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

Outline

- Overfitting
- Regularization
  - Ridge regression
  - Lasso
  - Linear regression
  - Logistic regression
  - SVM
- Principal Component Analysis
- Generalization error estimation by bootstrapping

Settings

- Suppose m features $A_1, \ldots, A_m$ and a set of possible target values Y
- Suppose development data as a set of instances

$$Data = \{(\mathbf{x}_i, y_i), \mathbf{x}_i = \langle x_i^1, \ldots, x_i^m \rangle, y_i \in Y\},$$

where $\mathbf{x}_i$ are feature vectors and y_i are the corresponding target values

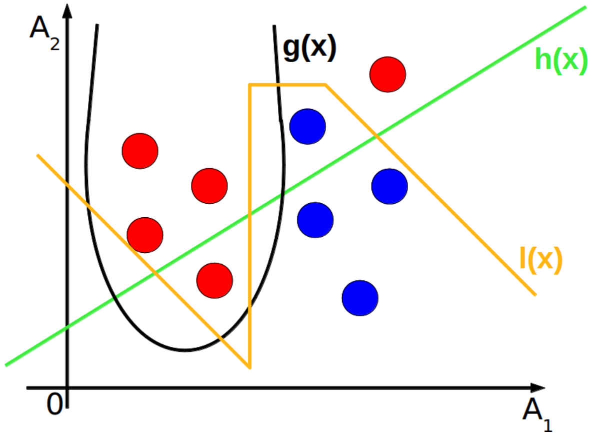
Let $h^\star$ be a best approximation of c trained on $Data$.

Model complexity

Model complexity is the number of hypothesis parameters

$$\boldsymbol{\Theta} = < \Theta_0, \ldots, \Theta_m >$$

Model complexity – example

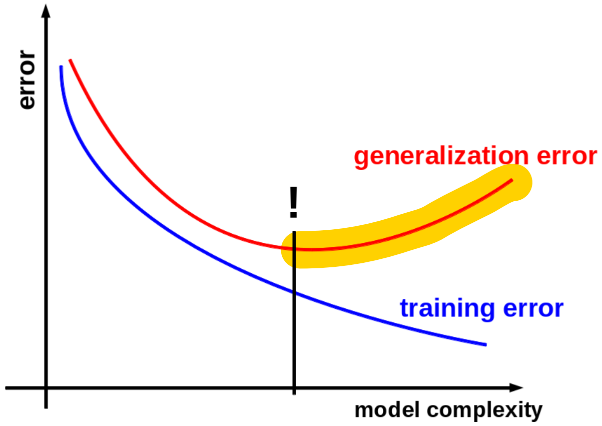


Model complexity – example

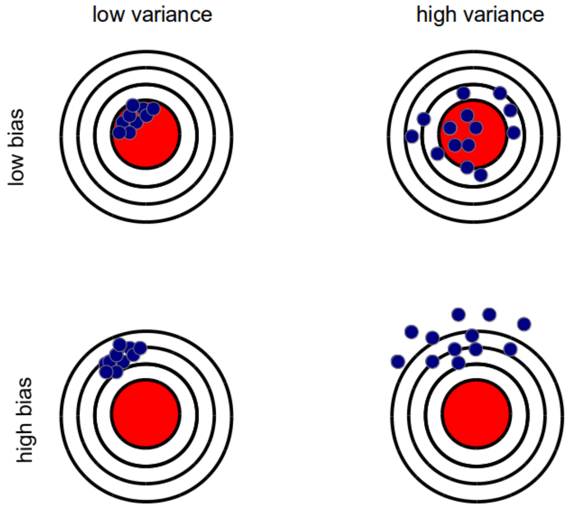
- $h(\mathbf{x})$: a straight line – determined by *two* parameters of the prediction function
  – doesn't fit two examples
- $h_2(\mathbf{x})$: a parabola – determined by *three* parameters of the prediction function
  – doesn't fit one example
- $h_3(\mathbf{x})$: a curve – determined by *many* parameters of the prediction function
  – perfectly fits all examples

Model complexity and overfitting

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



Bias and variance



Bias and variance

Generalization error $\mathrm{error}_{\mathcal{D}}(h)$ measures how well a hypothesis *h* generalizes beyond the used training data set, to unseen data with distribution $\mathcal{D}$. Usually it is defined as follows

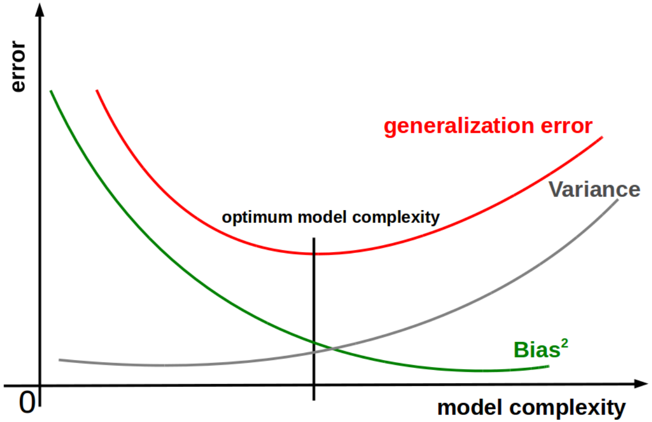
- for **regression**: $\mathrm{error}_{\mathcal{D}}(h) = \mathsf{E}\left(\hat{y}_i - y_i\right)^2$
- for **classification**: $\mathrm{error}_{\mathcal{D}}(h) = \mathsf{Pr}\left(\hat{y}_i \neq y_i\right)$

