Outline

- Overfitting
- Regularization
  - Ridge regression
  - Lasso
  - Linear regression
  - Logistic regression
  - SVM
- Principal Component Analysis
- Generalization error estimation by bootstrapping
• Suppose $m$ features $A_1, \ldots, A_m$ and a set of possible target values $Y$
• Suppose development data as a set of instances

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

where $x_i$ are feature vectors and $y_i$ are the corresponding target values

Let $h^*$ be a best approximation of $c$ trained on $Data$. 
Model complexity is the number of hypothesis parameters

$$\Theta = \langle \Theta_0, \ldots, \Theta_m \rangle$$
Model complexity – example
Model complexity – example

- \( h(x) \): a straight line – determined by *two* parameters of the prediction function
  – doesn’t fit two examples

- \( h_2(x) \): a parabola – determined by *three* parameters of the prediction function
  – doesn’t fit one example

- \( h_3(x) \): 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

low variance

low bias

high variance

high bias
Generalization error $\text{error}_\mathcal{D}(h)$ 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**: $\text{error}_\mathcal{D}(h) = E (\hat{y}_i - y_i)^2$
- for **classification**: $\text{error}_\mathcal{D}(h) = \Pr (\hat{y}_i \neq y_i)$

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

$$\text{error}_\mathcal{D}(h) = \text{Bias}^2 + \text{Variance}$$
Bias and variance

- underfitting = high bias
- overfitting = high variance
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 \( \Theta^* \) that minimizes loss function for the data \( D \)
  \[ \Theta^* = \arg\min_{\Theta} \text{loss}(\Theta) \]

- Regularization
  \[ \Theta^* = \arg\min_{\Theta} \text{loss}(\Theta) + \lambda \times \text{penalty}(\Theta) \]
  where \( \lambda \geq 0 \) is a tuning parameter
Regularization – Ridge regression

\[
\text{penalty}(\Theta) = \Theta_1^2 + \cdots + \Theta_m^2
\]

\(\Theta_1^2 + \cdots + \Theta_m^2\) is the \(l_2\) norm

\[
\Theta^* = \arg\min_{\Theta} \text{loss}(\Theta) + \lambda (\Theta_1^2 + \cdots + \Theta_m^2)
\]

The penalty is applied to \(\Theta_1, \ldots, \Theta_m\), but not to \(\Theta_0\), since the goal is to regularize the estimated association between each feature and the target value.
Ridge regression

$\Theta^* = \arg\min_{\Theta} \text{loss}(\Theta) + \lambda \star (\Theta_1^2 + \cdots + \Theta_m^2)$

- **Let** $\Theta_{\lambda_1}^*, \ldots, \Theta_{\lambda_m}^*$ be ridge regression parameter estimates for a particular value of $\lambda$

- **Let** $\Theta_1^*, \ldots, \Theta_m^*$ be unregularized parameter estimates

- $0 \leq \frac{\Theta_{\lambda_1}^2 + \cdots + \Theta_{\lambda_m}^2}{\Theta_1^2 + \cdots + \Theta_m^2} \leq 1$

- **When** $\lambda = 0$, then $\Theta_{\lambda}^* = \Theta_i^*$ for $i = 1, \ldots, m$

- **When** $\lambda$ is extremely large, then $\Theta_{\lambda}^* = 0$ for $i = 1, \ldots, m$

- **When** $\lambda$ between, we are fitting a model and skrinking the parameters
Ridge regression

![Graph showing the relationship between Log Lambda and Coefficients. The x-axis represents Log Lambda, ranging from -4 to 4, and the y-axis represents Coefficients, ranging from -4 to 4. The graph contains multiple lines, each representing a different coefficient value. The lines are color-coded and show how the coefficients change as Log Lambda varies.]
Regularization – Lasso

\[
\text{penalty}(\Theta) = |\Theta_1| + \cdots + |\Theta_m|
\]

\[|\Theta_1| + \cdots + |\Theta_m| \text{ is the } \ell_1 \text{ norm}\]

\[
\Theta^* = \arg\min_{\Theta} \text{loss}(\Theta) + \lambda * (|\Theta_1| + \cdots + |\Theta_m|)
\]
Lasso

\[ \Theta^* = \operatorname{argmin}_{\Theta} \text{loss}(\Theta) + \lambda \ast (|\Theta_1| + \cdots + |\Theta_m|) \]

