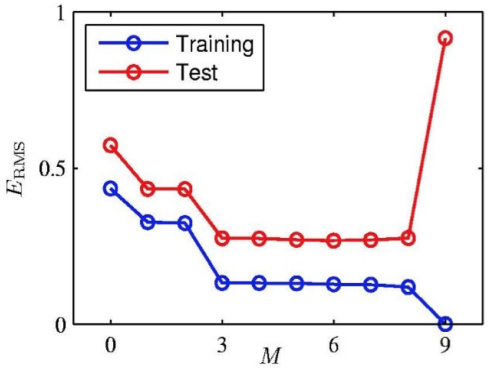
Data Set size







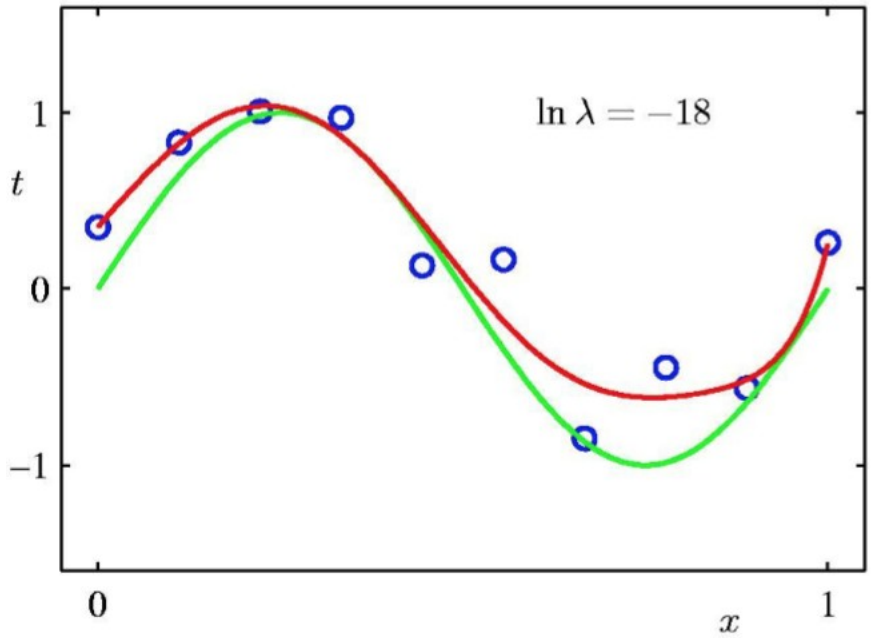
Regularization

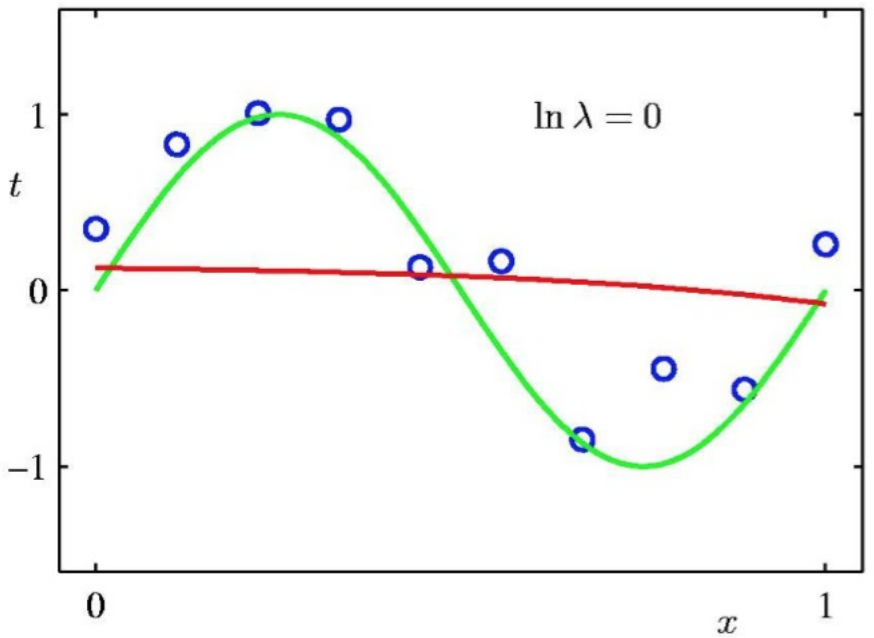


- Complex models tend to over fit
- Need to penalize the complexity of the model

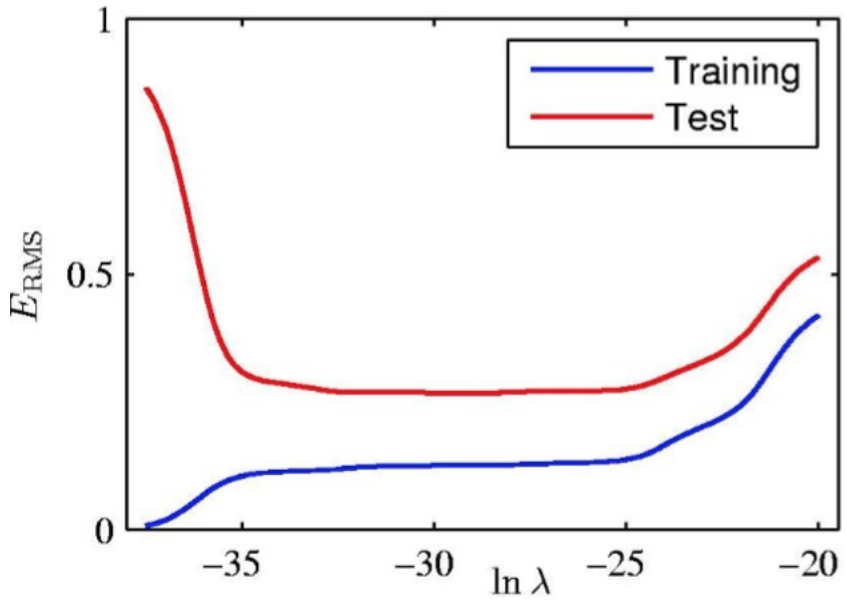
Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$





Regularization: Error vs λ



	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

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

Generalization

- 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(x,y) unknown probability distribution => 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 $L_s(w)$ and penalizing the model complexity via the regularization function $R(w)$

$$\operatorname*{argmin}_{\mathbf{w}} \left(L_S(\mathbf{w}) + R(\mathbf{w}) \right).$$

- The simplest one is $R(w) = \lambda ||w||^2$, λ scalar value and $||w||^2 = \sqrt{\sum w_i^2}$
- There fore: $A(S) = \operatorname*{argmin}_{\mathbf{w}} \left(L_S(\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \right).$

Squared Loss - Ridge regression

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

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

- Solving for gradient = 0 we get:

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

- λ ≥ 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:

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

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

Regularization of the intercept

- regression $\boldsymbol{Y} = \boldsymbol{wX}$ => $\boldsymbol{Y} = w_0 + w_1 x_1 \ldots + w_m x_m$
- => for $\boldsymbol{X=0}$: $|Y| = w_0$
- Regularization based on the idea that overfitting on $\boldsymbol{Y}$ is caused by a being "overly specific",
  - usually resulting in large values of **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.
- $\boldsymbol{Y} = \boldsymbol{w_{1..m}X} + w_0$,
  - $\boldsymbol{w_{1..m}}$ is multiplied with the explaining variables, $\boldsymbol{w_0}$ added to it.
- Regularizing $\boldsymbol{w_0}$ may increase bias…

Ridge regularization computation

- $X_{Nx(M+1)}$: matrix of input features (N rows, $M+1$ columns, m weights and the intercept)
- Y_N: the actual outcome variable
- $\hat{Y}_N$: predicted values of Y
- 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):

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

Ridge regularization computation

Regression algorithm – with gradient descent

```
w=0; η: learning rate
while no convergence do
   for each feature j in {0,1...M}
      determine the gradient
      w(t+1) = w(t) – η*gradient
```

gradient for the j[th] weight: $\frac{\partial}{\partial w_j} Cost\ (W) = -2 \sum_{i=1}^{N} x_{ij} \left\{ y_i - \sum_{k=0}^{M} w_k\ x_{ik} \right\}$

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: $$\frac{\partial}{\partial w_j} Cost\,(W) = -2\sum_{i=1}^{N} x_{ij}\left\{y_i - \sum_{k=0}^{M} w_k\, x_{ik}\right\} + 2\lambda w_j$$
- The update rule

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

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

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 $(1\text{-}2\lambda\eta)$
  - then applies the same update rule as simple linear regression.
- explains why the coefficients reduce to small numbers but never zero.