Decomposition of $\mathit{error}_{\mathcal{D}}(h)$

$$\mathit{error}_{\mathcal{D}}(h) = \mathrm{Bias}^2 + \mathrm{Variance}$$

Bias and variance

- underfitting = high bias
- overfitting = high variance



Regularization

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

$$\mathbf{\Theta} = < \Theta_0, \Theta_1, \ldots, \Theta_m >$$

using $\mathbf{\Theta}^\star$ that minimizes `loss` function for the data D

$$\mathbf{\Theta}^\star = \underset{\mathbf{\Theta}}{\operatorname{argmin}} \operatorname{loss}(\mathbf{\Theta})$$

- **Regularization**

$$\mathbf{\Theta}^\star = \underset{\mathbf{\Theta}}{\operatorname{argmin}} \operatorname{loss}(\mathbf{\Theta}) + \lambda * \textbf{penalty}(\mathbf{\Theta})$$

where $\lambda \geq 0$ is a **tuning parameter**

Regularization – Ridge regression

$$\text{penalty}(\boldsymbol{\Theta}) = \Theta_1^2 + \cdots + \Theta_m^2$$

$\Theta_1^2 + \cdots + \Theta_m^2$ is the ℓ_2 norm

$$\boldsymbol{\Theta}^\star = \operatorname*{argmin}_{\boldsymbol{\Theta}} \text{loss}(\boldsymbol{\Theta}) + \lambda * (\Theta_1^2 + \cdots + \Theta_m^2)$$

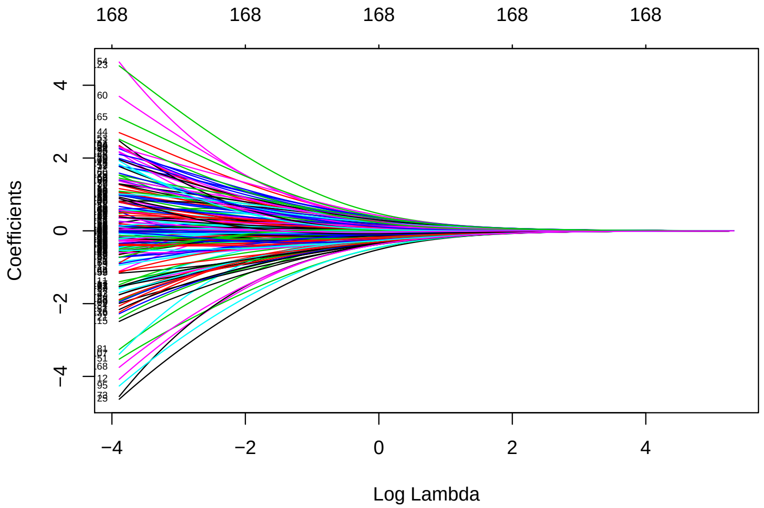
The penalty is applied to $\Theta_1, \ldots, \Theta_m$, but not to Θ_0, since the goal is to regularize the estimated association between each feature and the target value.

Ridge regression

$$\boldsymbol{\Theta}^\star = \underset{\boldsymbol{\Theta}}{\mathrm{argmin}}\ \mathrm{loss}(\boldsymbol{\Theta}) + \lambda * (\Theta_1^2 + \cdots + \Theta_m^2)$$

- **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
- $0 \leq \frac{{\Theta^\star}^2_{\lambda 1} + \cdots + {\Theta^\star}^2_{\lambda m}}{{\Theta^\star_1}^2 + \cdots + {\Theta^\star_m}^2} \leq 1$
- **When** $\lambda = 0$, **then** $\Theta^\star_{\lambda i} = \Theta^\star_i$ for $i = 1, \ldots, m$
- **When** λ is extremely large, **then** $\Theta^\star_{\lambda i} = 0$ for $i = 1, \ldots, m$
- **When** λ between, we are fitting a model and skrinking the parameteres

Ridge regression



Regularization – Lasso

$$\text{penalty}(\boldsymbol{\Theta}) = |\Theta_1| + \cdots + |\Theta_m|$$

$|\Theta_1| + \cdots + |\Theta_m|$ is the ℓ_1 norm

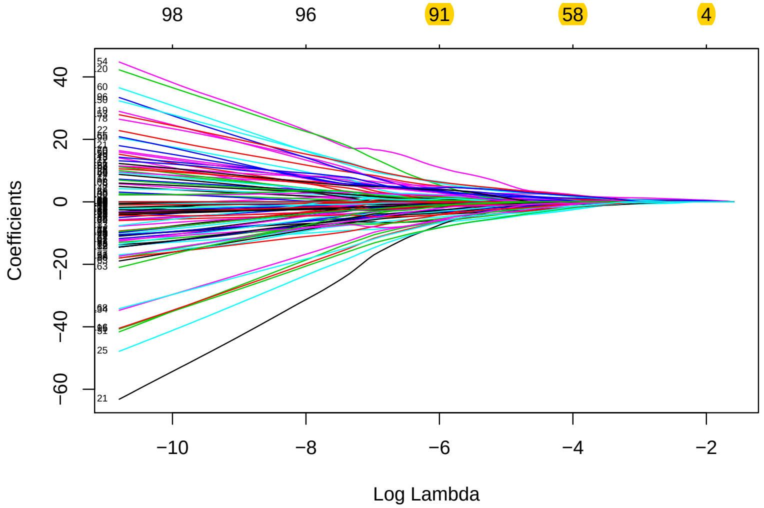
$$\boldsymbol{\Theta}^\star = \operatorname*{argmin}_{\boldsymbol{\Theta}} \text{loss}(\boldsymbol{\Theta}) + \lambda * (|\Theta_1| + \cdots + |\Theta_m|)$$

Lasso

$$\mathbf{\Theta}^\star = \underset{\mathbf{\Theta}}{\mathrm{argmin}}\ \mathrm{loss}(\mathbf{\Theta}) + \lambda * (|\Theta_1| + \cdots + |\Theta_m|)$$

- **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



98
96
91
58
4
Coefficients
40
20
0
−20
−40
−60
−10
−8
−6
−4
−2
Log Lambda
54
20
60
96
30
19
78
22
21
63
68
94
31
25
21

