Introduction to Machine Learning
NPFL 054

http://ufal.mff.cuni.cz/course/npfl054

Barbora Hladká
hladka@ufal.mff.cuni.cz

Martin Holub
holub@ufal.mff.cuni.cz

Charles University,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

Lecture #10

Outline

- Model complexity, overfitting, bias and variance
- Regularization – Ridge regression, Lasso
  - Linear regression
  - Logistic regression
- Instance-based learning

Model complexity

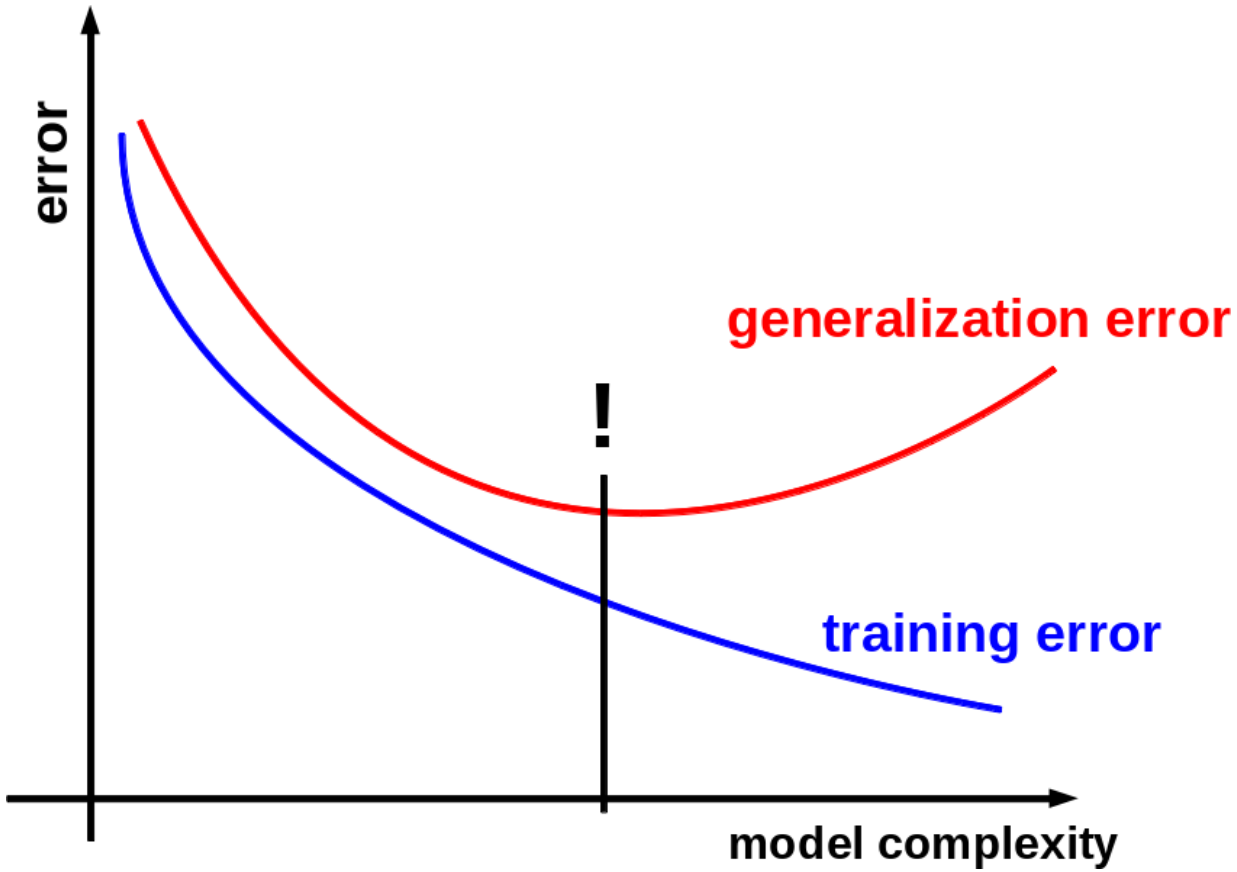
No universal definition

Heading for the regularization ...
model complexity is the number of hypothesis parameters

$$\Theta = \langle \theta_0, \ldots, \theta_m \rangle$$

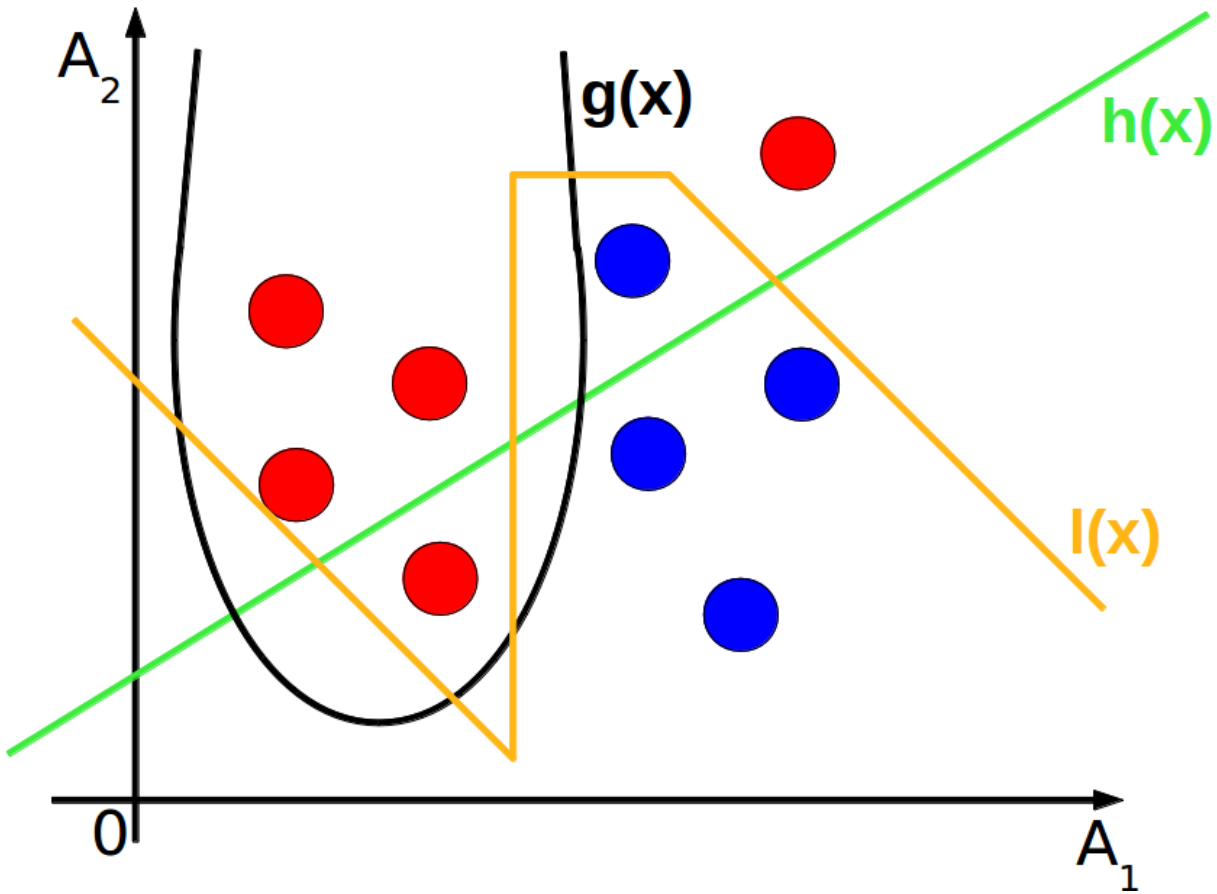
Model complexity

Finding a model that minimizes generalization error
. . . is one of central goals of the machine learning process



Model complexity

Complexity of decision boundary for classification



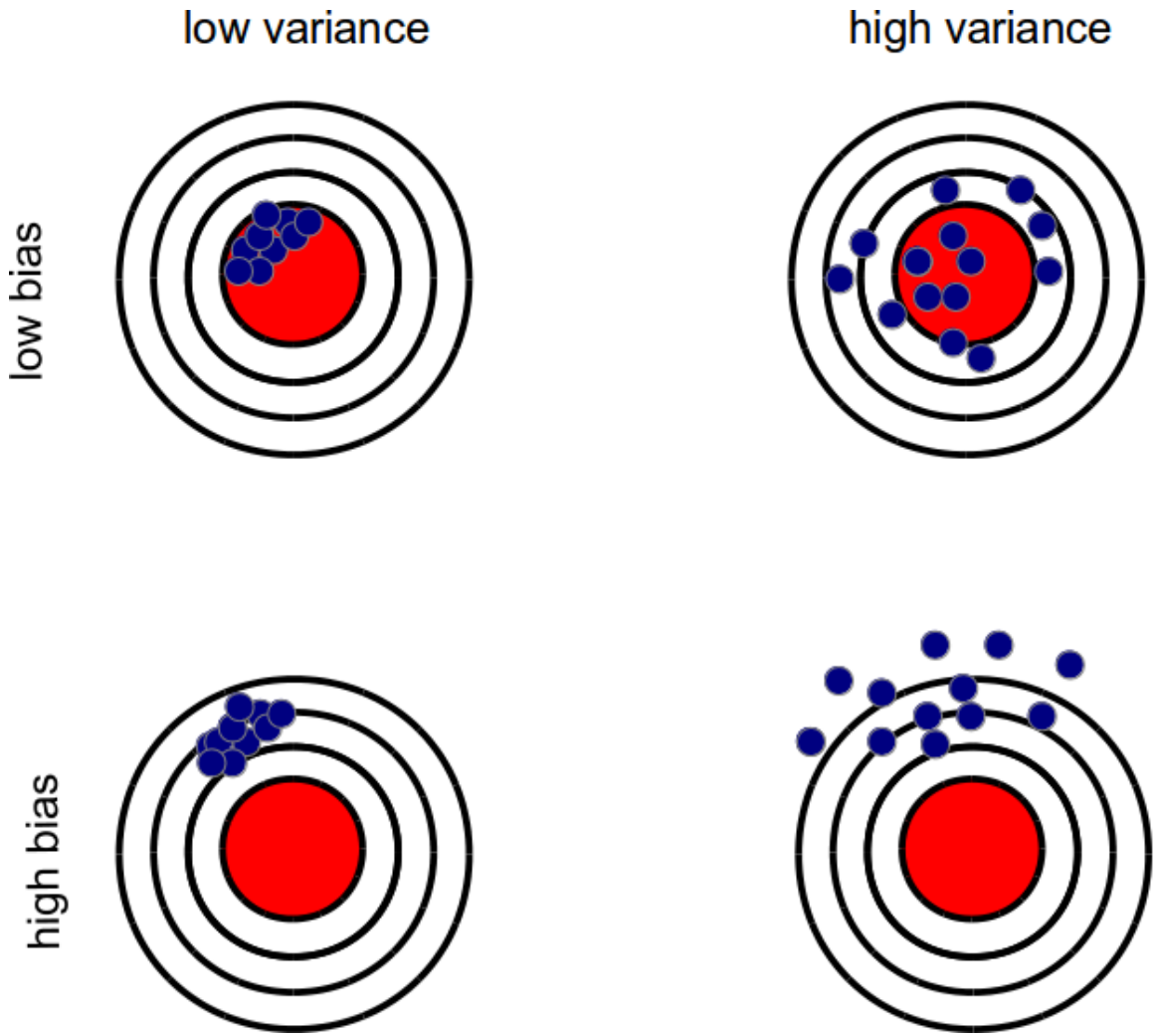
Bias and variance

1. Select a machine learning algorithm
2. Get k different training sets
3. Get k predictors

- **Bias** measures error that originates from the learning algorithm
  – how far off in general the predictions by k predictors are from the true output value

- **Variance** measures error that originates from the training data
  – how much the predictions for a test instance vary between k predictors

Bias and variance



Bias and variance

Generalization error $\text{error}_{\mathcal{D}}(\hat{f})$ measures how well a hypothesis $\hat{f}$ (f is a true target function) generalizes beyond the used training data set, to unseen data with distribution $\mathcal{D}$. Usually it is defined as follows

- for **regression**: $\text{error}_{\mathcal{D}}(\hat{f}) = \mathsf{E}\left[\hat{y}_i - y_i\right]^2$
- for **classification**: $\text{error}_{\mathcal{D}}(\hat{f}) = \mathsf{Pr}\left(\hat{y}_i \neq y_i\right)$

Decomposition of $error_{\mathcal{D}}(\hat{f}) = \text{Bias}^2 + \text{Variance} + \text{IrreducibleError}$

For simplicity, ignore IrreducibleError.