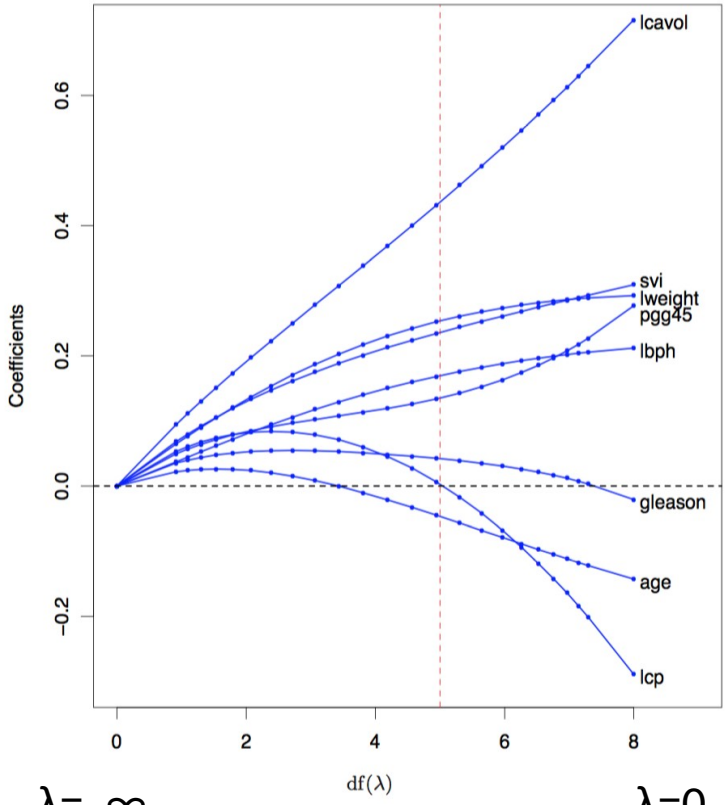
Squared Loss - Ridge regression

$$A(S) = \operatorname*{argmin}_{\mathbf{w}} \left(L_S(\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \right).$$

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

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
- λ->0: obtain the un-regularized solution
- λ-> ∞: w=0:
  - infinite weights on square of coefficients,
  - anything less than zero will make the error infinite.



λ= ∞ λ=0

Lasso regularization

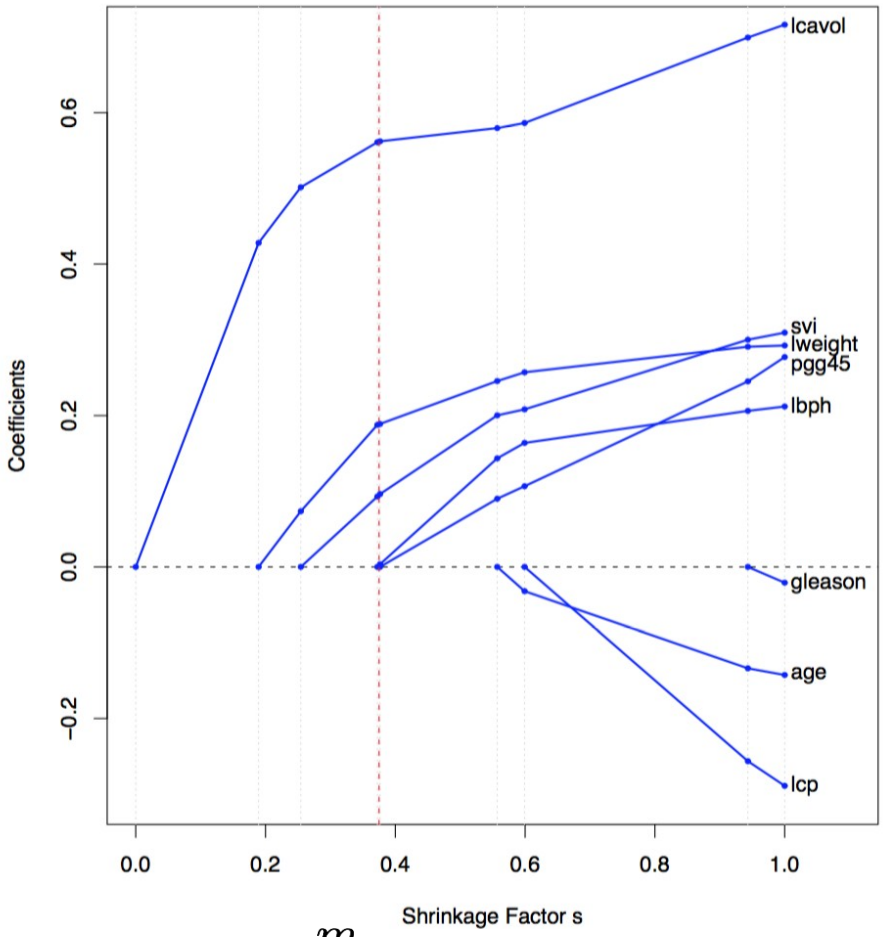
- The Lasso regression (Tibshirani 1996) is penalizing the sum of absolute values of the weights

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

- Subject to the constraint $\sum_{i=1}^{m} |w_i| \leq t$
- the solution for w_0 is $mean\ value\ of\ y$ thus we fit a model without the intercept w_0
- The constraint makes the solutions nonlinear in the y_i : no closed form expression as in ridge regression.
- Computing the lasso solution is a quadratic programming problem

Lasso regression

- choose $t_0 = \sum_{i=1}^{m} |w_i|$, where w_i the weights produced by the least square solution (i.e. non regularized results)
- for smaller values of t $(t>=0)$ 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



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

Computation of the Lasso solutions

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

Incremental Forward Stepwise Regression: FS

1. Start with $\boldsymbol{r} = \boldsymbol{y} - \bar{\boldsymbol{y}}$ and all $\boldsymbol{w}_j = 0$
2. Find predictor $\boldsymbol{x}_j$ most correlated with $\boldsymbol{r}$.
3. Update $\boldsymbol{w}_j \leftarrow \boldsymbol{w}_j + \delta_j$, where $\delta_j = \varepsilon * sign[corr(r, \boldsymbol{x}_j)]$;
4. Update $\mathbf{r} \leftarrow \mathbf{r} - \delta_j \boldsymbol{x}_j$, and repeat steps 2,3 until no predictor has any correlation with r.

_____

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

Computation of the Lasso solutions

- least angle regression procedure is a better approach
  - exploits the special structure of the lasso problem,
  - efficient way to compute the solutions simultaneously for all values of "w_i".
  - Least angle is based on the forward stepwise regression

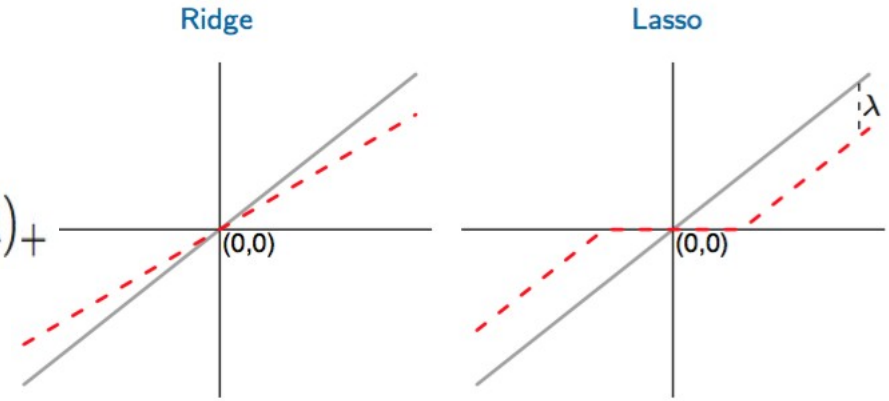
Least angle regression algorithm:

- Set all coefficients $w_j = 0$
- Find the predictor x_j most correlated with y
- Increase the coefficient w_j in the direction of the sign of its correlation with $\mathbf{y}$.
- Take residuals $r = y - \bar{y}$ along the way.
- Stop when some other predictor x_k has as much correlation with $\mathbf{r}$ as x_j has.
- Increase $(w_j w_k)$ in their joint least squares direction,
- until some other predictor x_m has as much correlation with the residual r.
- Continue until: all predictors are in the model

Comparing ridge and lasso

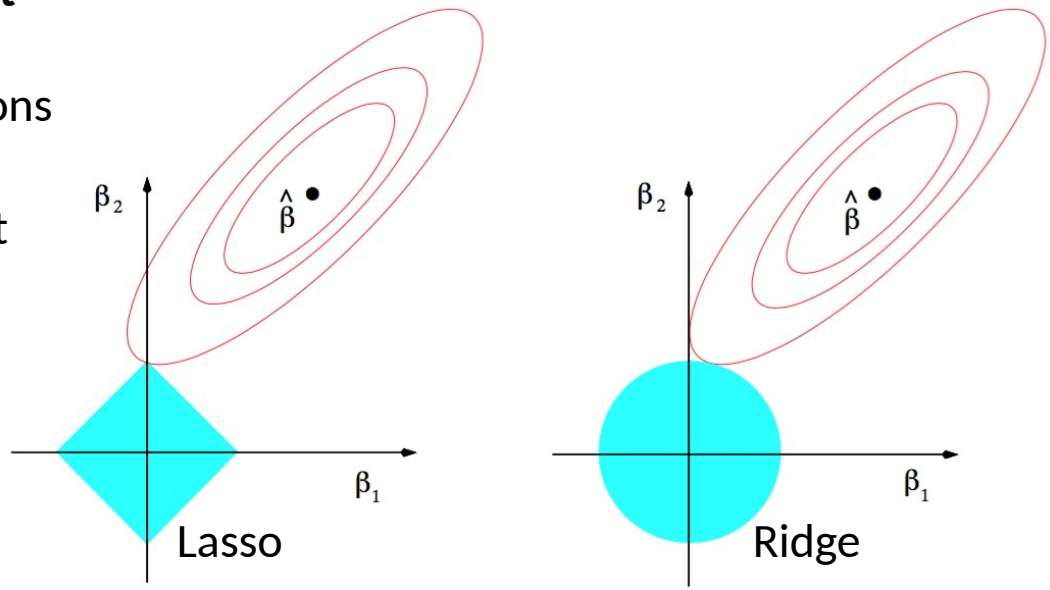
Ridge $\hat{\beta}_j/(1+\lambda)$

Lasso $\text{sign}(\hat{\beta}_j)(|\hat{\beta}_j| - \lambda)_+$



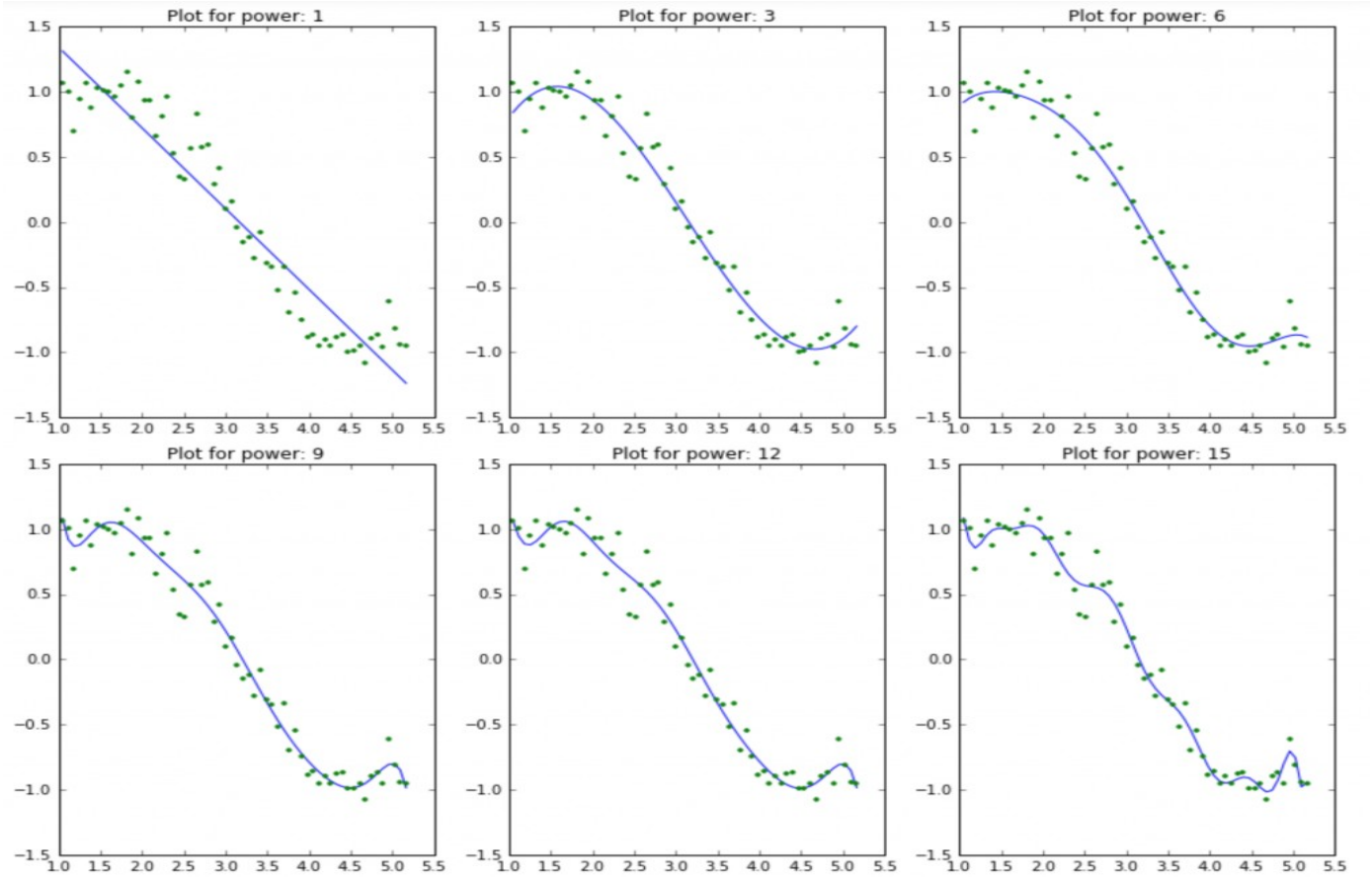
contours of the error and constraint functions.