Ridge regression and Lasso – comparison

Difference between Ridge regression and Lasso

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

Loss function

A loss function $\mathrm{L}(\hat{y}, y)$ measures the cost of predicting $\hat{y}$ when the true value is $y \in \{-1, +1\}$. Commonly used loss functions are

- **Squared** (RSS) $\mathrm{L}(\hat{y}, y) = (y - \hat{y})^2$
- **Zero-one** (0/1) $\mathrm{L}(\hat{y}, y) = \mathrm{I}(y\hat{y} \leq 0)$
  indicator variable I is 1 if $y\hat{y} \leq 0$, 0 otherwise
- **Hinge** $\mathrm{L}(\hat{y}, y) = \max(0, 1 - y\hat{y})$
- **Logistic** $\mathrm{L}(\hat{y}, y) = \max(0, \log(1 + e^{-y\hat{y}}))$
- **Exponential** $\mathrm{L}(\hat{y}, y) = e^{-y\hat{y}}$

Recap of linear regression

Linear regression is a regression algorithm

$$\boldsymbol{\Theta}^\star = \operatorname*{argmin}_{\boldsymbol{\Theta}} \sum_{i=1}^{n} (h(\mathbf{x}_i) - y_i)^2$$

where

- $h(\mathbf{x}) = \Theta_0 + \Theta_1 x_1 + \cdots + \Theta_m x_m$
- loss function = mean squared error

Recap of linear regression

Intepretation of Θ

- $h(\mathbf{x}) = \Theta_0 + \Theta_1 x_1 + \cdots + \Theta_m x_m$
- Θ_j gives an average change in a target value with one-unit change in feature A_j, holding other features fixed

Regularized linear regression

$$h(\mathbf{x}) = \Theta_0 + \Theta_1 x_1 + \cdots + \Theta_m x_m$$

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

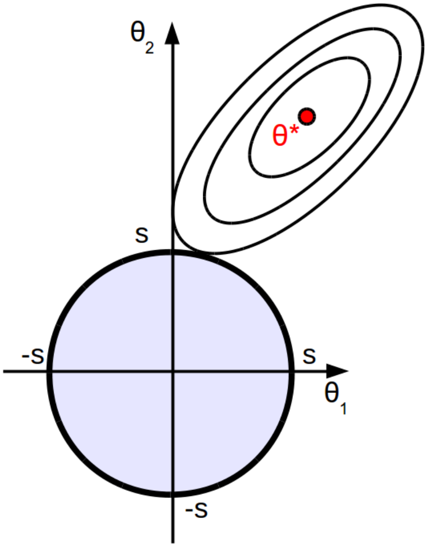
$$\boldsymbol{\Theta}^\star = \operatorname*{argmin}_{\boldsymbol{\Theta}} \sum_{i=1}^{n} (h(\mathbf{x}_i) - y_i)^2 + \lambda * \mathrm{penalty}(\boldsymbol{\Theta})$$

Ridge regression – alternative formulation

$$\boldsymbol{\Theta}^\star = \operatorname*{argmin}_{\boldsymbol{\Theta}} \sum_{i=1}^{n} (h(\mathbf{x}_i) - y_i)^2$$

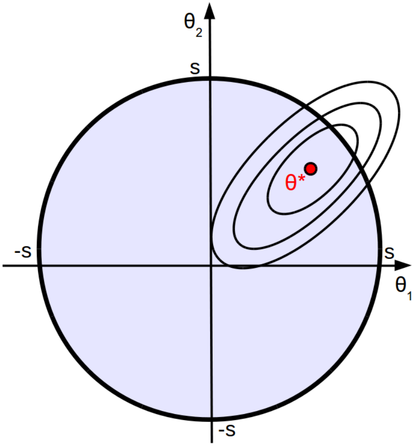
subject to $\Theta_1^2 + \cdots + \Theta_m^2 \leq s$

- the gray circle represents the feasible region for Ridge regression; the contours represent different loss values for the unconstrained model



Ridge regression – alternative formulation

- If *s* is large enough so that the minimum loss value falls into the region of **ridge regression** parameter estimates then the alternative formulation yields the primary solution.

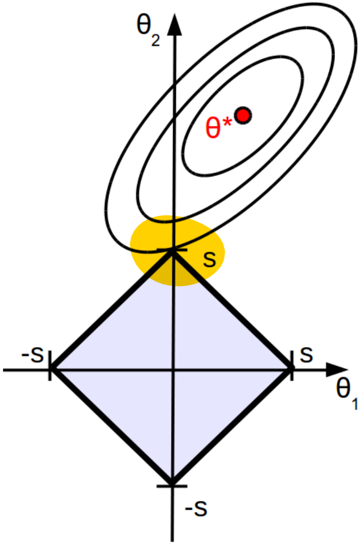


Lasso – alternative formulation

$$\boldsymbol{\Theta}^\star = \operatorname*{argmin}_{\boldsymbol{\Theta}} \sum_{i=1}^{n} (h(\mathbf{x}_i) - y_i)^2$$

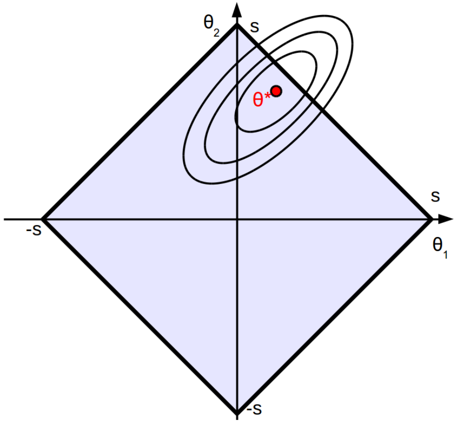
subject to $|\Theta_1| + \cdots + |\Theta_m| \leq s$

- the grey square represents the feasible region of the Lasso; the contours represent different loss values for the unconstrained model
- the feasible point that minimizes the loss is more likely to happen on the coordinates on the Lasso graph than on the Ridge regression graph since the Lasso graph is more angular



Lasso – alternative formulation

- If s is large enough so that the minimum loss value falls into the region of **loss** parameter estimates then the alternative formulation yields the primary solution.