- Let \( \Theta^*_\lambda_1, \ldots, \Theta^*_\lambda_m \) be lasso regression parameter estimates
- Let \( \Theta^*_1, \ldots, \Theta^*_m \) be unregularized parameter estimates
- When \( \lambda = 0 \), then \( \Theta^*_{\lambda_i} = \Theta^*_i \) for \( i = 1, \ldots, m \)
- When \( \lambda \) grows, then the impact of penalty grows
- When \( \lambda \) is extremely large, then \( \Theta^*_{\lambda_i} = 0 \) for \( i = 1, \ldots, m \)
Lasso

Coefficients vs Log Lambda

Log Lambda

Coefficients

-10 -8 -6 -4 -2

-60 -40 -20 0 20 40

98 96 91 58 4 1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 17 18 19 21 22 23 24 25 26 27 28 29 31 32 34 36 37 38 39 40 42 43 44 45 46 47 48 49 50 51 52 53 54 56 57 58 59 60 61 63 65 66 67 68 70 73 74 75 76 77 78 80 81 82 83 87 91 94 95 96 98 100 101 102 103 105 106 107 108 109 110 112 113 114 115 116 118 120 121 122 123 127 128 129 130 133 134 135 136 139 149 150 151 152 153 154 156 157 160 161 162 163 165 168

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

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

**Linear regression** is a regression algorithm

\[ \Theta^* = \arg\min_{\Theta} \sum_{i=1}^{n} (h(x_i) - y_i)^2 \]

where

- \( h(x) = \Theta_0 + \Theta_1 x_1 + \cdots + \Theta_m x_m \)
- loss function = mean squared error
Recap of linear regression

Interpretation of $\Theta$

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

\[ h(x) = \Theta_0 + \Theta_1 x_1 + \cdots + \Theta_m x_m \]

\[ \text{loss}(\Theta) = \text{RSS} = \sum_{i=1}^{n} (h(x_i) - y_i)^2 \]

\[ \Theta^* = \arg\min_\Theta \sum_{i=1}^{n} (h(x_i) - y_i)^2 + \lambda \times \text{penalty}(\Theta) \]
Ridge regression – alternative formulation

\[ \Theta^* = \arg\min_{\Theta} \sum_{i=1}^{n} (h(x_i) - y_i)^2 \]

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.
• 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

\[ \Theta^* = \arg\min_{\Theta} \sum_{i=1}^{n} (h(x_i) - y_i)^2 \]

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
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.
\[ \Theta^* = \arg\min_{\Theta} \text{loss}(\Theta) + \lambda \times \text{penalty}(\Theta) \]

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** $h(x) = \Pr(Y = 1|x; \Theta)$

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

  where $\Theta = \langle \Theta_0, \ldots, \Theta_m \rangle$

- **prediction function** of $x$

  $$= \begin{cases} 1 & \text{if } h(x) \geq 0.5 \\ 0 & \text{if } h(x) < 0.5 \end{cases}$$
Recap of logistic regression

- \[ \frac{h(x)}{1 - h(x)} = \text{odds ratio} \]
- \[ \log \frac{h(x)}{1 - h(x)} = \Theta^T x \]
- recall linear regression

\[ h(x) = \Theta^T x \]
Recap of logistic regression

Interpretation of $\Theta$

Suppose $\Theta = \langle \Theta_0, \Theta_1 \rangle$

- linear regression $h(x) = \Theta_0 + \Theta_1 x_1$: $\Theta_1$ gives an average change in a target value with one-unit change in $A_1$

- logistic regression $\log \frac{h(x)}{1-h(x)} = \Theta_0 + \Theta_1 x_1$: $\Theta_1$ gives an average change in logit $h(x)$ with one-unit change in $A_1$
Recap of logistic regression

Estimating $\Theta$ by maximizing the likelihood

- **loss function**

$$L(\Theta) = \ell(y_1, \ldots, y_n; \Theta, X)$$

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

- **optimization task**

$$\Theta^* = \arg\max_{\Theta} L(\Theta)$$

$$= \arg\min_{\Theta} -L(\Theta)$$

$$= \arg\min_{\Theta} \sum_{i=1}^{n} -y_i \log P(y_i|x_i; \Theta) - (1 - y_i) \log (1 - P(y_i|x_i; \Theta))$$
Recap of logistic regression