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



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

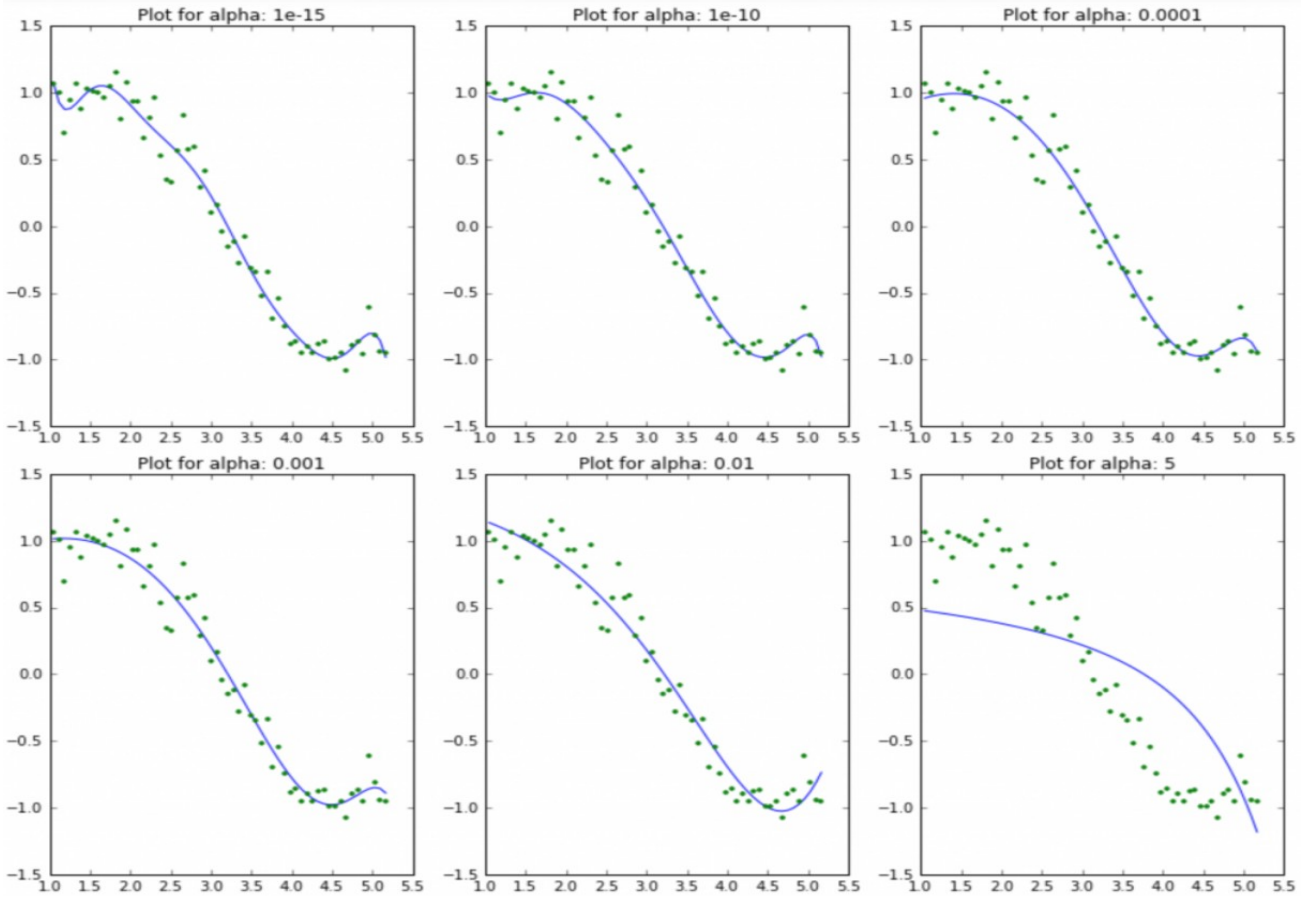
- Regression: coefficient values vs model complexity (polynomial degree)

	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	-1
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	-1

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: regression weights vs λ 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.

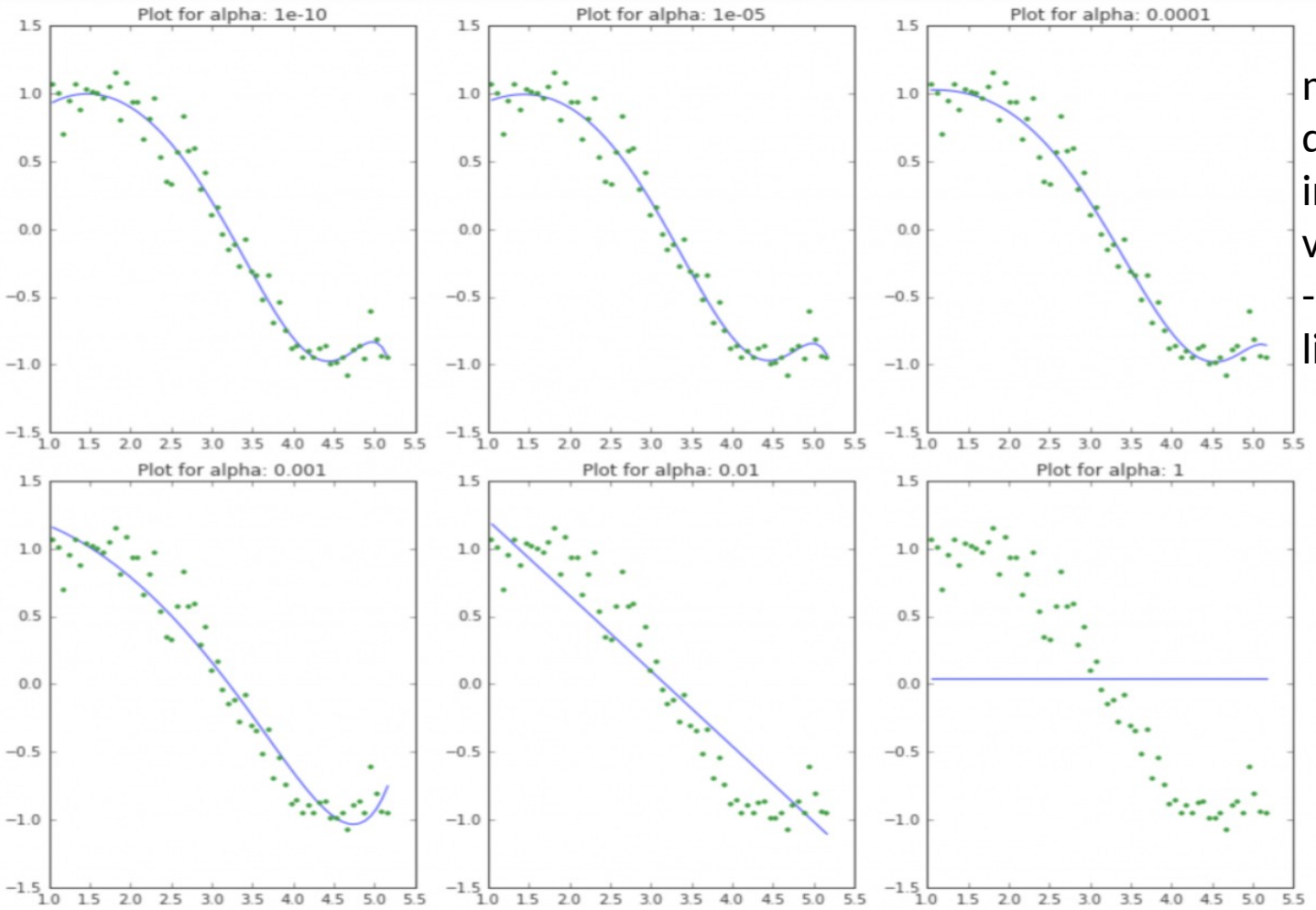
3.High λ values lead to significant under-fitting.

Note rapid increase in RSS for λ > 1

the coefficients are **very very small**, they are **NOT zero**.

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

Lasso regression effect on regression weights



model complexity decreases with increase in the values of λ.
- Notice straight line at λ=1.

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

Regularization and sparsity

- For the same values of λ, lasso coefficients << ridge coefficients.
- For the same λ, lasso has higher RSS (poorer fit) as compared to ridge regression

	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/

Other regularization approaches

Elastic Net

- *p* > *n* (# variables greater sample size) lasso can select only *n* covariates
- selecta only one covariate from any set of highly correlated covariates.
- even when *n* > *p*, if the covariates 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.

$$argmin_{\mathbf{w}\in R^p}\{|y - Xw|^2 + \lambda_1|\mathbf{w}| + \lambda_2|\mathbf{w}|^2\}$$

- Subject to: $(1-\alpha)|\mathbf{w}| + \alpha|\mathbf{w}|^2 \leq t$ 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 broup (i.e. topic modelling, biological applications)*

- Objective function:

$$argmin_{\mathbf{w}\in R^p}\{|y-\sum_{j=1}^{J}X_j w_j|^2+\lambda\sum_{j=1}^{J}|w_j|_{K_j}\}$$

where J: the variable groups and

$$|w|_{K_j}=(w^T K_j w)^{1/2}$$

Properties of Group Lasso:

i. entire groups are eliminated
ii. all variables of a group are present and same weight
iii. if groups contain 1 variable => lasso

- To weight variables within groups: Sparse Group Lasso

Regularization - Conclusion

- Ridge: all (or none) of the features in the model. Thus, the major advantage of ridge regression is coefficient shrinkage and reducing model complexity.
- Lasso: Along with shrinking coefficients, lasso performs feature selection - some of the coefficients become exactly zero

Typical Use Cases

- Ridge: *prevent overfitting* - includes all the features, useful in case of very high number of features
- Lasso: Provides *sparse solutions* – suitable for cases where #features are in millions or more = great computational advantage as features with zero coefficients can be ignored.