Regularization

$$\boldsymbol{\Theta}^{\star} = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}}\ \mathrm{loss}(\boldsymbol{\Theta}) + \lambda * \textbf{penalty}(\boldsymbol{\Theta})$$

where $\lambda \geq 0$ is a **tuning parameter**

Recap of logistic regression

Logistic regression is a classification algorithm

Assume $Y = \{0, 1\}$

- **modeling the probability** $h(\mathbf{x}) = \Pr(Y = 1|\mathbf{x}; \mathbf{\Theta})$

$$h(\mathbf{x}) = g(\mathbf{\Theta}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{\Theta}^T\mathbf{x}}} \text{ , where } \mathbf{\Theta} = \langle \Theta_0, \dots, \Theta_m \rangle$$

- **prediction function** of $\mathbf{x}$

$$= \begin{cases} 1 \ \mathit{if} & h(\mathbf{x}) \geq 0.5 \\ 0 \ \mathit{if} & h(\mathbf{x}) < 0.5 \end{cases}$$

Recap of logistic regression

- $\frac{h(\mathbf{x})}{1 - h(\mathbf{x})}$ = odds ratio
- **log odds is linear**

$$\log \frac{h(\mathbf{x})}{1 - h(\mathbf{x})} = \mathbf{\Theta}^T \mathbf{x}$$

- **recall linear regression**

$$h(\mathbf{x}) = \mathbf{\Theta}^T \mathbf{x}$$

Recap of logistic regression

Interpretation of $\boldsymbol{\Theta}$

Suppose $\boldsymbol{\Theta} =< \Theta_0, \Theta_1 >$

- linear regression $h(\mathbf{x}) = \Theta_0 + \Theta_1 x_1$: Θ_1 gives an average change in a target value with one-unit change in A_1
- logistic regression $\log \frac{h(\mathbf{x})}{1-h(\mathbf{x})} = \Theta_0 + \Theta_1 x_1$: Θ_1 gives an average change in logit $h(\mathbf{x})$ with one-unit change in A_1

Recap of logistic regression

Estimating $\boldsymbol{\Theta}$ by **maximizing the likelihood**

- **loss function**

$$\begin{aligned} L(\boldsymbol{\Theta}) &= \ell(y_1, \dots, y_n; \boldsymbol{\Theta}, \mathbf{X}) \\ &= \sum_{i=1}^{n} y_i \log \mathrm{P}(y_i|\mathbf{x_i}; \boldsymbol{\Theta}) + (1 - y_i) \log(1 - \mathrm{P}(y_i|\mathbf{x_i}; \boldsymbol{\Theta})) \end{aligned}$$

- **optimization task**

$$\begin{aligned} \boldsymbol{\Theta}^\star &= \mathrm{argmax}_{\boldsymbol{\Theta}}\ L(\boldsymbol{\Theta}) \\ &= \mathrm{argmin}_{\boldsymbol{\Theta}}\ - L(\boldsymbol{\Theta}) \\ &= \mathrm{argmin}_{\boldsymbol{\Theta}}\ \sum_{i=1}^{n} -y_i \log \mathrm{P}(y_i|\mathbf{x_i}; \boldsymbol{\Theta}) - (1 - y_i) \log(1 - \mathrm{P}(y_i|\mathbf{x_i}; \boldsymbol{\Theta})) \end{aligned}$$

Recap of logistic regression

Multinomial logistic regression $Y = \{y_1, \dots, y_k\}$

- train k one-versus-all binary classifiers $h_i^\star, i = 1, \dots, k$
- classify $\mathbf{x}$ into the class K that maximizes $h_K^\star(\mathbf{x})$

Regularized logistic regression

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

$$\mathbf{\Theta}^\star = \underset{\mathbf{\Theta}}{\mathrm{argmin}} \; -\ell(y_1, \ldots, y_n; \mathbf{\Theta}, \mathbf{X}) + \lambda * \mathrm{penalty}(\mathbf{\Theta})$$

SVM and Logistic regression

Logistic regression with Ridge regression

$$L(\mathbf{\Theta}) = -[\sum_{i=1}^{n} y_i \log(h(\mathbf{x}_i)) + (1 - y_i) \log(1 - h(\mathbf{x}_i))] + \lambda \sum_{j=1}^{m} \Theta_j^2$$

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

SVM and Logistic regression

Logistic regression with Ridge regression

$$J(\boldsymbol{\Theta}) = -[\sum_{i=1}^{n} y_i \log(h(\mathbf{x}_i)) + (1-y_i)\log(1-h(\mathbf{x}_i))] + \lambda \sum_{j=1}^{m} \Theta_j^2 =$$

$$= \sum_{i=1}^{n} y_i(-\log(h(\mathbf{x}_i))) + (1-y_i)(-\log(1-h(\mathbf{x}_i))) + \lambda \sum_{j=1}^{m} \Theta_j^2 =$$

$$= \sum_{i=1}^{n} y_i L_1(\boldsymbol{\Theta}) + (1-y_i) L_0(\boldsymbol{\Theta}) + \lambda \sum_{j=1}^{m} \Theta_j^2$$