Bias and variance

Regression

$$error_{\mathcal{D}}(\hat{f}) = (E[\hat{f}(\mathbf{x})] - f(\mathbf{x}))^2 + E[(\hat{f}(\mathbf{x}) - E[\hat{f}(\mathbf{x})])^2]$$

Bias and variance

Classificaton

Zero-one (0-1) loss function $\mathrm{L}(\hat{y}, y) = \mathrm{I}(\hat{y}y \leq 0)$, *indicator variable* I is 1 if $y\hat{y} \leq 0$, 0 otherwise

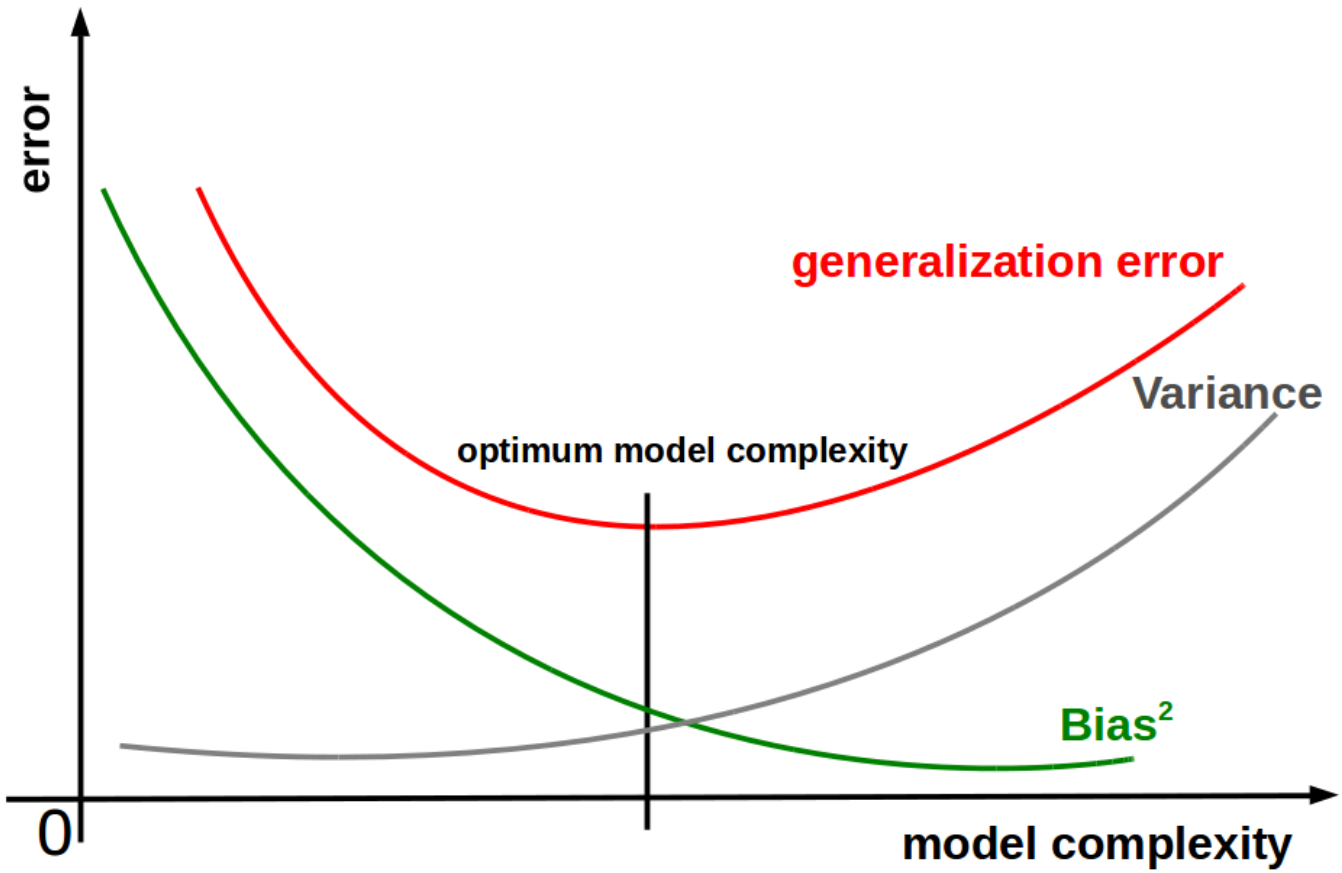
	regression	classification
Single loss	RSS	0-1
Expexted loss	$E[(y - \hat{y})^2]$	$E[\mathrm{L}(y, \hat{y})]$
Main prediction $E[\hat{y}]$	mean	majority vote
Bias^2	$(y - E[\hat{y}])^2$	$\mathrm{L}(y, E[\hat{y}])$
Variance	$E[(E[\hat{y}] - \hat{y})^2]$	$E[\mathrm{L}(\hat{y}, E[\hat{y}])]$

For more details see

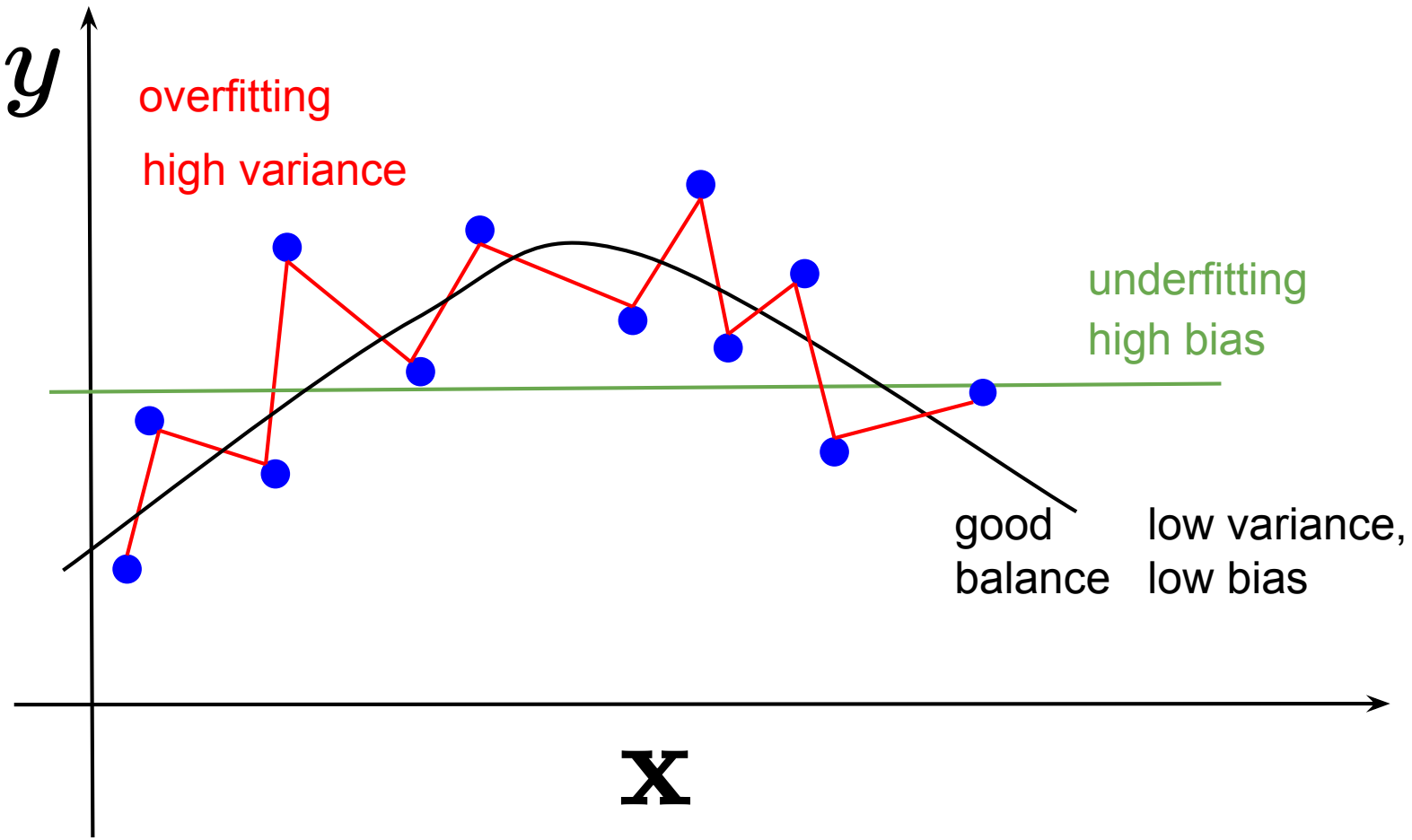
- Thomas G Dietterich and Eun Bae Kong. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Tech. rep. Technical report, Department of Computer Science, Oregon State University, 1995. [url]
- Pedro Domingos. "A unified bias-variance decomposition". In: Proceedings of 17th International Conference on Machine Learning. 2000, pp. 231–238. [url]

Bias and variance

- underfitting = high bias
- overfitting = high variance



Bias and variance



Prevent overfitting

We want a model in between which is

- powerful enough to model the underlying structure of data
- not so powerful to model the structure of the training data

Let's prevent overfitting by **complexity regularization**,
a technique that regularizes the parameter estimates, or equivalently, shrinks the parameter estimates towards zero.