Presence of Highly Correlated Features

Ridge: works well for highly correlated features - will include all of them in the model , coefficients will adjust

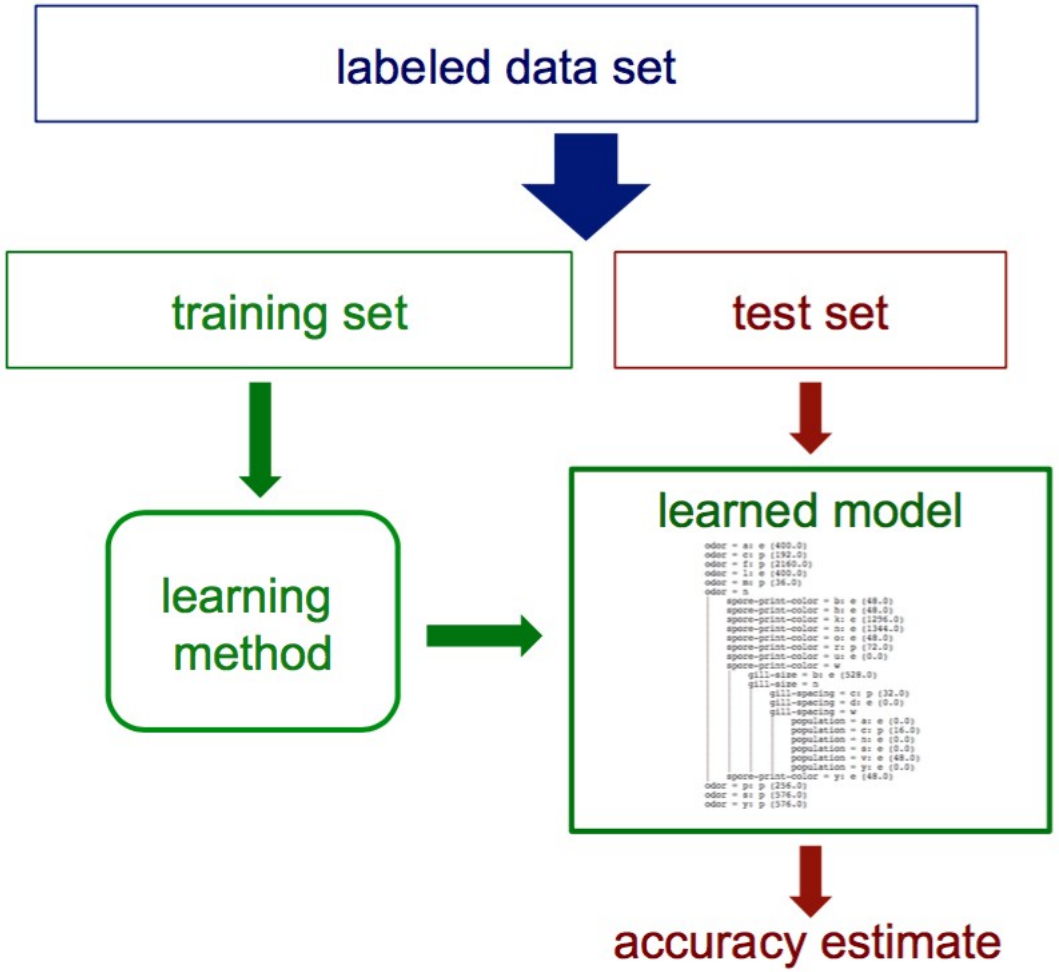
Lasso: selects any feature among the highly correlated ones, reduces the coefficients of the rest to zero.

For highly correlated variables even small λ values give significant sparsity

Outline

- Regularization
- **Machine learning evaluation**

Test sets



Find an unbiased estimate of the accuracy of a learned model?