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

$$\mathrm{argmin}_{\boldsymbol{\Theta}} L(\boldsymbol{\Theta}) = \mathrm{argmin}_{\boldsymbol{\Theta}} \sum_{j=1}^{m} \Theta_j^2 + C[\sum_{i=1}^{n} y_i L_1(\boldsymbol{\Theta}) + (1-y_i) L_0(\boldsymbol{\Theta})]$$

where $L_1(\boldsymbol{\Theta}) = -\log \frac{1}{1+e^{-\Theta^T \mathbf{x}}}$ and $L_0(\boldsymbol{\Theta}) = -\log(1 - \frac{1}{1+e^{-\Theta^T \mathbf{x}}})$

SVM and Logistic regression

- **Logistic regression**

$$\mathrm{argmin}_{\boldsymbol{\Theta}} \sum_{j=1}^{m} \Theta_j^2 + C \sum_{i=1}^{n} \log(1 + e^{-\overline{y_i}\boldsymbol{\Theta}^T \mathbf{x}_i})$$

where

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

- **SVM**

$$\mathrm{argmin}_{\boldsymbol{\Theta}} \sum_{j=1}^{m} \Theta_j^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i \boldsymbol{\Theta}^T \mathbf{x}_i)$$

Soft-margin is equivalent to the regularization problem

SVM and Logistic regression

Hinge loss: $\max(0, 1 - y_i\mathbf{\Theta}^T\mathbf{x})$

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

The Hinge loss is a convex function.

SVM and Logistic regression

- $\xi_i \geq 0$ is equivalent to $\xi_i = \max(0, 1 - y_i \mathbf{\Theta}^T \mathbf{x}_i)$
- $\mathrm{argmin}_{\mathbf{\Theta}}\ J(\mathbf{\Theta}) = \mathrm{argmin}_{\mathbf{\Theta}} C \sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{\Theta}^T \mathbf{x}_i) + \sum_{j=1}^{m} \Theta_j^2 =$

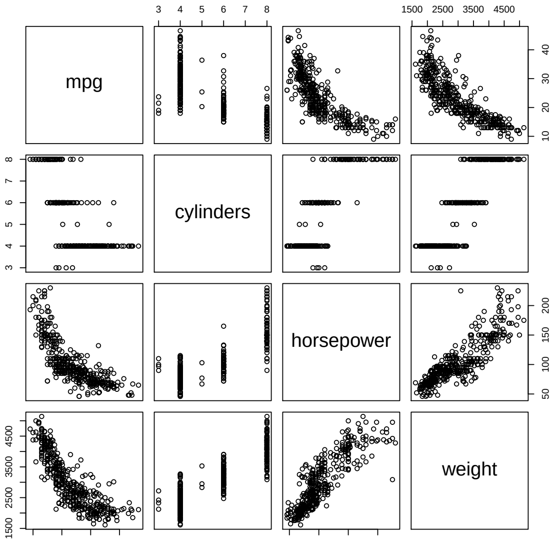
$$= \mathrm{argmin}_{\mathbf{\Theta}} C \sum_{i=1}^{n} \xi_i + \sum_{j=1}^{m} \Theta_j^2$$

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

Principal Component Analysis (PCA)

- a tool to analyze the data
- a tool to do dimensionality reduction

Auto data set



Basic concepts needed

- measures of center and spread, covariance and correlation from data analysis
- eigenvectors, eigenvalues, dot product, basis from linear algebra

Data analysis

How two features are related

Both covariance and correlation indicate how closely two features relationship follows a straight line.

- **Covariance** measures the degree of the linear relationship between two features

$$\mathrm{cov}(X, Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

  - > 0 both features increase or decrease together
  - < 0 while one feature increases the other decreases
  - $= 0$ features are independent of each other

Data analysis

- **Covariance matrix** of features $\mathrm{A}_1, \dots, \mathrm{A}_m$ represents covariance among them

$$\mathrm{COV}(\mathrm{A}_1, \dots, \mathrm{A}_m) = \begin{pmatrix} \mathrm{var}(\mathrm{A}_1) & \mathrm{cov}(\mathrm{A}_1, \mathrm{A}_2) & \dots & \mathrm{cov}(\mathrm{A}_1, \mathrm{A}_m) \\ \mathrm{cov}(\mathrm{A}_2, \mathrm{A}_1) & \mathrm{var}(\mathrm{A}_2) & \dots & \mathrm{cov}(\mathrm{A}_2, \mathrm{A}_m) \\ \dots & \dots & \dots & \dots \\ \mathrm{cov}(\mathrm{A}_m, \mathrm{A}_1) & \mathrm{cov}(\mathrm{A}_m, \mathrm{A}_2) & \dots & \mathrm{var}(\mathrm{A}_m) \end{pmatrix}$$

Data analysis

How two features are related

- **Correlation** measures the degree to which the features tend to move together.

$$-1 \leq \mathrm{cor}(X, Y) = \frac{\mathrm{cov}(X, Y)}{s_X s_Y} \leq 1$$

Data analysis
Auto data set

```
> cov(Auto[c("mpg", "cylinders", "horsepower", "weight")])

#                   mpg   cylinders  horsepower     weight
# mpg          60.91814  -10.352928  -233.85793  -5517.441
# cylinders   -10.35293    2.909696    55.34824   1300.424
# horsepower -233.85793   55.348244  1481.56939  28265.620
# weight    -5517.44070 1300.424363 28265.62023 721484.709

> cor(Auto[c("mpg", "cylinders", "horsepower", "weight")])

#                   mpg  cylinders horsepower     weight
# mpg         1.0000000 -0.7776175 -0.7784268 -0.8322442
# cylinders  -0.7776175  1.0000000  0.8429834  0.8975273
# horsepower -0.7784268  0.8429834  1.0000000  0.8645377
# weight     -0.8322442  0.8975273  0.8645377  1.0000000
```

Linear algebra

- **Eigenvector $\mathbf{u}$, eigenvalue λ: $\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$**
  - $\mathbf{u}$ does not change its direction under the transformation
  - $\lambda\mathbf{u}$ scales a vector $\mathbf{u}$ by λ; it changes its length, not its direction