Regularization

A machine learning algorithm
estimates hypothesis parameters $\Theta = \langle \theta_0, \theta_1, \ldots, \theta_m \rangle$
using $\Theta^\star$ that minimizes loss function L
for training data $Data = \{\langle \mathbf{x}_i, y_i \rangle, \mathbf{x}_i = \langle x_{1i}, \ldots, x_{mi} \rangle, y_i \in Y\}$

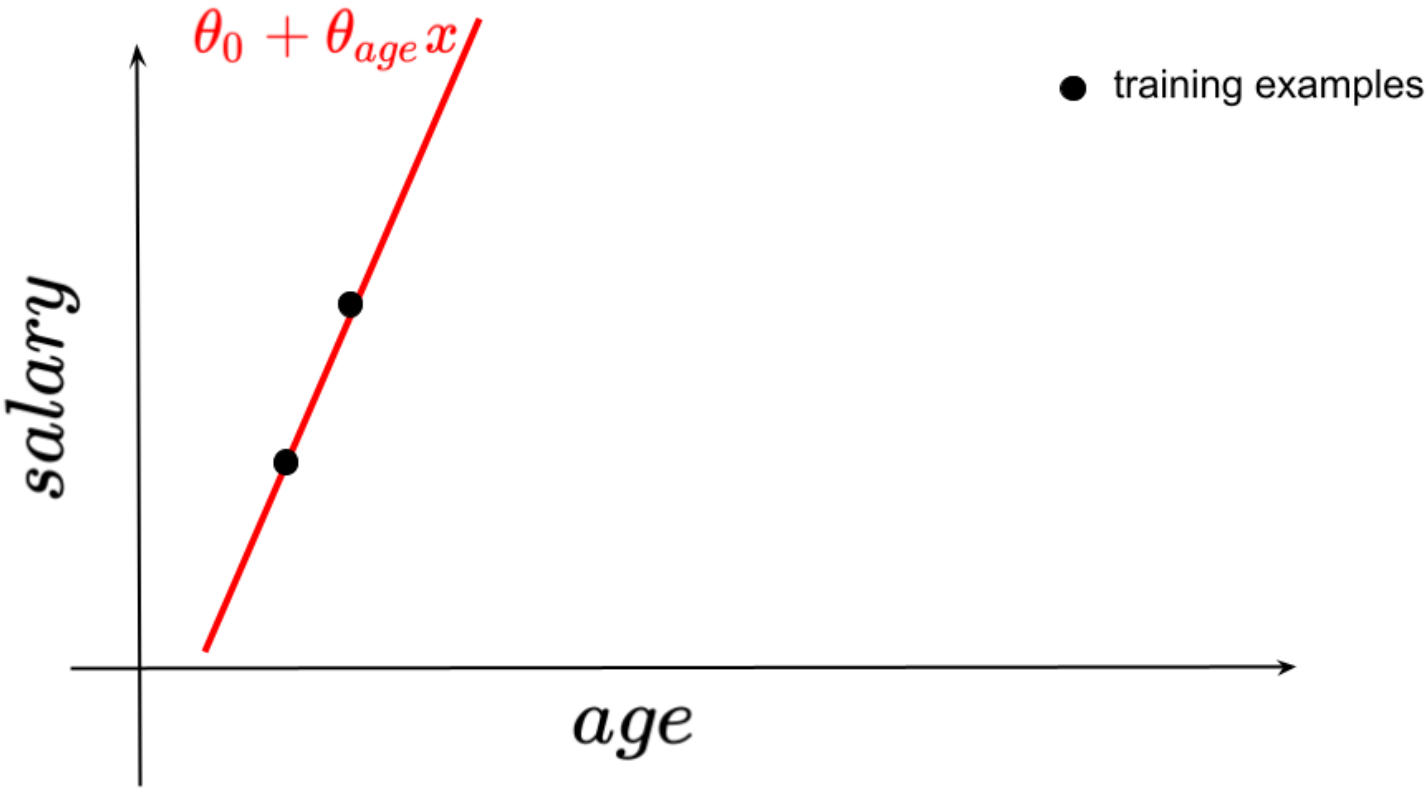
$$\Theta^\star = \mathrm{argmin}_\Theta \mathrm{L}(\Theta)$$

Regularization

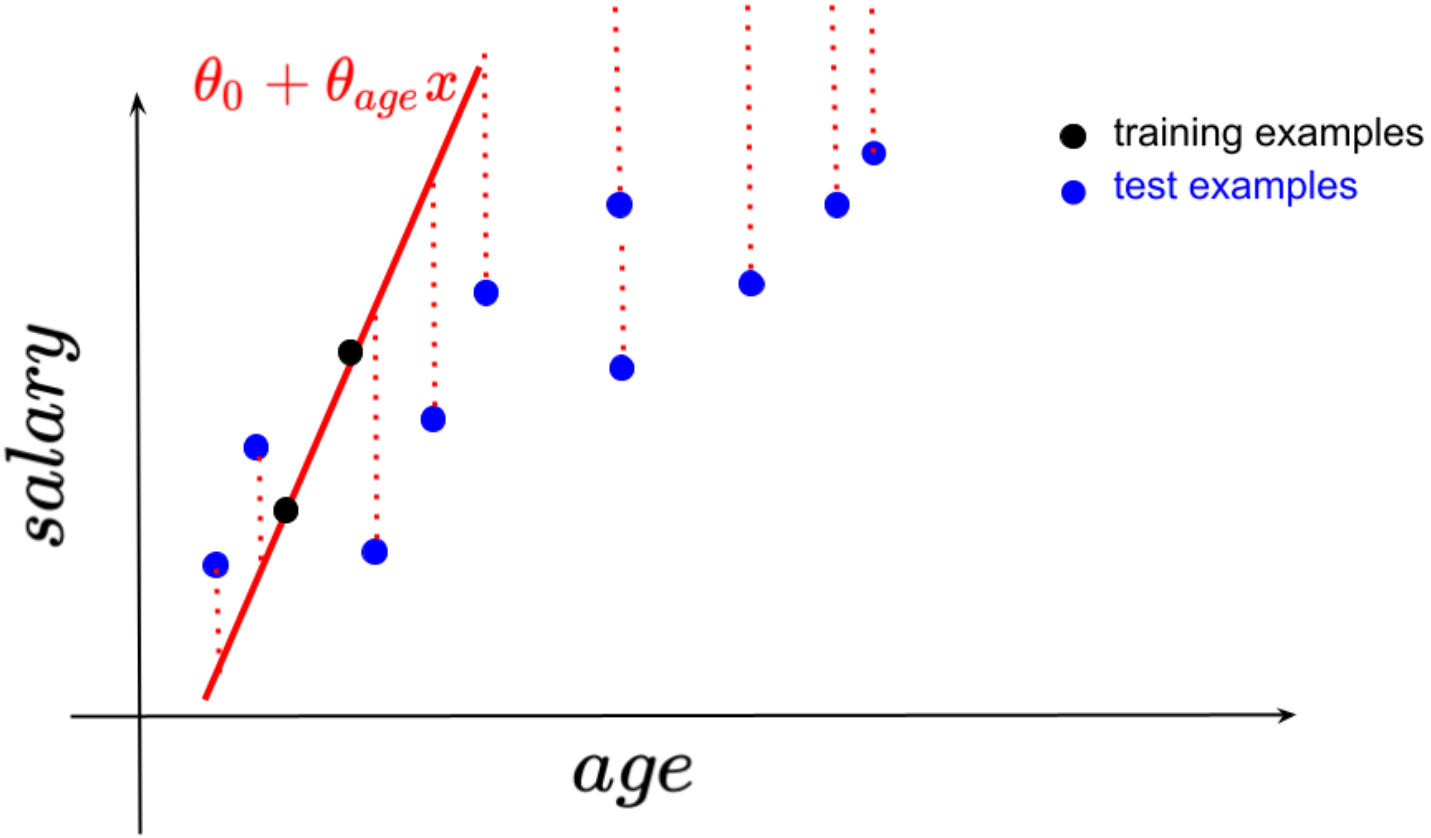
$$\Theta^\star_R = \mathrm{argmin}_\Theta \mathrm{L}(\Theta) + \lambda \cdot \textbf{penalty}(\Theta), \text{ where } \lambda \geq 0 \text{ is a tuning parameter}$$

Infact, the penalty is applied to $\theta_1, \ldots, \theta_m$, but not to θ_0 since the goal is to regularize the estimated association between each feature and the target value.