http://pages.cs.wisc.edu/~dpage/cs760/evaluating.pdf

Learning curves

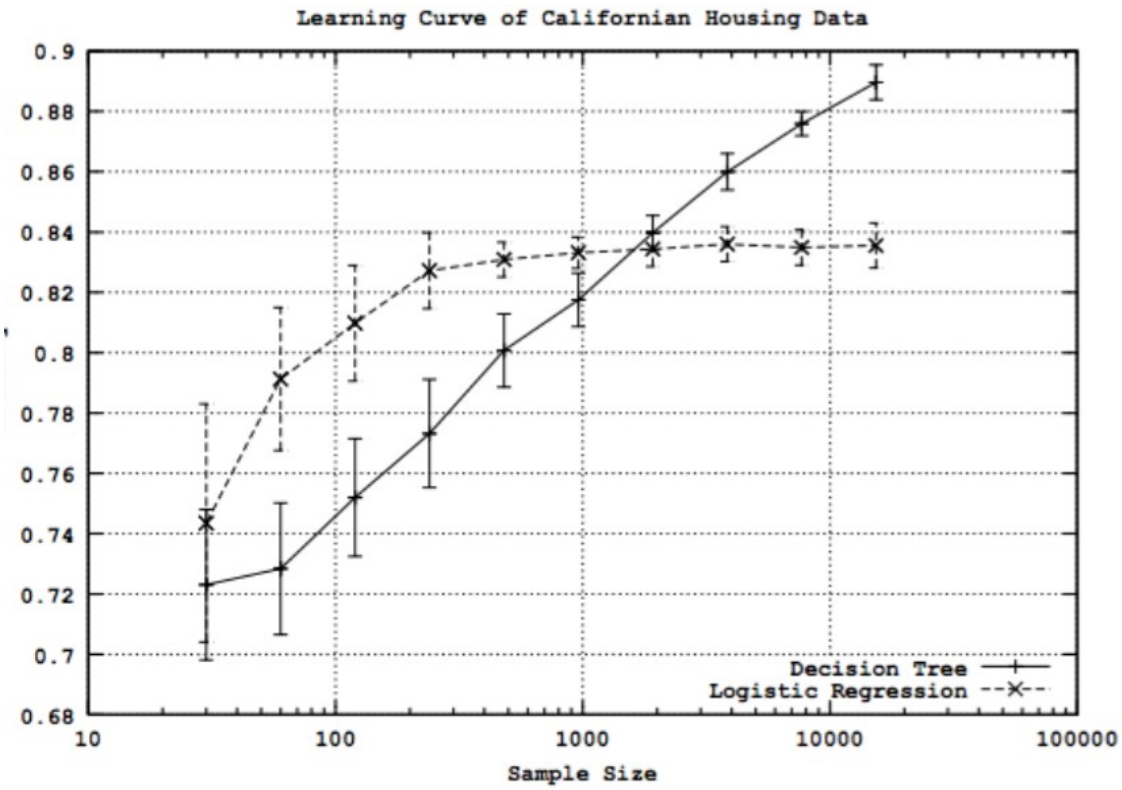


Figure from Perlich et al. *Journal of Machine Learning Research*, 2003

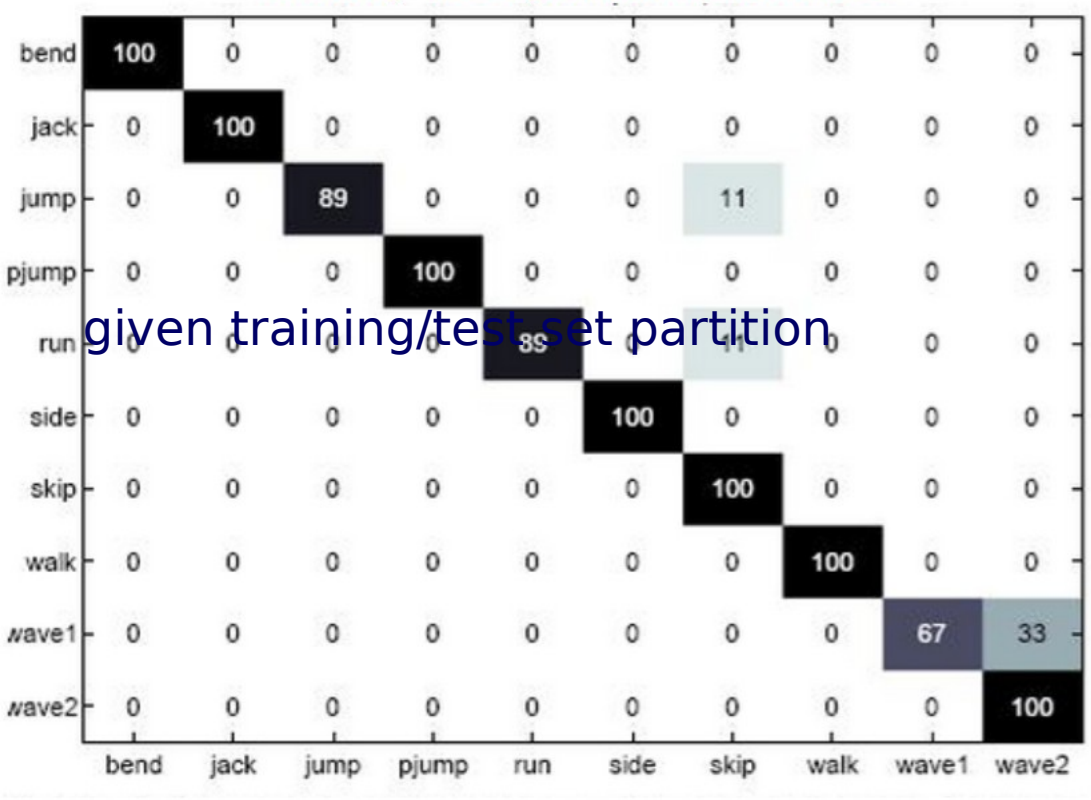
- Accuracy of prediction vs. training set size.

  given training/test set partition

  for each sample size s on learning curve (optionally repeat *n* times)
  - randomly select *s* instances from training set
  - learn model
  - evaluate model on test set to determine accuracy *a*
  - plot (*s*, avg. accuracy and error bars)

Confusion Matrix

activity recognition from video



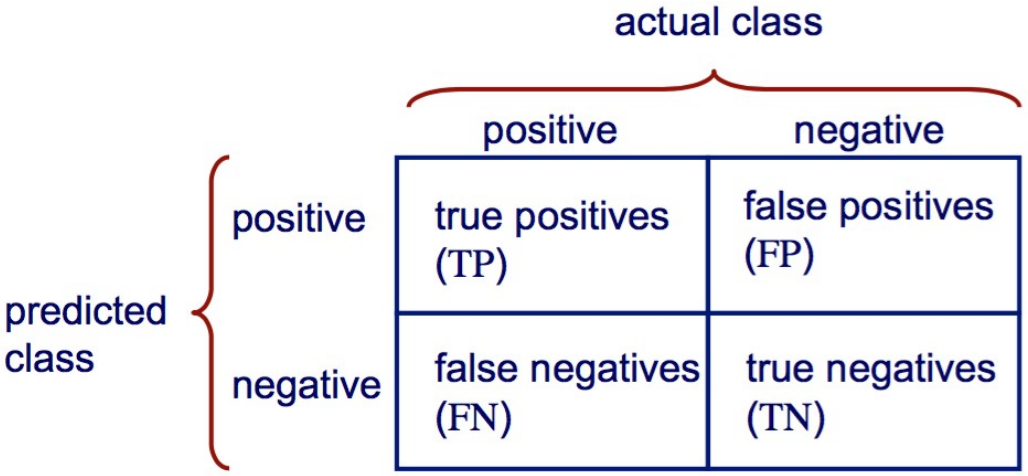
actual class

given training/test set partition

predicted class

figure from vision.jhu.edu

Confusion matrix for 2-class problems



$$\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

$$\text{true positive rate (recall)} = \frac{TP}{\text{actual pos}} = \frac{TP}{TP + FN}$$

$$\text{false positive rate} = \frac{FP}{\text{actual neg}} = \frac{FP}{TN + FP}$$

- Accuracy may not be useful measure in cases where
- there is a large class skew
  - Is 94% accuracy good if 93% of the instances are negative?
- cost sensitive classification, getting a positive wrong costs more than getting a negative wrong
  - i.e in a a medical domain false negative results in failure to treat a disease

ROC curve

- *Receiver Operating Characteristic* (*ROC*) or **ROC curve**, is a graphical plot illustrating the performance of a binary classifier system as its discrimination threshold (probability) is varied
- Plot true positive rate vs false positive rate for each possible classification threshold
  - Assume classifiers that produce for each data point a probability for positive/negative classification
- ROC visualizes performance for all classification thresholds

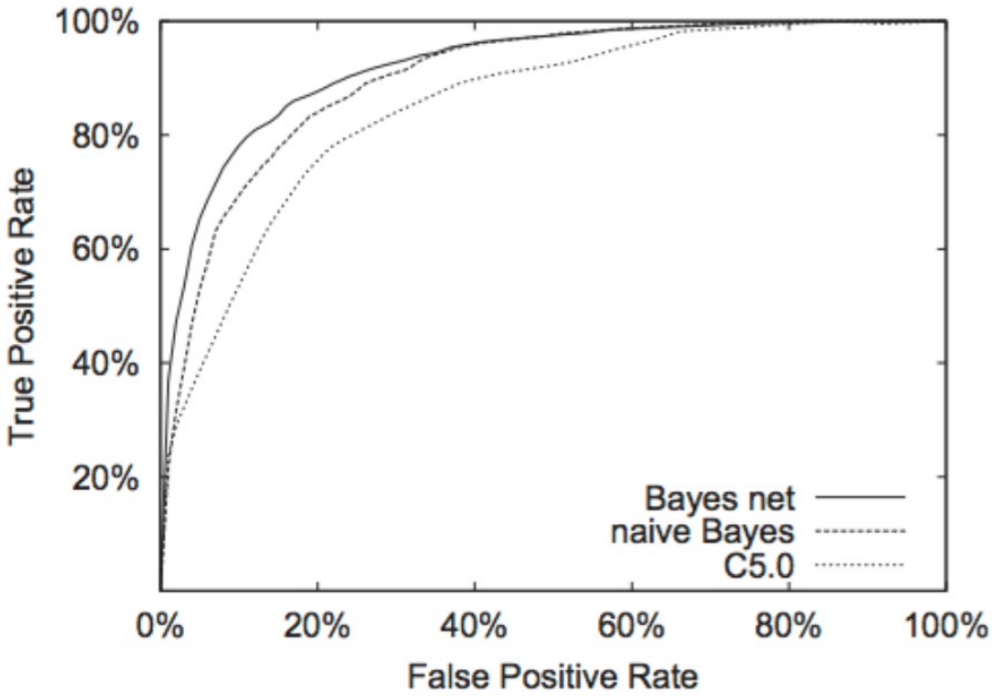
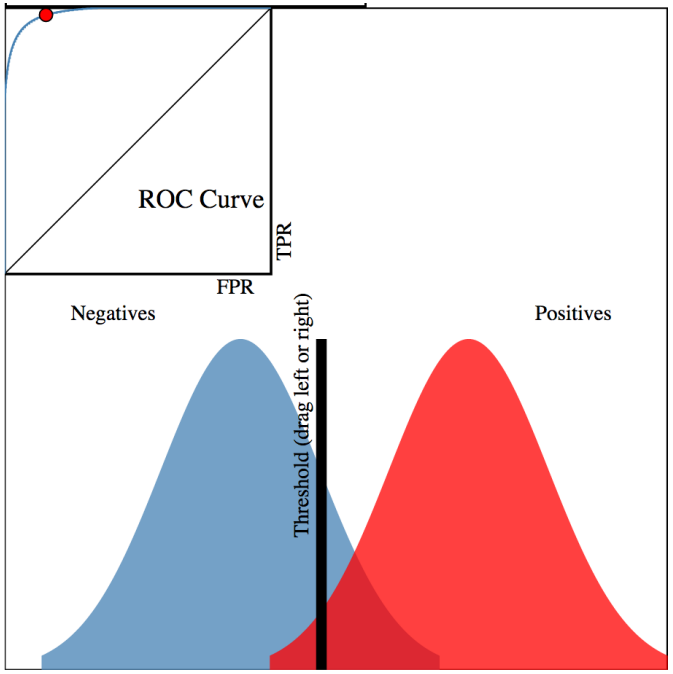