1. The covariance matrix of an $n \times m$ matrix $\mathbf{X}$ is an $m \times m$ symmetric matrix given by $\frac{1}{n-1}\mathbf{X}\mathbf{X}^{\mathrm{T}}$
2. Any symmetric matrix $m \times m$ has a set of orthonormal eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$ and associated eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$
   - for any i, $\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i$
   - $||\mathbf{v}_i|| = 1$
   - $\mathbf{v}_i \mathbf{v}_j = 0$ if $i \neq j$
3. $\mathbf{A}$ is a symmetric $m \times m$ matrix and $\mathbf{E}$ is an $m \times m$ matrix whose i-th column is the i-th eigenvector of $\mathbf{A}$. The eigenvectors are ordered in terms of decreasing values of their associated eigenvalues. Then there is a diagonal matrix $\mathbf{D}$ such that $\mathbf{A} = \mathbf{E}\mathbf{D}\mathbf{E}^{\mathrm{T}}$
4. If the rows of $\mathbf{E}$ are orthogonal, then $\mathbf{E}^{-1} = \mathbf{E}^{\mathrm{T}}$

Linear algebra

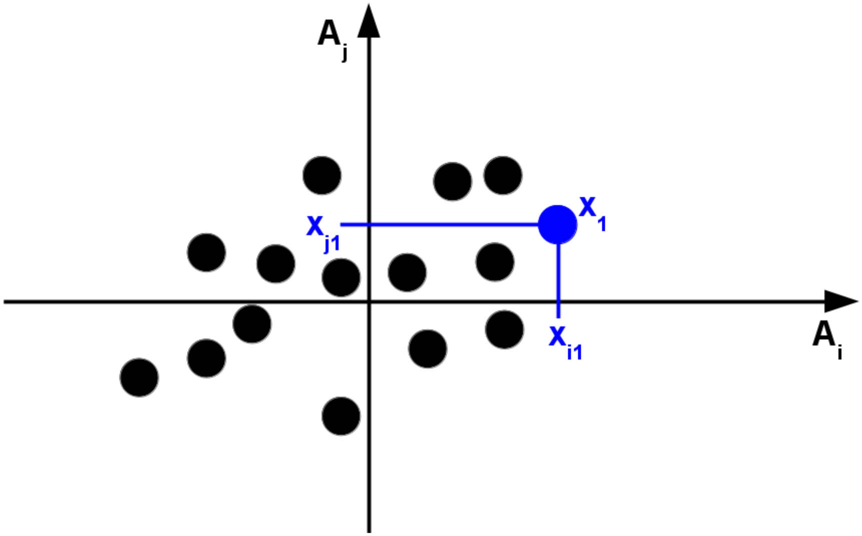
- **Dot product** of $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{R}^m$: $\mathbf{x}_1\mathbf{x}_2 = \sum_{i=1}^{m} x_{1_i} x_{2_i}$
- **Basis** of $\mathcal{R}^m$ is a set of linearly independent vectors $\mathbf{u}_1, \dots, \mathbf{u}_m$:
  - none of them is a linear combination of other vectors
  - $\mathbf{u}_i\mathbf{u}_j = 0$, $i, j = 1, \dots m$, $i \neq j$
  - any $\mathbf{u} = c_1\mathbf{u}_1 + \dots + c_m\mathbf{u}_m$
  - for example, the standard basis of the *3*-dimensional Euclidean space $\mathcal{R}^3$ consists of $\mathbf{x} = (1, 0, 0), \mathbf{y} = (0, 1, 0), \mathbf{z} = (0, 0, 1)$. It is an example of orthonormal basis, so called *naive* basis $\mathbf{I}$

Principal Component Analysis

- **instances** $Data = \{\mathbf{x}_i; \mathbf{x}_i \in \mathcal{R}^m\}, |Data| = n$
- **features** $Attr = \{\mathrm{A}_1, \ldots, \mathrm{A}_m\}$
- **representation of** *Data* **for derivation**: $m \times n$ matrix $\mathbf{X}$

$$\mathbf{X} = \begin{pmatrix} x_{11} & \ldots & x_{1n} \\ x_{21} & \ldots & x_{2n} \\ \ldots & \ldots & \ldots \\ x_{m1} & \ldots & x_{mn} \end{pmatrix}$$

PCA



PCA

Which features to keep?

- features that change a lot, i.e. high variance
- features that do not depend on others, i.e. low covariance

Which features to ignore?

- features with some noise, i.e. low variance

PCA

PCA principles

1. high correlation $\sim$ high redundancy
2. the most important feature has the largest variance

PCA

- **Question**
  Is there any other representation of **X** to extract the most important features?
- **Answer**
  Another basis

$$\mathbf{P}^{\mathrm{T}}\mathbf{X} = \mathbf{Z}$$

where **P** transforms **X** into **Z**

PCA

Heading for the P matrix

$$\mathbf{P} = \begin{pmatrix} \mathbf{p}_{11} & \dots & \dots & \mathbf{p}_{1m} \\ \mathbf{p}_{21} & \dots & \dots & \mathbf{p}_{2m} \\ \dots & \dots & \dots & \dots \\ \mathbf{p}_{m1} & \dots & \dots & \mathbf{p}_{mm} \end{pmatrix}$$

- **Principal components** of $\mathbf{X}$ are the vectors $\mathbf{p}_i = \langle p_{1i}, \dots, p_{mi} \rangle$
- **Principal component loadings** of $\mathbf{p}_i$ are the elements $p_{i1}, \dots, p_{im}$