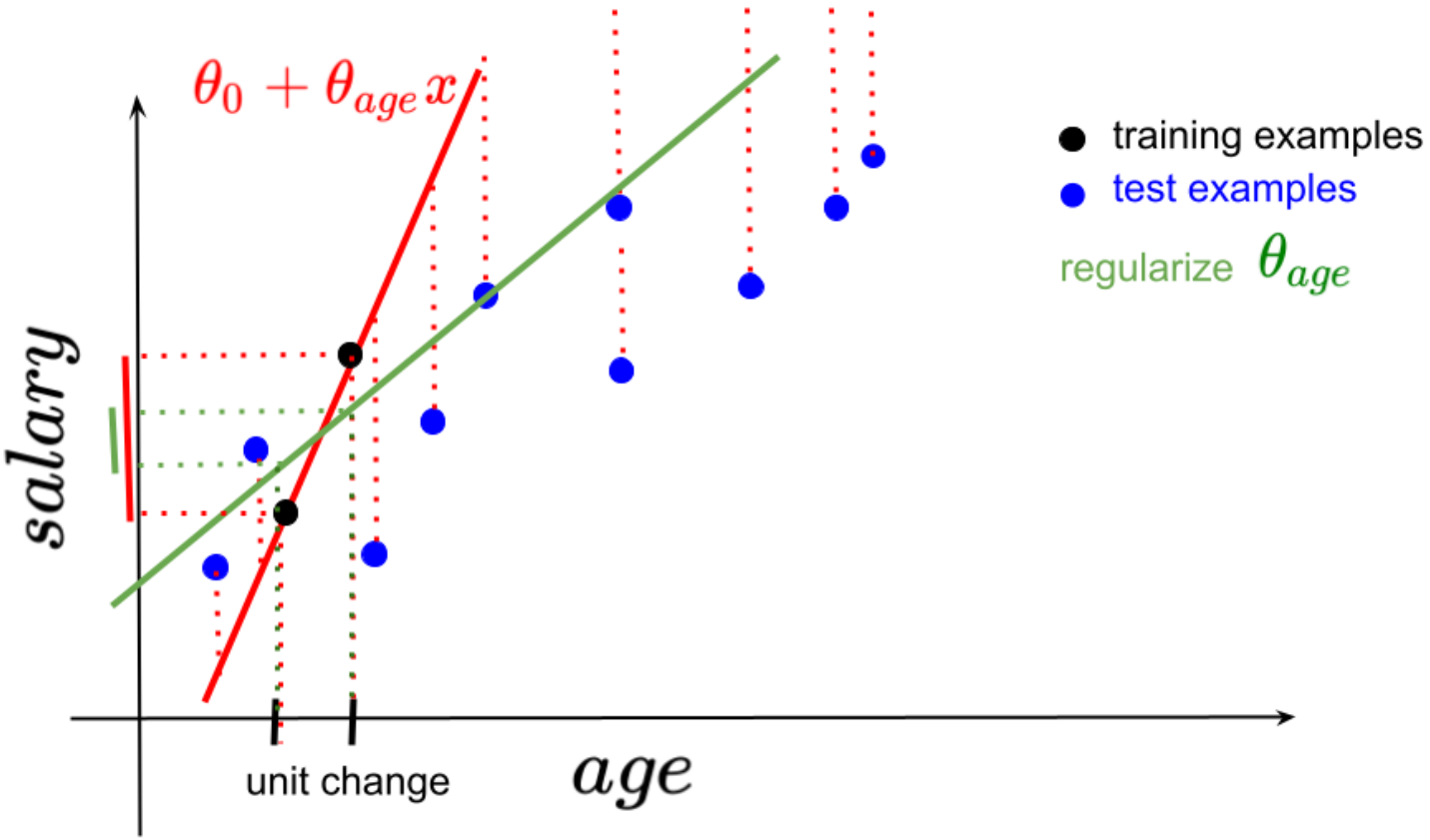
Regularized linear regression



Regularized linear regression



Regularized linear regression



Regularized linear regression

$$f(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m$$

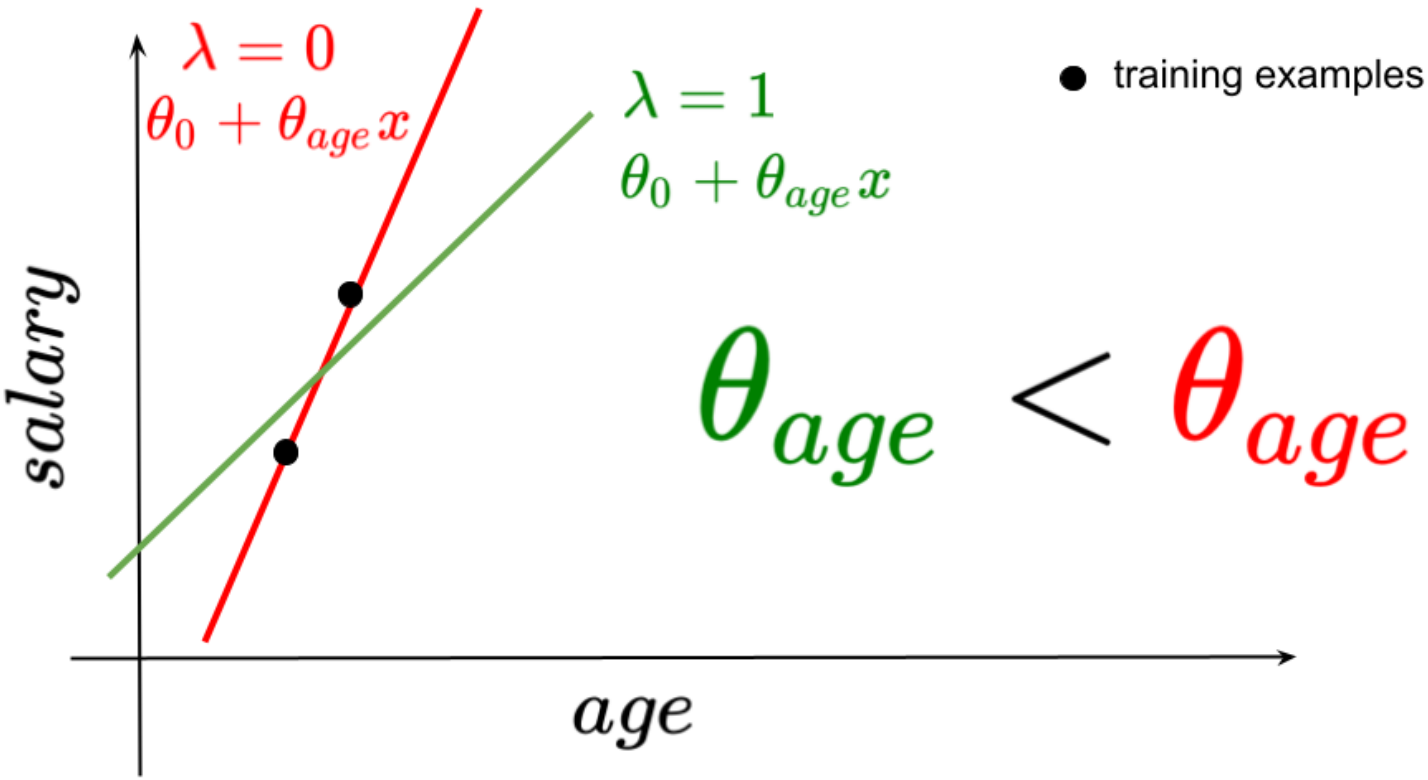
$$\mathrm{L}(\Theta) = RSS = \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2$$

$$\Theta_R^\star = \mathrm{argmin}_\Theta [RSS + \lambda \cdot \mathsf{penalty}(\Theta)]$$

Ridge regression

$$\Theta_R^\star = \text{argmin}_\Theta[RSS + \lambda \cdot (\theta_1^2 + \cdots + \theta_m^2)]$$

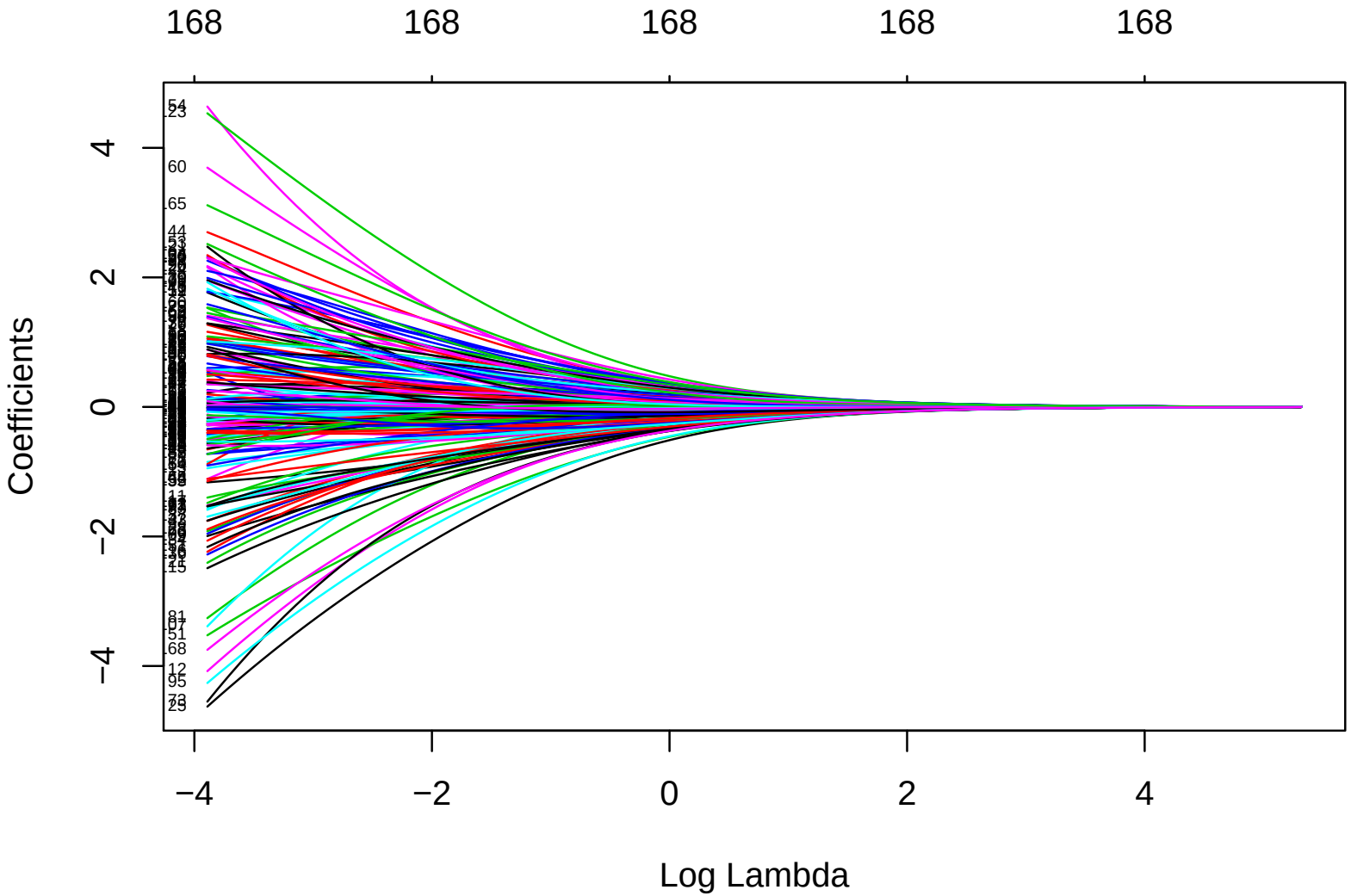
The larger λ, θ_{age} gets asymptotically closer to 0 and *salary* is less sensitive to *age*



Ridge regression

- Let $\theta^\star_{\lambda_1}, \ldots, \theta^\star_{\lambda_m}$ be ridge regression parameter estimates for a particular value of λ
- Let $\theta^\star_1, \ldots, \theta^\star_m$ be unregularized parameter estimates
- $0 \leq \frac{\theta^{\star 2}_{\lambda_1} + \cdots + \theta^{\star 2}_{\lambda_m}}{\theta^{\star 2}_1 + \cdots + \theta^{\star 2}_m} \leq 1$ ... the amount that the ridge regression parameter estimates have been shrunken towards; a small value indicates that they have been shrunken very close to zero
- **When** $\lambda = 0$, **then** $\theta^\star_{\lambda_i} = \theta^\star_i$ for $i = 1, \ldots, m$
- **When** λ is extremely large, **then** $\theta^\star_{\lambda_i}$ is very small for $i = 1, \ldots, m$
- **When** λ between, we are fitting a model and skrinking the parameteres