Multinomial logistic regression \( Y = \{y_1, \ldots, y_k\} \)

- train \( k \) one-versus-all binary classifiers \( h_i^*, i = 1, \ldots, k \)
- classify \( x \) into the class \( K \) that maximizes \( h_K^*(x) \)
Regularized logistic regression

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

\[ \Theta^* = \arg\min_{\Theta} -\ell(y_1, \ldots, y_n; \Theta, X) + \lambda \times \text{penalty}(\Theta) \]
Logistic regression with Ridge regression

\[ L(\Theta) = -\left[ \sum_{i=1}^{n} y_i \log(h(x_i)) + (1 - y_i) \log(1 - h(x_i)) \right] + \lambda \sum_{j=1}^{m} \Theta_j^2 \]

\[ \Theta^* = \arg\min_{\Theta} L(\Theta) \]
Logistic regression with Ridge regression

\[
L(\Theta) = -\left[\sum_{i=1}^{n} y_i \log(h(x_i)) + (1 - y_i) \log(1 - h(x_i))\right] + \lambda \sum_{j=1}^{m} \Theta_j^2 = \\
= \sum_{i=1}^{n} y_i (-\log(h(x_i))) + (1 - y_i)(-\log(1 - h(x_i))) + \lambda \sum_{j=1}^{m} \Theta_j^2 = \\
= \sum_{i=1}^{n} y_i L_1(\Theta) + (1 - y_i)L_0(\Theta) + \lambda \sum_{j=1}^{m} \Theta_j^2
\]

\[
A + \lambda B \equiv CA + B, \ C = \frac{1}{\lambda}
\]

\[
\arg\min_{\Theta} L(\Theta) = \arg\min_{\Theta} \sum_{j=1}^{m} \Theta_j^2 + C \left[ \sum_{i=1}^{n} y_i L_1(\Theta) + (1 - y_i)L_0(\Theta) \right]
\]

where \( L_1(\Theta) = -\log \frac{1}{1+e^{-\Theta^T x}} \) and \( L_0(\Theta) = -\log(1 - \frac{1}{1+e^{-\Theta^T x}}) \)
• **Logistic regression**

\[
\arg\min_{\Theta} \sum_{j=1}^{m} \Theta_j^2 + C \sum_{i=1}^{n} \log(1 + e^{-\bar{y}_i\Theta^T x_i})
\]

where

\[
\bar{y}_i = \begin{cases} 
-1 & \text{if } y_i = 0 \\
1 & \text{if } y_i = 1 
\end{cases}
\]

• **SVM**

\[
\arg\min_{\Theta} \sum_{j=1}^{m} \Theta_j^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i\Theta^T x_i)
\]

Soft-margin is equivalent to the regularization problem
**Hinge loss**: \( \max(0, 1 - y_i \Theta^T x) \)

1. \( y_i \Theta^T x_i > 1 \): no contribution to loss
2. \( y_i \Theta^T x_i = 1 \): no contribution to loss
3. \( y_i \Theta^T x_i < 1 \): contribution to loss

The Hinge loss is a convex function.
\( \xi_i \geq 0 \) is equivalent to \( \xi_i = \max(0, 1 - y_i \Theta^T x_i) \)

\[
\text{argmin}_\Theta L(\Theta) = \text{argmin}_\Theta C \sum_{i=1}^{n} \max(0, 1 - y_i \Theta^T x_i) + \sum_{j=1}^{m} \Theta_j^2 =
\]

\[
= \text{argmin}_\Theta C \sum_{i=1}^{n} \xi_i + \sum_{j=1}^{m} \Theta_j^2
\]

s.t. \( \Theta^T x_i \geq 1 - \xi_i \) if \( y_i = 1 \) and \( \Theta^T x_i \leq -1 + \xi_i \) 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
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

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

- $> 0$ 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 $A_1, \ldots, A_m$ represents covariance among them

$$\text{COV}(A_1, \ldots, A_m) = \begin{pmatrix}
\text{var}(A_1) & \text{cov}(A_1, A_2) & \ldots & \text{cov}(A_1, A_m) \\
\text{cov}(A_2, A_1) & \text{var}(A_2) & \ldots & \text{cov}(A_2, A_m) \\
\ldots & \ldots & \ldots & \ldots \\
\text{cov}(A_m, A_1) & \text{cov}(A_m, A_2) & \ldots & \text{var}(A_m)
\end{pmatrix}$$
How two features are related