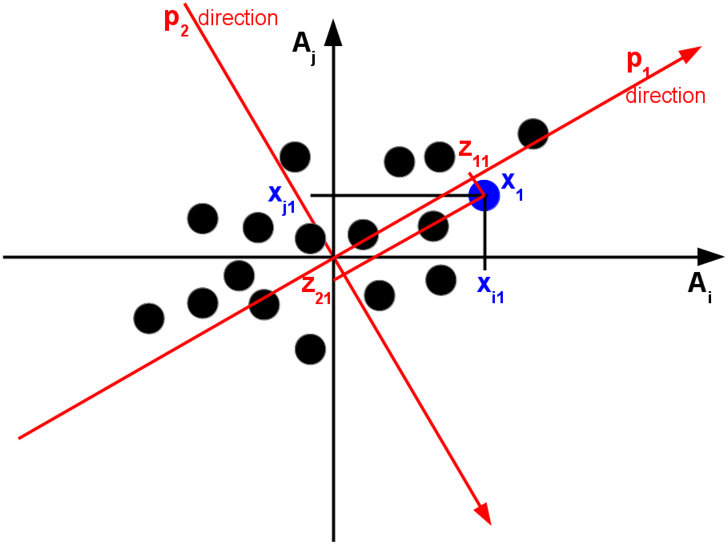
PCA

Heading for P

$$\mathbf{Z} = \begin{pmatrix} \mathbf{p}_1\mathbf{x}_1 & \dots & \dots & \mathbf{p}_1\mathbf{x}_n \\ \mathbf{p}_2\mathbf{x}_1 & \dots & \dots & \mathbf{p}_2\mathbf{x}_n \\ \dots & \dots & \dots & \dots \\ \mathbf{p}_m\mathbf{x}_1 & \dots & \dots & \mathbf{p}_m\mathbf{x}_n \end{pmatrix}$$

***i*-principal component scores** of n instances are $\mathbf{p}_i\mathbf{x}_1, \mathbf{p}_i\mathbf{x}_2, \dots, \mathbf{p}_i\mathbf{x}_n$.

PCA



PCA

Heading for P

- What is a good choice of $\mathbf{P}$?
- What features we would like $\mathbf{Z}$ to exhibit?

Goal: $\mathbf{Z}$ is a new representation of $\mathbf{X}$: the new features are linear combinations of the original features whose weights are given by $\mathbf{P}$.

The covariance matrix of $\mathbf{Z}$ is diagonal and the entries on the diagonal are in descending order, i.e. the covariance of any pair of distinct features is zero, and the variance of each of our new features is listed along the diagonal.

PCA

Heading for P

- $\mathbf{p}_i\mathbf{x}_j$ is a projection of $\mathbf{x}_j$ on $\mathbf{p}_i$
- principal components are new basis vectors to represent $\mathbf{x}_j$, $j = 1, \ldots, n$
- changing the basis does not change data, it changes their representation

PCA

The covariance matrix $\mathrm{cov}(\mathrm{A}_1, \mathrm{A}_2, \ldots, \mathrm{A}_\mathrm{m})$:

- on the diagonal, large values correspond to interesting structure
- off the diagonal, large values correspond to high redundancy

Derivation of PCA

1. preprocessing *Data*: mean normalization to get centered data $\rightarrow \mathbf{X}$
2. $\mathrm{cov}(\mathbf{X}) = \mathbf{A} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^{\mathrm{T}}$
3. Compute eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_m$ and eigenvalues $\lambda_1, \dots, \lambda_m$ of $\mathbf{A}$
4. Take the eigenvectors, order them by eigenvalues, i.e. by significance, highest to lowest: $\mathbf{p}_1, \dots, \mathbf{p}_m, \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_m$
5. The principal components $\mathbf{p}_1, \dots, \mathbf{p}_m$ become columns of $\mathbf{P}$

$$\boldsymbol{p}_i = \begin{pmatrix} p_{1i} \\ \dots \\ p_{mi} \end{pmatrix}$$

Properties of PCA

$$\mathbf{P}^{\mathrm{T}}\mathbf{X} = \mathbf{Z}$$

The i-th diagonal value of $\mathrm{cov}(\mathbf{Z})$ is the variance of $\mathbf{X}$ along $\mathbf{p_i}$.

$$\mathbf{Z} = \begin{pmatrix} \mathbf{p}_1\mathbf{x}_1 & \dots & \dots & \mathbf{p}_1\mathbf{x}_n \\ \mathbf{p}_2\mathbf{x}_1 & \dots & \dots & \mathbf{p}_2\mathbf{x}_n \\ \dots & \dots & \dots & \dots \\ \mathbf{p}_m\mathbf{x}_1 & \dots & \dots & \mathbf{p}_m\mathbf{x}_n \end{pmatrix}$$

We calculate a rotation of the original coordinate system such that all non-diagonal elements of the new covariance matrix become zero. The eigenvectors (principal components) define the basis of the new coordinate axes and the eigenvalues correspond to the diagonal elements of the new covariance matrix. So the eigenvalues, by definition, define the variance along the corresponding principal components.

Properties of PCA

$$cov(\mathbf{P}^{\mathrm{T}}\mathbf{X}) \overset{\text{see } \underline{\text{p.}}47.1}{=} \frac{1}{n-1}(\mathbf{P}^{\mathrm{T}}\mathbf{X})(\mathbf{P}^{\mathrm{T}}\mathbf{X})^{\mathrm{T}} =$$

$$\frac{1}{n-1}\mathbf{P}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{P} \overset{\text{let } \mathbf{A}=\mathbf{X}\mathbf{X}^{T}}{=} \frac{1}{n-1}\mathbf{P}^{\mathrm{T}}\mathbf{A}\mathbf{P} =$$

$$\overset{\text{see } \underline{\text{p.}}47.3}{=} \frac{1}{n-1}\mathbf{P}^{\mathrm{T}}(\mathbf{P}\mathbf{D}\mathbf{P}^{\mathrm{T}})\mathbf{P} \overset{\text{see } \underline{\text{p.}}47.4}{=} \frac{1}{n-1}\mathbf{P}^{T}(\mathbf{P}^{T})^{-1}\mathbf{D}\mathbf{P}^{T}(\mathbf{P}^{T})^{-1} = \frac{1}{n-1}\mathbf{D}$$

Properties of PCA

A geometric interpretation for the first principal component $\mathbf{p}_1$

It defines a direction in feature space along which the data vary the most. If we project the n instances $\mathbf{x}_1, \dots, \mathbf{x}_n$ onto this direction, the projected values are the principal component scores $z_{11}, \dots, z_{n1}$ themselves.