Ridge regression

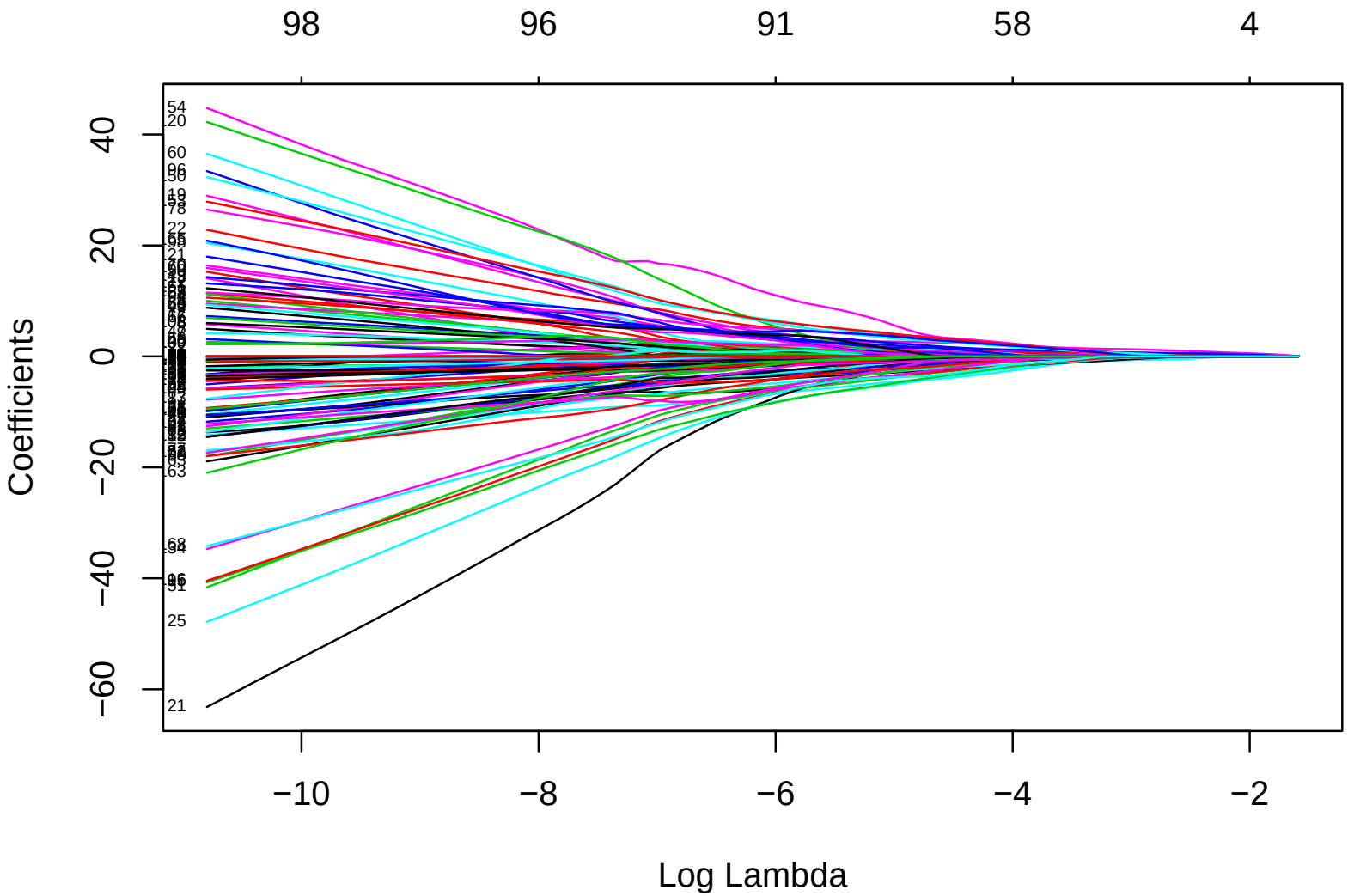


Lasso

$$\text{penalty}(\Theta) = |\theta_1| + \cdots + |\theta_m|$$

- Let $\theta^\star_{\lambda_1}, \ldots, \theta^\star_{\lambda_m}$ be lasso regression parameter estimates
- Let $\theta^\star_1, \ldots, \theta^\star_m$ be unregularized parameter estimates
- **When** $\lambda = 0$, **then** $\theta^\star_{\lambda_i} = \theta^\star_i$ for $i = 1, \ldots, m$
- **When** λ grows, **then** the impact of penalty grows
- **When** λ is extremely large, **then** $\theta^\star_{\lambda_i} = 0$ for $i = 1, \ldots, m$

Lasso



Ridge regression and Lasso

Ridge regression shrinks all the parameters but eliminates none, while the Lasso can shrink some parameters to zero.

Elastic net

$$\text{penalty}(\Theta) = \lambda_1 \cdot (|\theta_1| \ + \cdots + |\theta_m|) + \lambda_2 \cdot (\theta_1^2 + \cdots + \theta_m^2)]$$

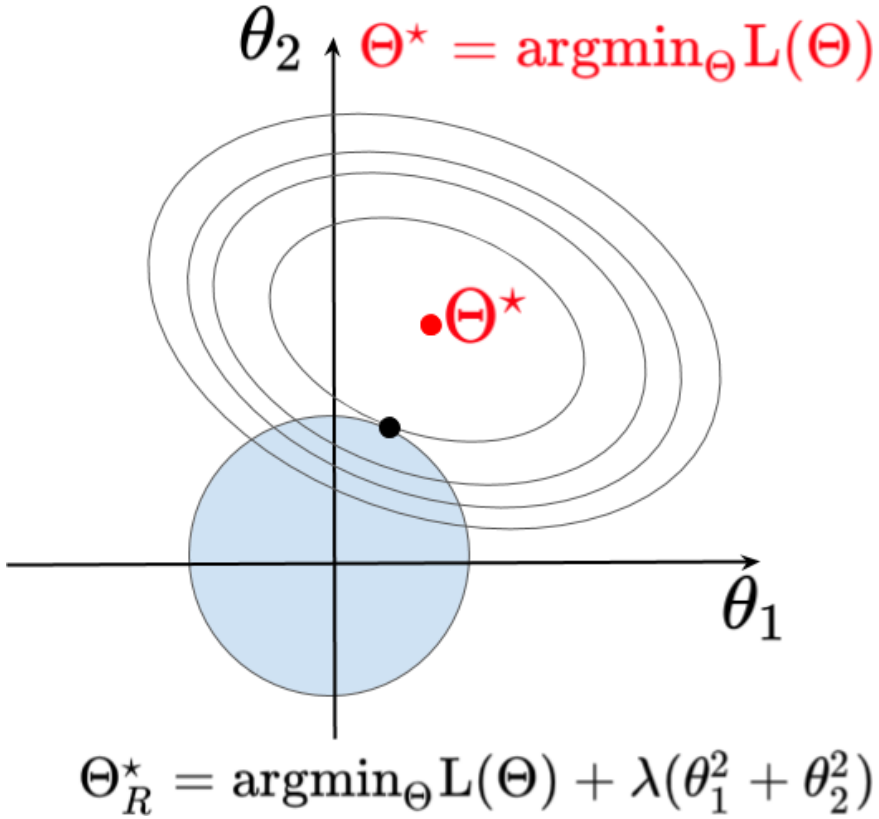
$0 \leq \lambda_1, \lambda_2$ are tuning parameters

Ridge regression
Alternative formulation

$$\Theta_R^\star = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2$$

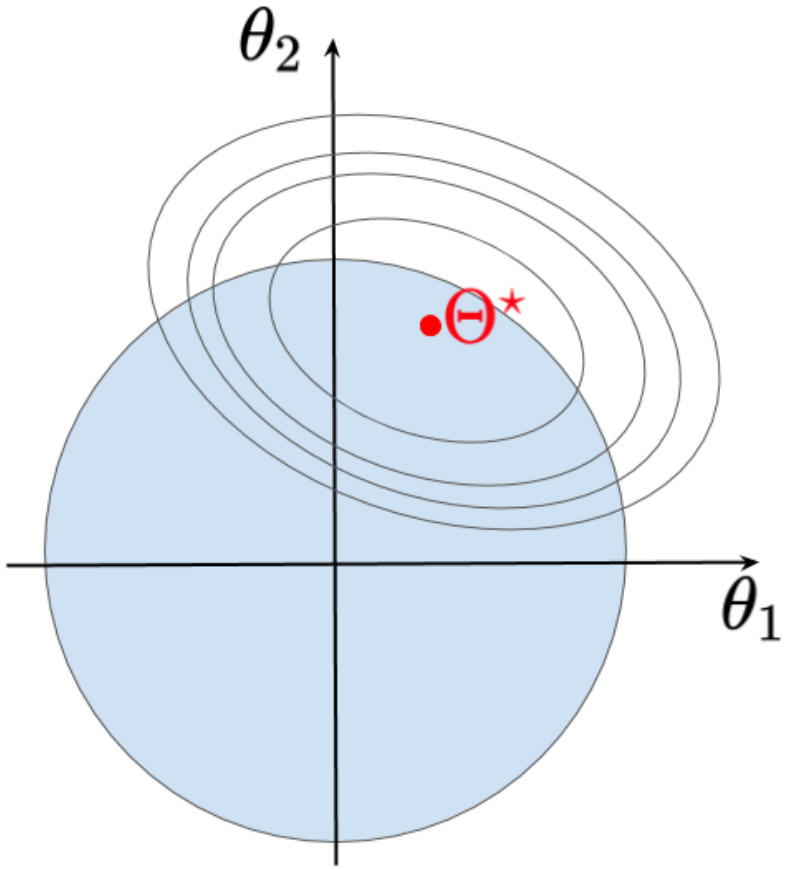
subject to $\theta_1^2 + \cdots + \theta_m^2 \leq s$

- the gray circle represents the feasible region for Ridge regression
- the contours represent different RSS values for the unregularized model



Ridge regression
Alternative formulation

- If s is large enough, i.e. $\lambda = 0$, so that the minimum RSS value falls into the region of **ridge regression** parameter estimates then the alternative formulation yields the least square estimates.