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

\[-1 \leq \text{cor}(X, Y) = \frac{\text{cov}(X, Y)}{s_X s_Y} \leq 1\]
Data analysis
Auto data set

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

#       mpg cylinders horsepower weight
# mpg  60.91814  -10.35293  -233.85793  -5517.441
# cylinders  -10.35293   2.909696   55.348241   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
```
• **Eigenvector** $u$, **eigenvalue** $\lambda$: $Au = \lambda u$
  • $u$ does not change its direction under the transformation
  • $\lambda u$ scales a vector $u$ by $\lambda$; it changes its length, not its direction

1. The covariance matrix of an $n \times m$ matrix $X$ is an $m \times m$ symmetric matrix given by $\frac{1}{n-1}XX^T$

2. Any symmetric matrix $m \times m$ has a set of orthonormal eigenvectors $v_1, v_2, \ldots, v_m$ and associated eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$
   • for any $i$, $Av_i = \lambda_i v_i$
   • $\|v_i\| = 1$
   • $v_i v_j = 0$ if $i \neq j$

3. $A$ is a symmetric $m \times m$ matrix and $E$ is an $m \times m$ matrix whose $i$-th column is the $i$-th eigenvector of $A$. The eigenvectors are ordered in terms of decreasing values of their associated eigenvalues. Then there is a diagonal matrix $D$ such that $A = EDE^T$

4. If the rows of $E$ are orthogonal, then $E^{-1} = E^T$
• **Dot product** of $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^m$: $\mathbf{x}_1 \cdot \mathbf{x}_2 = \sum_{i=1}^{m} x_1^i x_2^i$

• **Basis** of $\mathbb{R}^m$ is a set of linearly independent vectors $\mathbf{u}_1, \ldots, \mathbf{u}_m$:
  - none of them is a linear combination of other vectors
  - $\mathbf{u}_i \cdot \mathbf{u}_j = 0$, $i, j = 1, \ldots, m$, $i \neq j$
  - any $\mathbf{u} = c_1 \mathbf{u}_1 + \cdots + c_m \mathbf{u}_m$

  • for example, the standard basis of the 3-dimensional Euclidean space $\mathbb{R}^3$ 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 *naïve basis*.
• instances $Data = \{x_i; x_i \in \mathbb{R}^m\}, |Data| = n$
• features $Attr = \{A_1, \ldots, A_m\}$
• representation of $Data$ for derivation: $m \times n$ matrix $X$

$$X = \begin{pmatrix}
x_{11} & \cdots & x_{1n} \\
x_{21} & \cdots & x_{2n} \\
\vdots & \ddots & \vdots \\
x_{m1} & \cdots & x_{mn}
\end{pmatrix}$$
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 principles

1. high correlation $\sim$ high redundancy
2. the most important feature has the largest variance
• **Question**  
  Is there any other representation of $X$ to extract the most important features?

• **Answer**  
  Another basis

$$P^T X = Z$$

where $P$ transforms $X$ into $Z$
Heading for the P matrix

\[ P = \begin{pmatrix} p_{11} & \cdots & \cdots & p_{1m} \\ p_{21} & \cdots & \cdots & p_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ p_{m1} & \cdots & \cdots & p_{mm} \end{pmatrix} \]

- **Principal components** of \( X \) are the vectors \( p_i = \langle p_{1i}, \ldots, p_{mi} \rangle \)
- **Principal component loadings** of \( p_i \) are the elements \( p_{i1}, \ldots, p_{im} \)
Heading for $P$

$$Z = \begin{pmatrix} p_1 x_1 & \ldots & \ldots & p_1 x_n \\ p_2 x_1 & \ldots & \ldots & p_2 x_n \\ \ldots & \ldots & \ldots & \ldots \\ p_m x_1 & \ldots & \ldots & p_m x_n \end{pmatrix}$$

$i$-principal component scores of $n$ instances are $p_i x_1, p_i x_2, \ldots, p_i x_n$. 
PCA

Heading for P

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

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

The covariance matrix of $Z$ 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.
• $p_i x_j$ is a projection of $x_j$ on $p_i$

• principal components are new basis vectors to represent $x_j$, $j = 1, \ldots, n$

• changing the basis does not change data, it changes their representation
PCA

The covariance matrix \( \text{cov}(A_1, A_2, \ldots, A_m) \):

- 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 → \( \mathbf{X} \)

2. \( \text{cov}(\mathbf{X}) = \mathbf{A} = \frac{1}{n-1} \mathbf{X} \mathbf{X}^\text{T} \)

3. Compute eigenvectors \( \mathbf{v}_1, \ldots, \mathbf{v}_m \) and eigenvalues \( \lambda_1, \ldots, \lambda_m \) of \( \mathbf{A} \)

4. Take the eigenvectors, order them by eigenvalues, i.e. by significance, highest to lowest: \( \mathbf{p}_1, \ldots, \mathbf{p}_m, \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \)

5. The principal components \( \mathbf{p}_1, \ldots, \mathbf{p}_m \) become columns of \( \mathbf{P} \)

\[
\mathbf{p}_i = \begin{pmatrix} p_{1i} \\ \cdots \\ p_{mi} \end{pmatrix}
\]
Properties of PCA

\[ P^T X = Z \]

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

\[
Z = \begin{pmatrix}
  p_1 x_1 & \ldots & \ldots & p_1 x_n \\
  p_2 x_1 & \ldots & \ldots & p_2 x_n \\
  \ldots & \ldots & \ldots & \ldots \\
  p_m x_1 & \ldots & \ldots & p_m x_n
\end{pmatrix}
\]

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

\[ \text{cov}(P^T X) \stackrel{\text{see p.47.1}}{=} \frac{1}{n-1} (P^T X)(P^T X)^T = \]

\[ \frac{1}{n-1} P^T X X^T P \text{ let } A \equiv X X^T \quad \frac{1}{n-1} P^T A P = \]

\[ \text{see p.47.3} \quad \frac{1}{n-1} P^T (P D P^T) P \text{ see p.47.4} \quad \frac{1}{n-1} P^T (P^T)^{-1} D P^T (P^T)^{-1} = \frac{1}{n-1} D \]
A geometric interpretation for the first principal component $p_1$

It defines a direction in feature space along which the data vary the most. If we project the $n$ instances $x_1, \ldots, x_n$ onto this direction, the projected values are the principal component scores $z_{11}, \ldots, z_{n1}$ 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 $\mathbf{X}$: $\sum_{j=1}^{m} \text{var}(A_j) = \sum_{i=1}^{m} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$ (assuming feature normalization)
- variance expressed by $\mathbf{p}_k$: $\frac{1}{n} \sum_{i=1}^{n} z_{ki}^2$
- $\text{PVE}(\mathbf{p}_k) = \frac{\sum_{i=1}^{n} z_{ki}^2}{\sum_{i=1}^{m} \sum_{i=1}^{n} x_{ij}^2}$
- $\text{PVE}(\mathbf{p}_1, \ldots, \mathbf{p}_M) = \sum_{i=1}^{M} \text{PVE}(\mathbf{p}_i), \ M \leq m$
PCA
Auto data set