figure from Bockhorst et al., *Bioinformatics* 2003

- An excellent visualization: http://www.navan.name/roc/

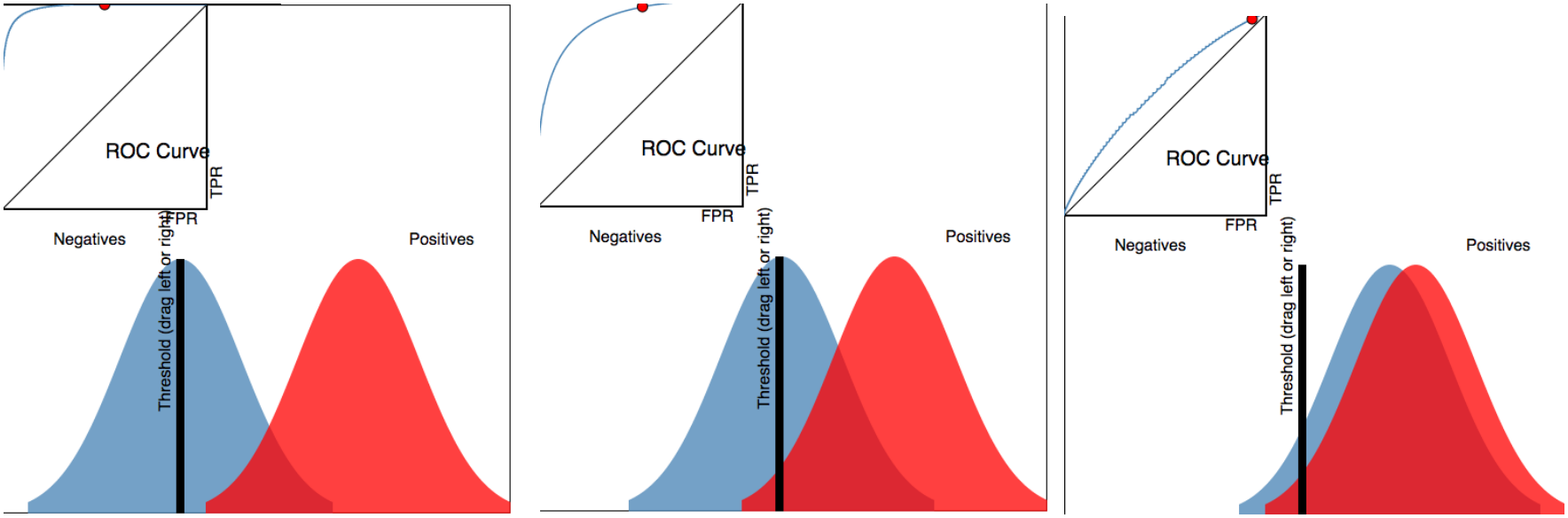
ROC curves

- For each threshold i in [0,1]
  - find $FPRi, TPRi$
  - plot $(FPRi, TPRi)$



An excellent visualization: http://www.navan.name/roc/

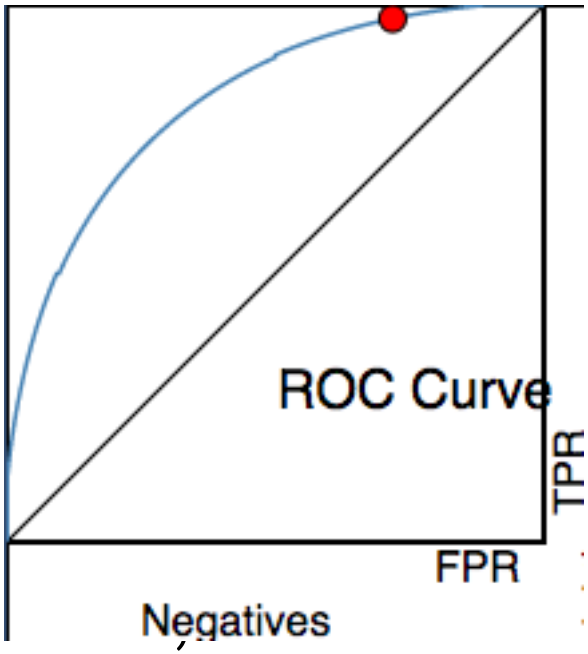
ROC curves

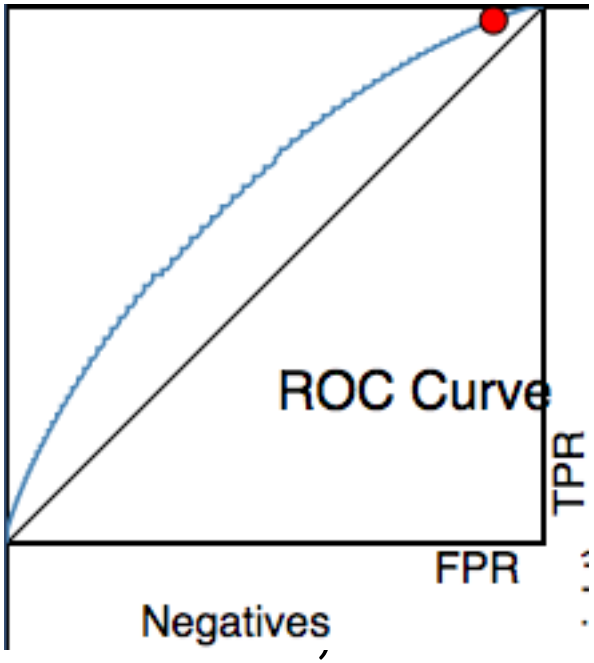


- Class overlap towards random guessing

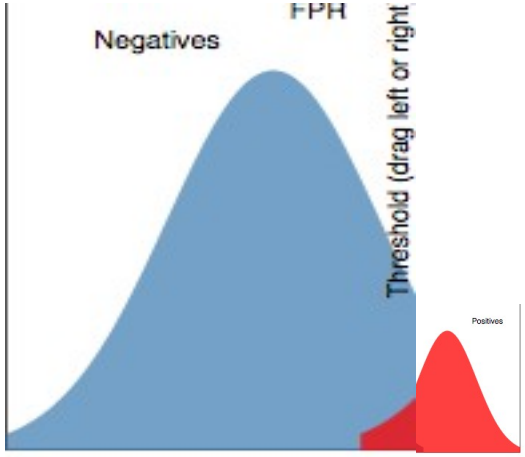
AUC curve

- Quantify the performance of the classifier



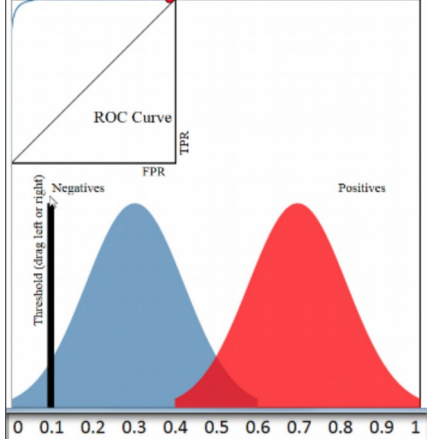


ROC curves issues



- not-balanced classes
- Non normal distributions
- ROC curves robust to non proper probabilities – only the ranking order counts
  - Good solution for highly unbalanced classes
- extended to **classification problems with three or more classes**
  - Class1 vs classes 2&3
  - Class2 vs classes 1&3
  - Class3 vs classes 1&2