Proportion of Variance Explained (PVE)

How much of the information in a given data set is lost by projecting the instances onto the first few principal components?

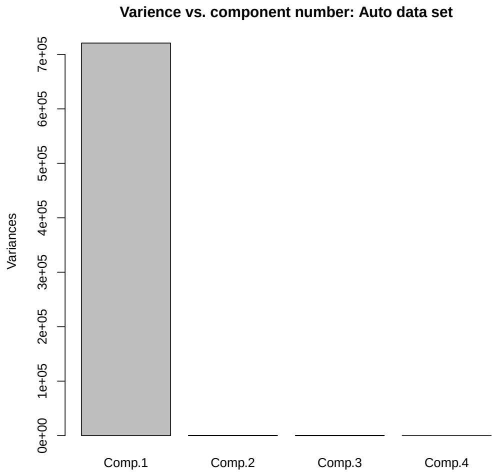
In other words, how much of the variance in the data is not contained in the first few principal components?

- total variance in $\mathbf{X}$: $\sum_{j=1}^{m} \mathrm{var}(A_j) = \sum_{i=1}^{m} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$ (assuming feature normalization)
- variance expressed by $\mathbf{p}_k$: $\frac{1}{n} \sum_{i=1}^{n} z_{ki}^2$
- $\mathrm{PVE}(\mathbf{p}_k) = \frac{\sum_{i=1}^{n} z_{ki}^2}{\sum_{i=1}^{m} \sum_{i=1}^{n} x_{ij}^2}$
- $\mathrm{PVE}(\mathbf{p}_1, \ldots, \mathbf{p}_M) = \sum_{i=1}^{M} \mathrm{PVE}(\mathbf{p}_i)$, $M \leq m$

PCA
Auto data set

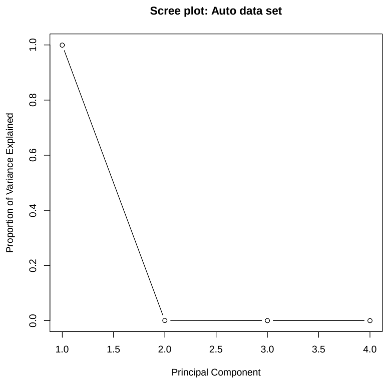
```
> summary(pca.auto)

# Importance of components:
#                           Comp.1    Comp.2   Comp.3   Comp.4
# Standard deviation     848.9957204 1.93e+01 4.21e+00 7.14e-01
# Proportion of Variance   0.9994566 5.18e-04 2.46e-05 7.07e-07
# Cumulative Proportion    0.9994566 9.99e-01 9.99e-01 1.00e+00

> pca.auto$loadings

# Loadings:
#            Comp.1 Comp.2 Comp.3 Comp.4
# mpg                       0.999
# cylinders                        -1.000
# horsepower        -0.998
# weight     -0.999
```

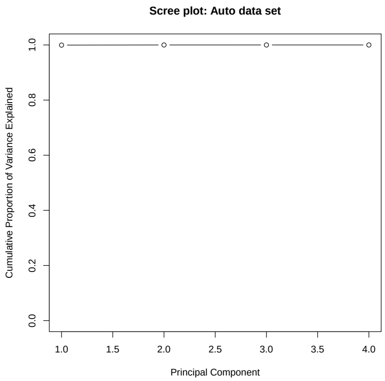
PCA
Auto data set



PCA
Auto data set

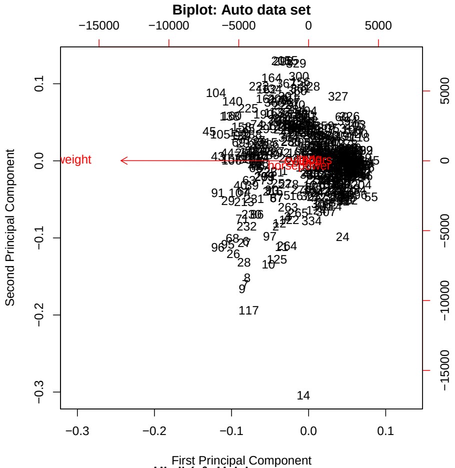
Scree plot





PCA
Auto data set

Biplot displays both the principal component scores and the principal component loadings.



PCA
Auto data set

Comment on biplot

The first two principal components for the Auto data set. The black numbers represent the scores for the first two principal components. The red arrows indicate the first two principal component loading vectors (with axes on the bottom and left). For example, the loading for `weight` on the first component is -0.999, and its loading on the second principal component 0 (the word `weight` is centered at the point (-0.999, 0)).

PCA

In general, a $m \times n$ matrix $\mathbf{X}$ has $\min(n-1, m)$ distinct principal components.

- **Question**
  How many principal components are needed?
- **Answer**
  Unfortunately, there is no single answer to this question. Study scree plots.

Generalization error estimation by bootstrapping

- Suppose a development data of n examples
- Train a model on the data
- Test the model on the data
- Get training error = optimistic error e_l
- Repeat 200 times
  - Randomly select n examples with replacement and train a model
    on average, 63.2% of the original sample
  - Test the model on the examples not used in the training
    on average, 36.8% of the orginal sample
  - Get test error
- Get mean test error = pesimistic error e_o
- **generalization error estimation =** $0.368 * e_l + 0.632 * e_o$