```r
> a <- Auto[,c("mpg", "cylinders", "horsepower", "weight")]
> pca.a <- prcomp(a, scale = TRUE)
> summary(pca.a)

# Importance of components:
#       Comp.1 Comp.2 Comp.3 Comp.4
# Standard deviation 1.8704 0.49540 0.40390 0.30518
# Proportion of Variance 0.8746 0.06135 0.04078 0.02328
# Cumulative Proportion 0.8746 0.93593 0.97672 1.00000

> pca.a$rotation

          PC1     PC2     PC3     PC4
mpg  0.4833271 0.8550485 -0.02994982 0.1854453
cylinders -0.5033993 0.3818233 -0.55748381 -0.5385276
horsepower -0.4984381 0.3346173  0.79129092 -0.1159714
weight  -0.5143380 0.1055192 -0.24934614  0.8137252
```
Scree plot

Scree plot: Auto data set

Proportion of Variance Explained

Cumulative Proportion of Variance Explained

Principal Component

Scree plot: Auto data set

Principal Component
A biplot displays both the PC scores and the PC loadings.
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).
In general, a $m \times n$ matrix $X$ 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 original sample
  - Get test error
- Get mean test error = pessimistic error \( e_o \)
- **generalization error estimation** = \( 0.368 \times e_l + 0.632 \times e_o \)