ROC curve – setting the threshold

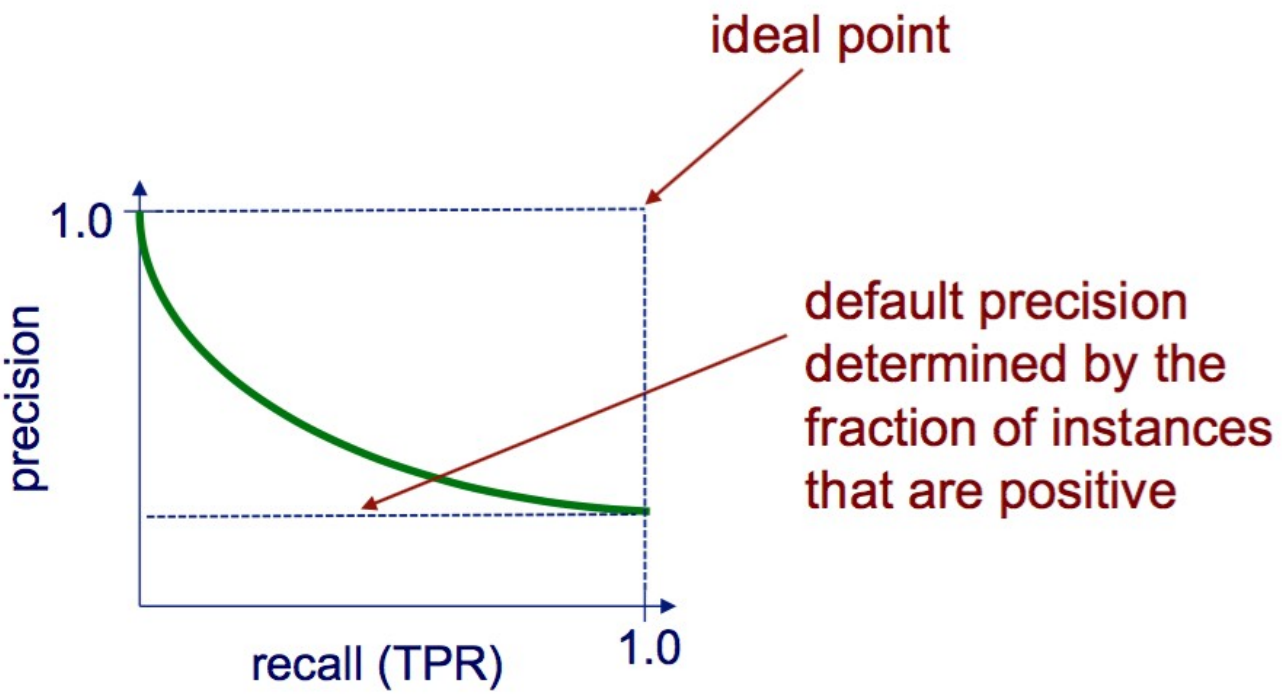


- how to set your classification threshold, to predict out-of-sample data.
- more of a **business decision**,
  - **minimize your False Positive Rate or**
  - **maximize your True Positive Rate**.
- i.e. classifier to predict credit card transaction might be fraudulent and thus should be reviewed by the credit card holder.
  - business decision set the threshold very low.
  - lot of false positives, but it would maximize the true positive rate
  - thus minimize the number of cases in which a real instance of fraud was not flagged for review.

Precision – recall curves

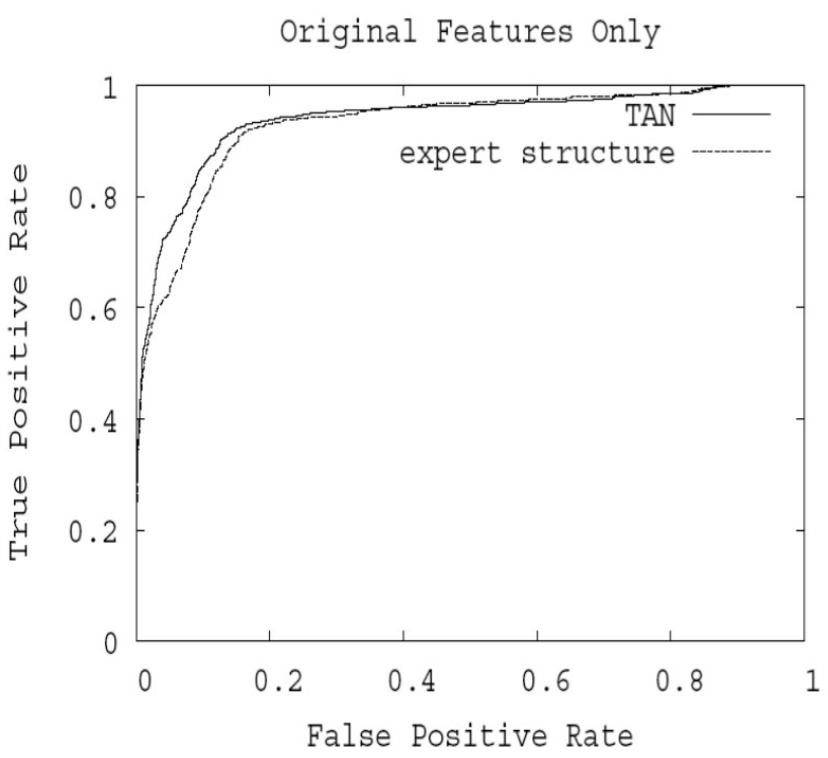
$$\text{recall (TP rate)} = \frac{TP}{\text{actual pos}} = \frac{TP}{TP + FN}$$

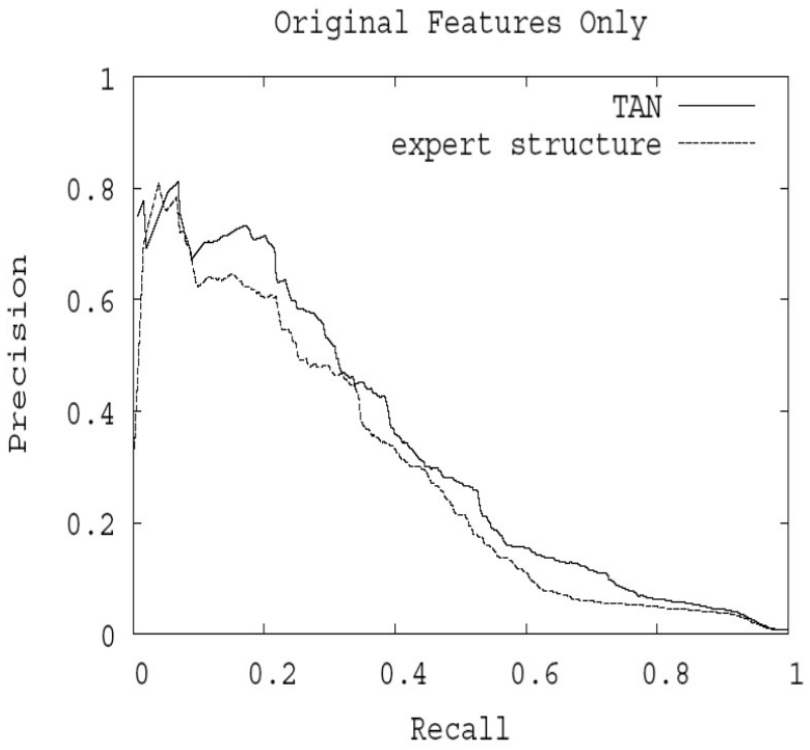
$$\text{precision} = \frac{TP}{\text{predicted pos}} = \frac{TP}{TP + FP}$$



PR vs ROC

ROC PR





- Is it possible to map points between the two plots?
- Are the two AUCs equivalent?

References

- Pattern Recognition and Machine Learning. Christopher M. Bishop. 2006
- The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Ed. 2009, Trevor Hastie, Robert Tibshirani, Jerome Friedman
- Shai Shalev-Shwartz and Shai Ben-David, Understanding Machine Learning: theory and algorithms, Cambridge University Press, 2014
- Regression shrinkage and selection via the Lasso, Tibshirani, 1996, journal of the Roual Statistics Society, 58(1), 267-288. http://statweb.stanford.edu/~tibs/lasso/lasso.pdf
- http://people.inf.elte.hu/kiss/13dwhdm/roc.pdf
- Zou, Hui; Hastie, Trevor (2005). "Regularization and Variable Selection via the Elastic Net".*Journal of the Royal Statistical Society*. Series B (statistical Methodology). Wiley. **67** (2): 301–20.
- A note on the group lasso and a sparse group lasso Jerome Friedman ∗ Trevor Hastie † and Robert Tibshirani. 2010, https://arxiv.org/pdf/1001.0736.pdf