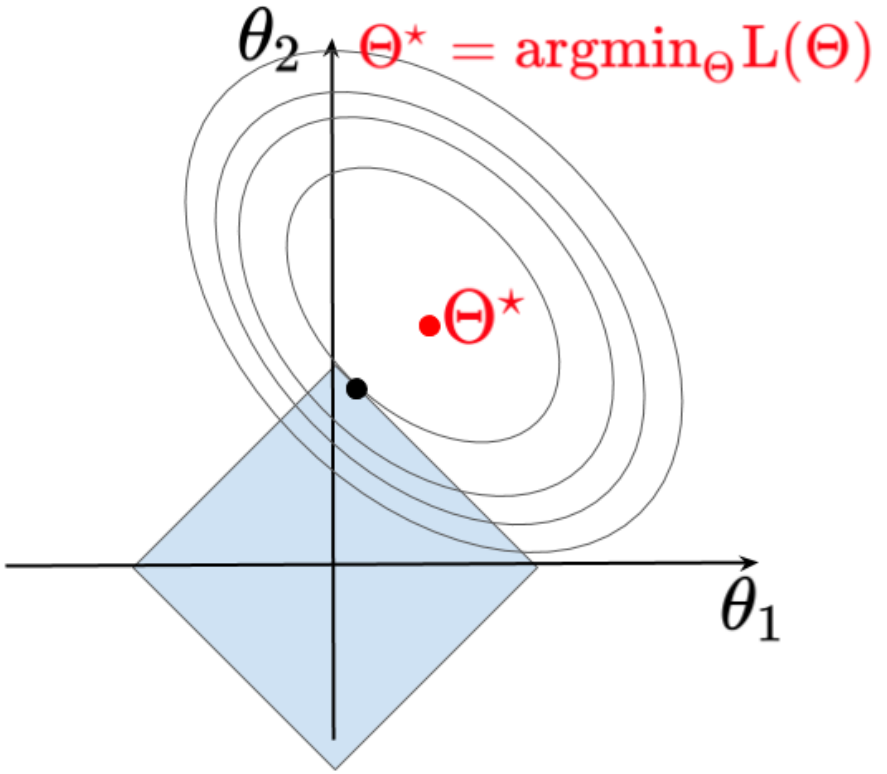


Lasso
Alternative formulation

$$\Theta_R^\star = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2$$

subject to $|\theta_1| + \cdots + |\theta_m| \leq s$

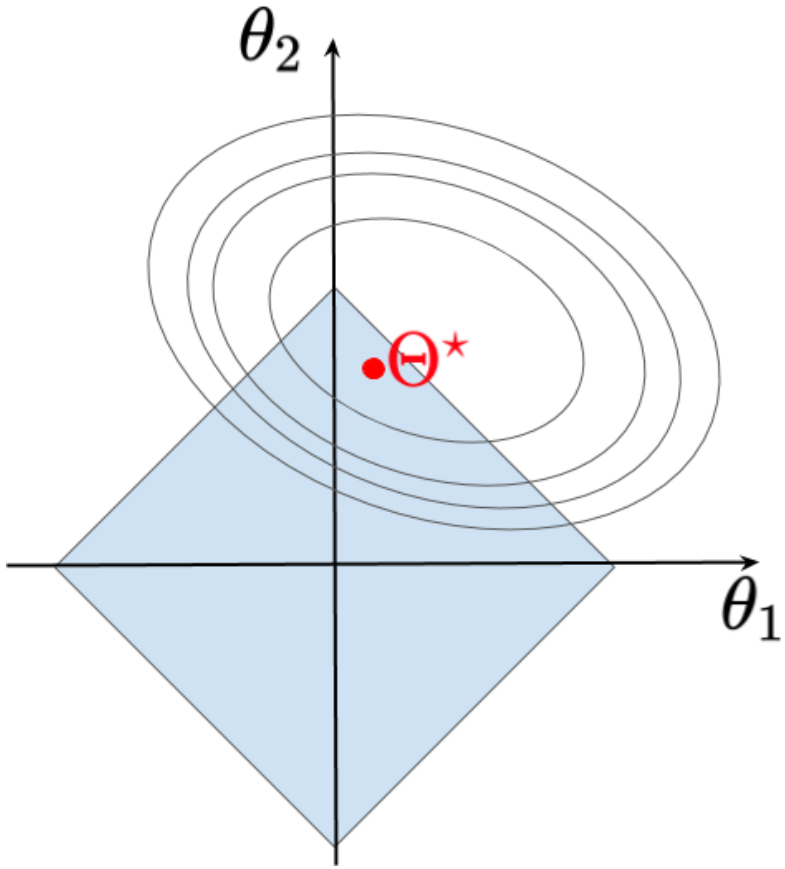
- the grey square represents the feasible region of the Lasso
- the contours represent different RSS values for the unregularized model



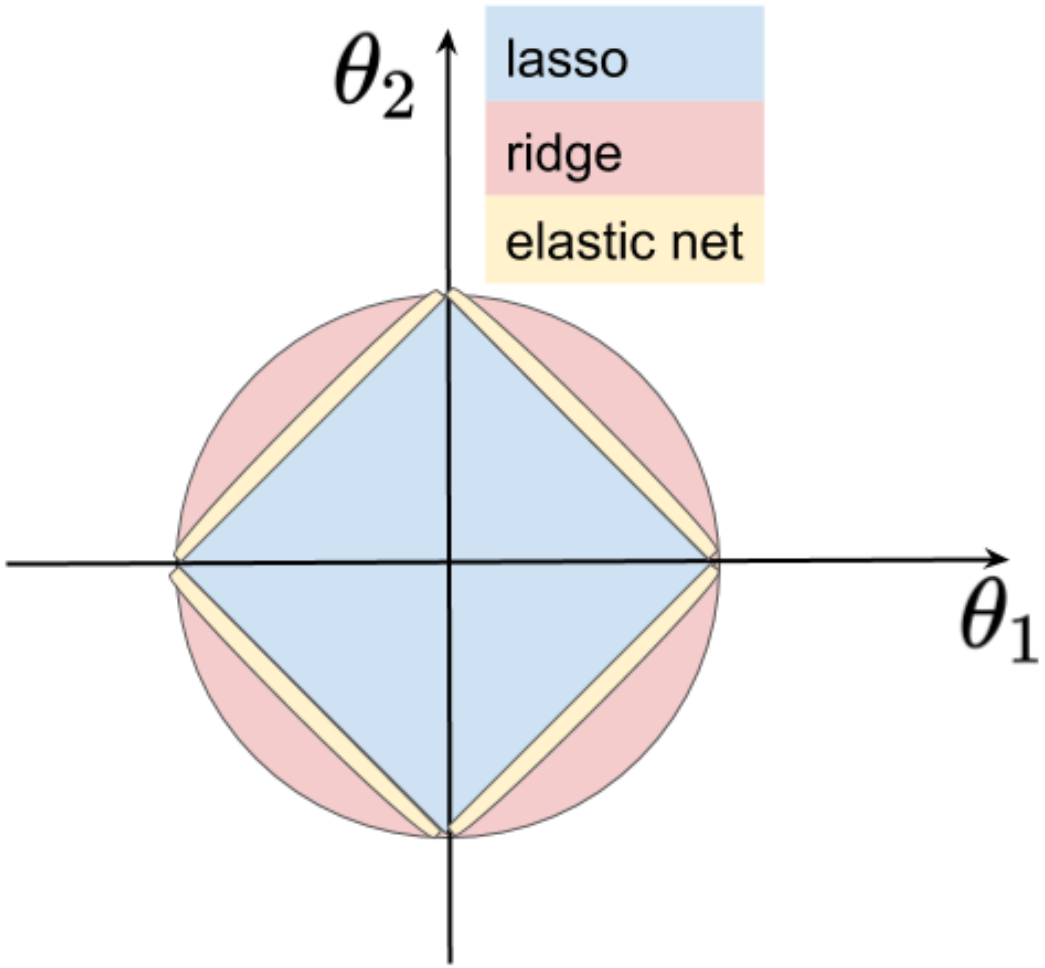
$$\Theta_R^\star = \operatorname{argmin}_\Theta \mathrm{L}(\Theta) + \lambda(|\theta_1| + |\theta_2|)$$

Lasso
Alternative formulation

- If s is large enough, i.e. $\lambda = 0$, so that the minimum RSS value falls into the region of **loss** parameter estimates then the alternative formulation yields the primary solution.



Elastic net

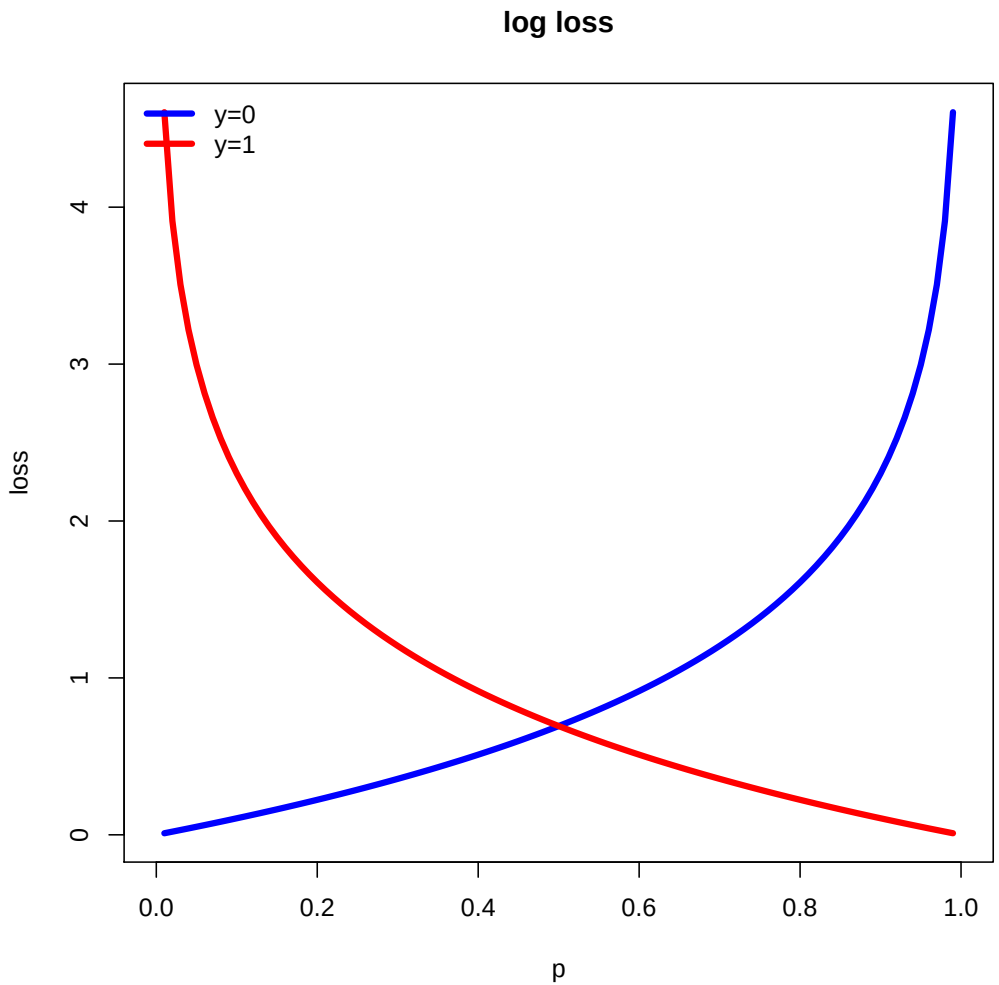


Logistic regression

$$f(\mathbf{x}) = \frac{1}{1 + e^{-\Theta^\top \mathbf{x}}}$$

Logistic regression

$$\mathrm{L}(\Theta) = -\sum_{i=1}^{n} y_i \log \mathrm{P}(y_i|\mathbf{x_i};\Theta) + (1-y_i)\log(1-\mathrm{P}(y_i|\mathbf{x_i};\Theta))$$



Regularized logistic regression
Ridge regression

$$\Theta_R^\star = \text{argmin}_\Theta - [\sum_{i=1}^{n} y_i \log(f(\mathbf{x}_i)) + (1 - y_i) \log(1 - f(\mathbf{x}_i))] + \lambda \sum_{j=1}^{m} \theta_j^2] =$$

$$= \text{argmin}_\Theta [\sum_{i=1}^{n} y_i (-\log(f(\mathbf{x}_i))) + (1 - y_i)(-\log(1 - f(\mathbf{x}_i))) + \lambda \sum_{j=1}^{m} \theta_j^2] =$$

$$= \text{argmin}_\Theta [\sum_{i=1}^{n} y_i L_1(\mathbf{\Theta}) + (1 - y_i) L_0(\mathbf{\Theta}) + \lambda \sum_{j=1}^{m} \theta_j^2]$$

Regularized logistic regression
Ridge regression

Since

$$\mathbf{A} + \lambda\mathbf{B} \equiv C\mathbf{A} + \mathbf{B}, C = \frac{1}{\lambda}$$

then

$$\Theta_R^\star = \text{argmin}_\Theta[\sum_{j=1}^{m} \theta_j^2 + C[\sum_{i=1}^{n} y_i L_1(\Theta) + (1 - y_i) L_0(\Theta)]]$$

where

$L_1(\Theta) = -\log(1/1 + e^{-\Theta^\top \mathbf{x}})$

$L_0(\Theta) = -\log(1 - 1/1 + e^{-\Theta^\top \mathbf{x}})$

Regularized logistic regression
Ridge regression

$$\Theta_R^\star = \text{argmin}_\Theta [\sum_{j=1}^{m} \theta_j^2 + C \sum_{i=1}^{n} \log(1 + e^{-\overline{y}_i \Theta^\top \mathbf{x}_i})]$$

where

$$\overline{y}_i = \begin{cases} -1 \ if & y_i = 0 \\ +1 \ if & y_i = 1 \end{cases}$$

SVM
Soft margin classifier

$$\Theta^\star = \text{argmin}_\Theta \sum_{j=1}^{m} \theta_j^2 + C \sum_{i=1}^{n} \xi_i$$

$\xi_i \geq 0$ is equivalent to $\xi_i = \max(0, 1 - y_i \Theta^\top \mathbf{x}_i)$, i.e.

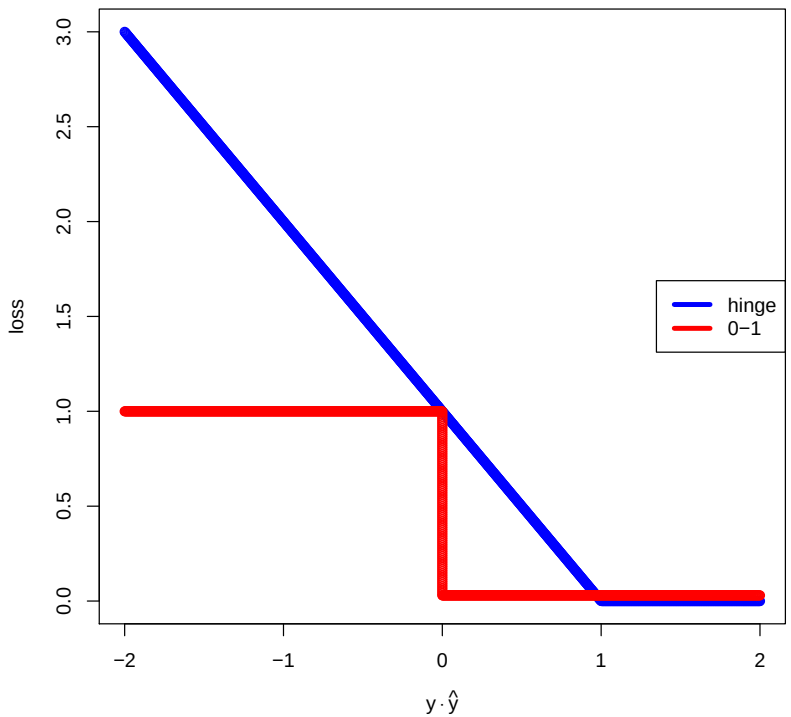
$$\Theta^\star = \text{argmin}_\Theta [\sum_{j=1}^{m} \theta_j^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i \Theta^\top \mathbf{x}_i)]$$

s.t. $\Theta^\top \mathbf{x}_i \geq 1 - \xi_i$ if $y_i = +1$ and $\Theta^\top \mathbf{x}_i \leq -1 + \xi_i$ if $y_i = -1$

SVM
Soft margin classifier

Hinge loss $= \max(0, 1 - y_i \Theta^\top \mathbf{x}_i)$

1. $y_i \Theta^\top \mathbf{x}_i > 1$: no contribution to loss
2. $y_i \Theta^\top \mathbf{x}_i = 1$: no contribution to loss
3. $y_i \Theta^\top \mathbf{x}_i < 1$: contribution to loss



SVM
Soft margin classifier

Soft-margin is equivalent to the regularization problem.

Instance-based learning
Key idea

- IBL methods = supervised ML methods
- IBL methods initially store training data, we call them *lazy* methods
- For a new instance, prediction is based on local similarity,
  i.e. a set of similar instances are retrieved and used for prediction
- IBL methods can construct a different approximation of a target function for each distinct test instance
- Both classification and regression

Instance-based learning
Key points

1. A distance metric
2. How many nearby neighbours look at?
3. A weighting function
4. How to fit with local points?

Instance-based learning
Distance metric

Recall distance used as dissimilarity metrics for clustering. The most common ones

- **Euclidean distance**

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sqrt{\sum_{r=1}^{m} (x_{i_r} - x_{j_r})^2}$$

- **Manhattan distance**

$$d(\mathbf{x_i}, \mathbf{x_j}) = \sum_{r=1}^{m} |x_{i_r} - x_{j_r}|$$

Instance-based learning
k-Nearest Neighbour algorithm

1. **A distance metric**: Euclidian (most widely used)
2. **How many nearby neighbours look at?** k training instances closest to **x**
3. **A weighting function**: unused
4. **How to fit with local points?**

- **k-NN classification**

$$f(\mathbf{x}) = \text{argmax}_{v \in Y} \sum_{i=1}^{k} \delta(v, y_i), \tag{1}$$

where $\delta(a, b) = 1$ if $a = b$, otherwise 0

- **k-NN regression**

$$f(\mathbf{x}) = \sum_{i=1}^{k} y_i / k \tag{2}$$

Instance-based learning
Distance-weighted k-NN algorithm

1. **A distance metric**: Euclidian (most widely used)
2. **How many nearby neighbours look at?** k training instances closest to $\mathbf{x}$
3. **A weighting function**: greater weight of closer neighbours, e.g.,

$$w_i(\mathbf{x}) \equiv \frac{1}{d(\mathbf{x}, \mathbf{x_i})^2}$$

4. **How to fit with local points?**

- **Classification**

$$f(\mathbf{x}) = \text{argmax}_{v \in Y} \sum_{i=1}^{k} w_i(\mathbf{x}) \delta(v, y_i) \tag{3}$$

- **Regression**

$$f(\mathbf{x}) = \sum_{i=1}^{k} w_i(\mathbf{x}) y_i / \sum_{i=1}^{k} w_i(\mathbf{x}) \tag{4}$$

Instance-based learning
Distance-weighted k-NN algorithm

Shepard's method

- **Classification**

$$f(\mathbf{x}) = \text{argmax}_{v \in Y} \sum_{i=1}^{n} w_i(\mathbf{x}) \delta(v, y_i) \tag{5}$$

- **Regression**

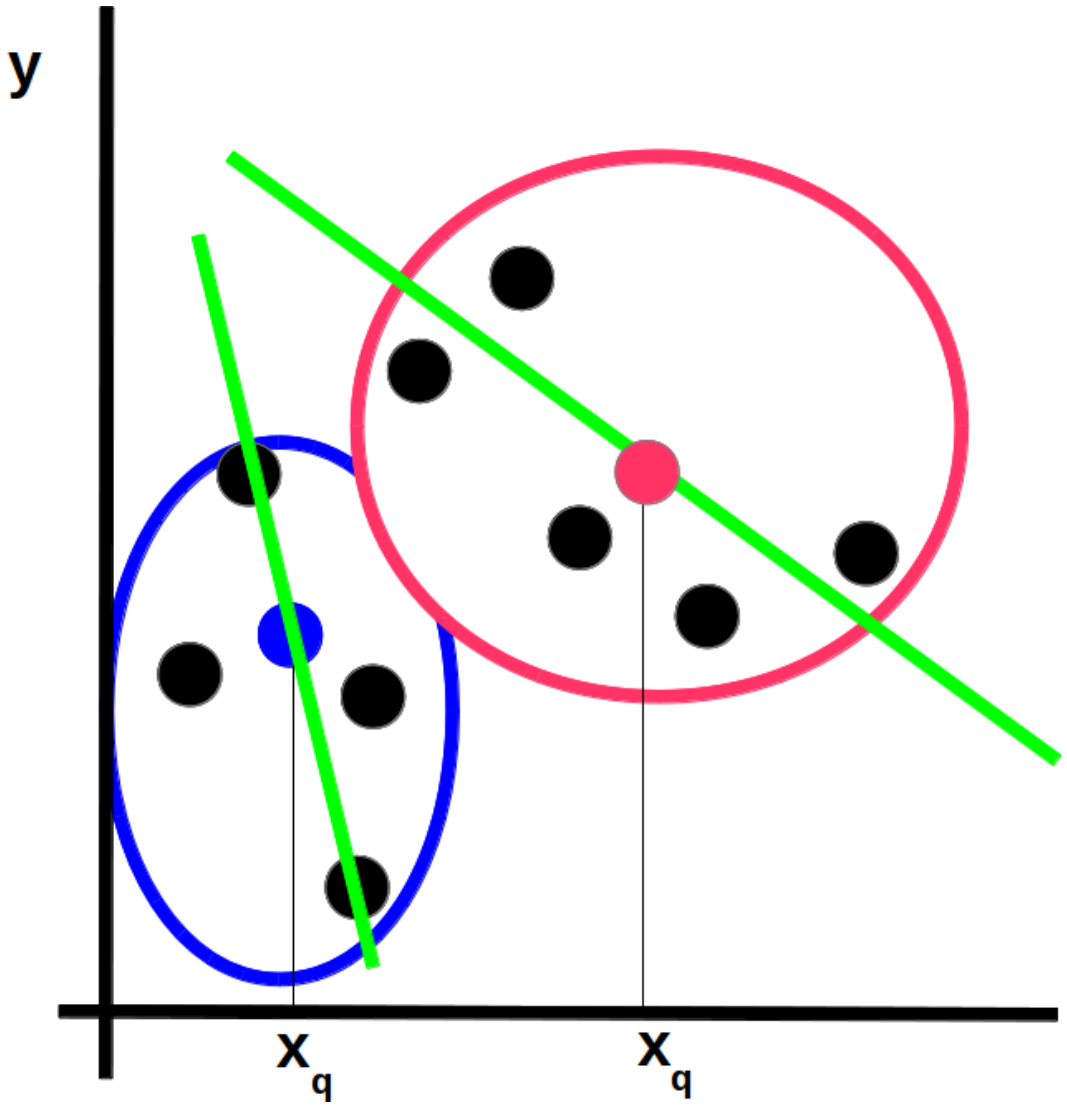
$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i(\mathbf{x}) y_i / \sum_{i=1}^{n} w_i(\mathbf{x}) \tag{6}$$

Instance-based learning
Locally weighted linear regression

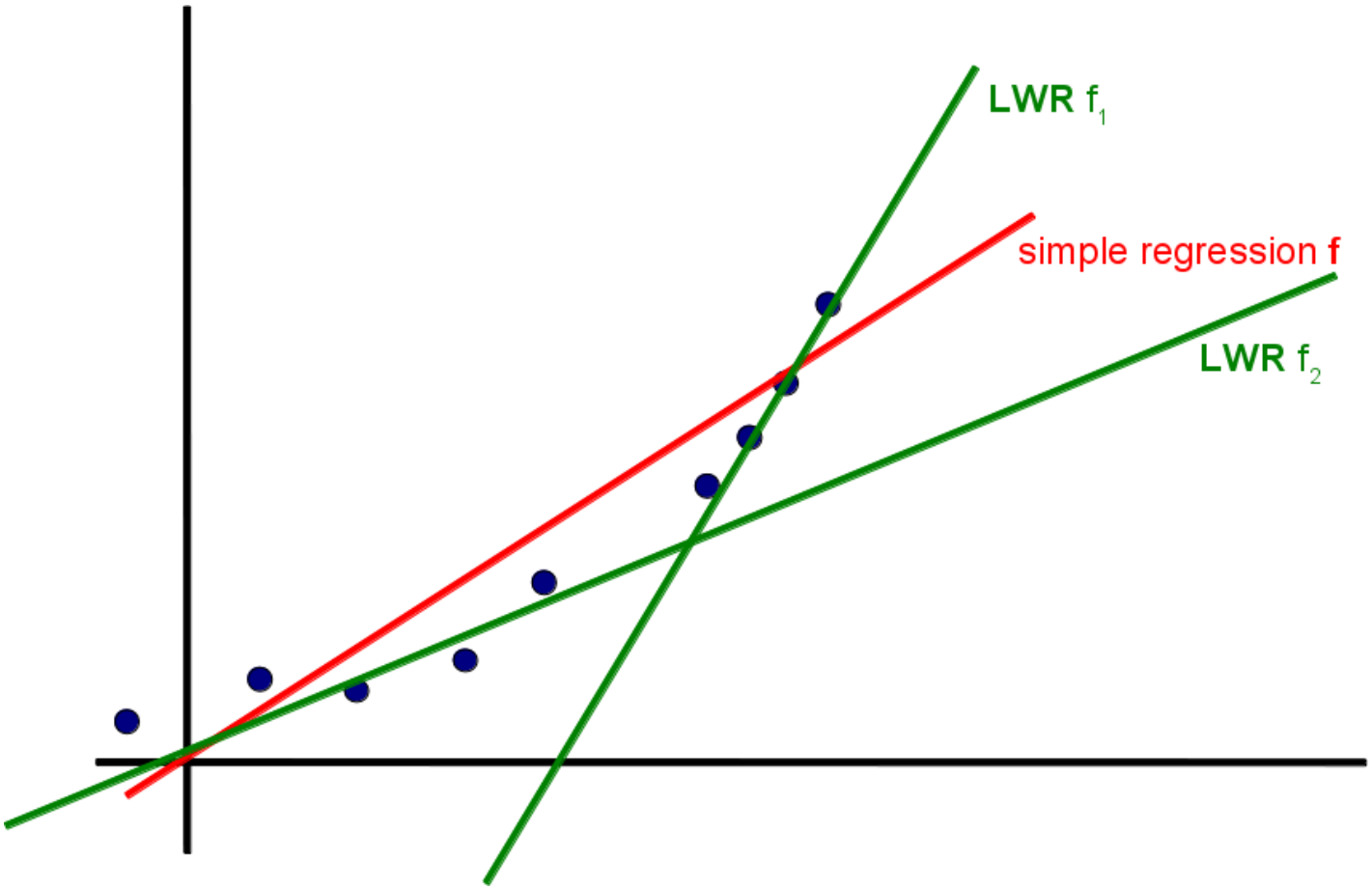
1. **A distance metric**: Euclidian (most widely used)
2. **How many nearby neighbours look at?** k training instances closest to $\mathbf{x}$
3. **A weighting function**: $w_i(\mathbf{x})$
4. **How to fit with local points?**

$$\mathbf{\Theta}^\star = \text{argmin}_{\mathbf{\Theta}} \sum_{i=1}^{k} w_i(\mathbf{x})(\Theta^T \mathbf{x}_i - y_i)^2 \tag{7}$$

Instance-based learning
Locally weighted linear regression

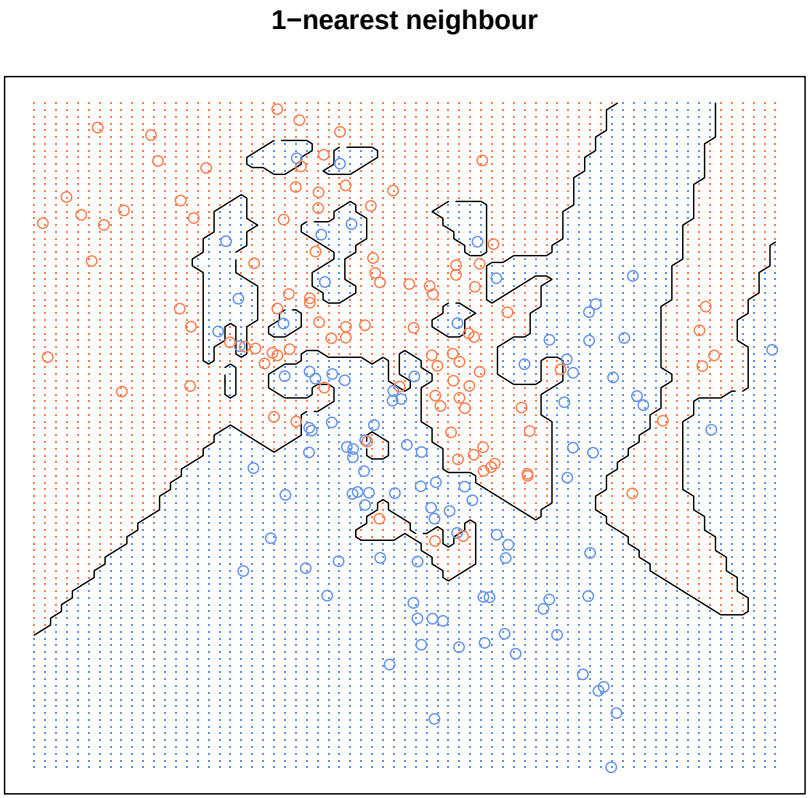


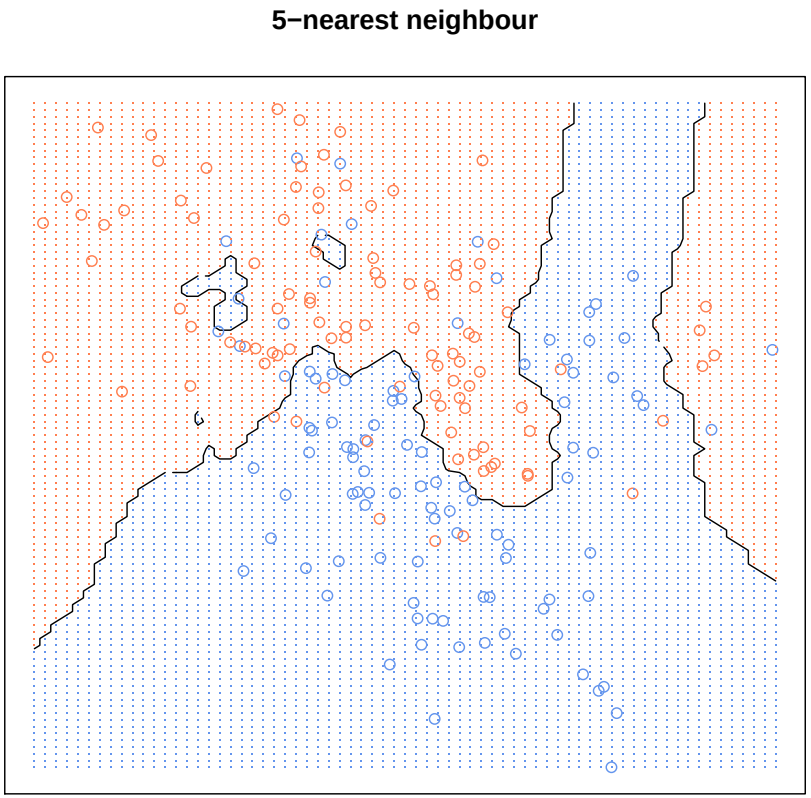
Instance-based learning
LW linear regression vs. simple regression



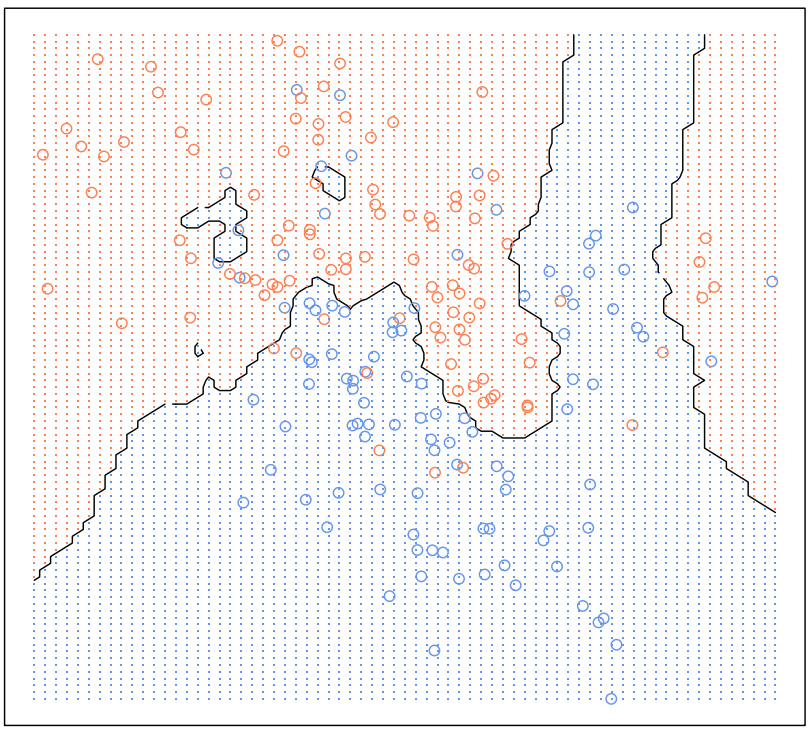
Bias and variance
k-Nearest Neighbor

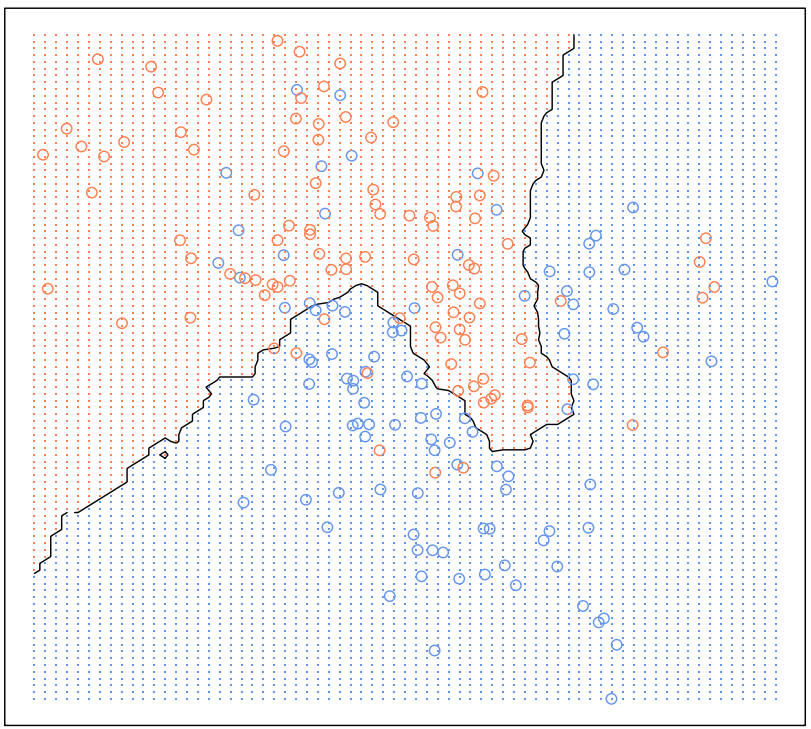
- $\uparrow k \rightarrow$ smoother decision boundary $\rightarrow \downarrow$ variance and $\uparrow$ bias
- $\downarrow k \rightarrow \uparrow$ variance and $\downarrow$ bias





Bias and variance
k-Nearest Neighbor





Summary of Examination Requirements

- Binary classifier using ROC curve (True Positive Rate vs. False Positive Rate)
- Model complexity, generalization error, Bias and variance
- Lasso and Ridge regularization for linear and logistic regression
- Soft margin classifier and regularization
- k-NN algorithm