# Statistical Machine Learning for Data Science

Zaid Harchaoui

DATA 558

Week 5

# Lecture 4: Outline

- Feature selection with LASSO

# Ridge Regression

Training data $(x_1, y_1), \ldots, (x_n, y_n)$ in $\mathbb{R}^d \times R$.

$$\min_{\beta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \|\beta\|_2^2 ,$$

that is, if you expand

$$\min_{\beta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left( y_i - \sum_{j=1}^{d} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{d} \beta_j^2 .$$

# Empirical risk and Regularization

$$\text{Residual Sum of Squares (RSS)} \quad \sum_{i=1}^{n}(y_i - x_i^T\beta)^2$$

$$\text{Regularization Penalty} \quad \lambda\|\beta\|_2^2$$

# Regularization

The term

$$\lambda \|\beta\|_2^2$$

is also called a *shrinkage penalty*.

It has the effect of *shrinking the estimates of* $\beta_j$ *towards 0*.

$$\begin{aligned} \lambda \approx 0 \quad & \text{no effect} \\ \lambda \to \infty \quad & \text{shrinkage towards 0} \end{aligned}$$

# Where is the intercept?

**Standardize the data before you solve the ridge regression problem.**

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{1/n \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)^2}}$$

Then the intercept is given by the simple formula

$$\hat{\beta}_0 = \frac{1}{n} \sum_{i=1}^{n} y_i \ .$$

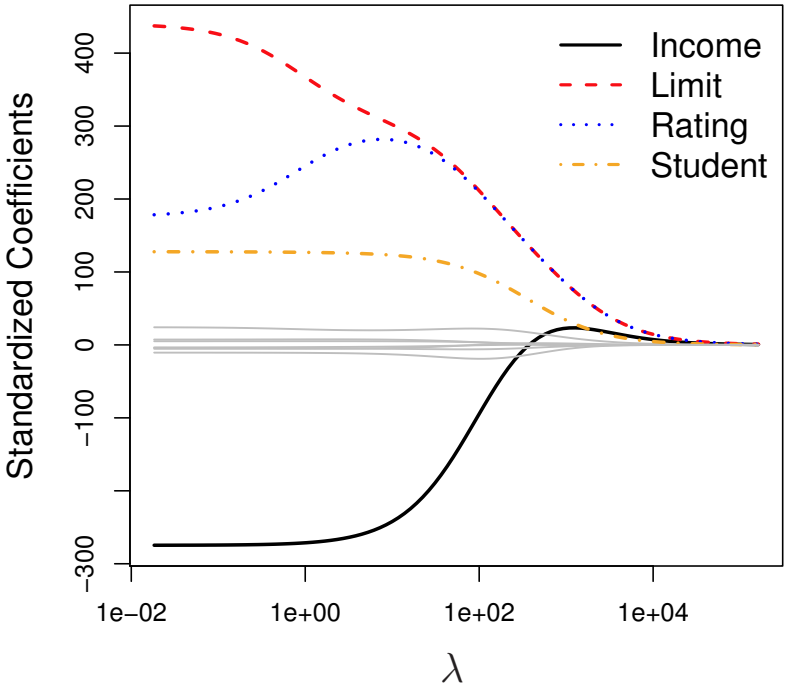
and does need to be found by the optimization algorithm.

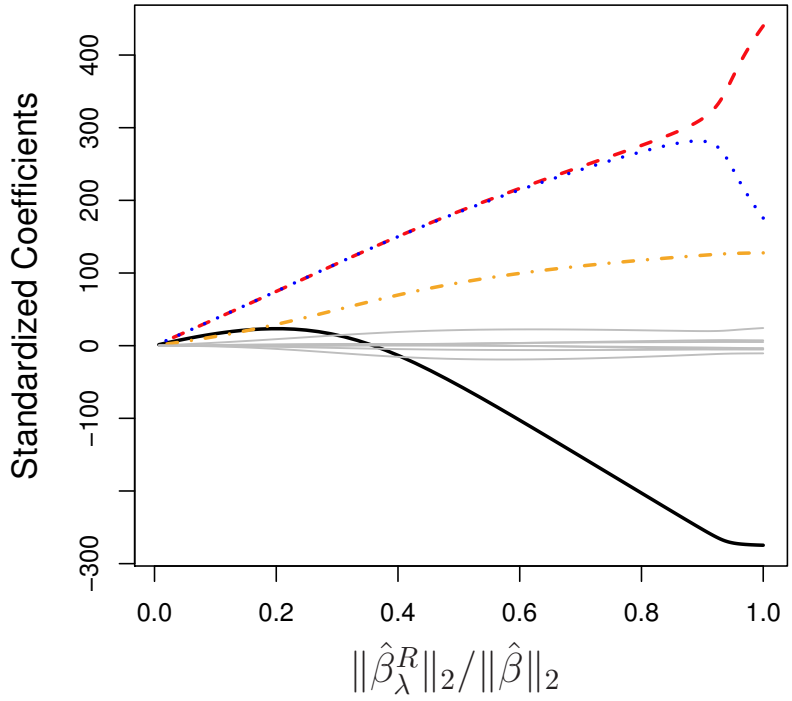
# Example: Credit Data

Predict **balance** (average credit card debt for a number of individuals) from the predictors:

- age
- cards (number of credit cards)
- education (years of education)
- income (in thousands of dollars)
- limit (credit limit)
- rating (credit rating
- gender
- student (student
- status (marital status)
- ethnicity (Caucasian, African American or Asian).

# Example: Credit Data




$\hat{\beta}_\lambda^R$ Ridge regression with reg. penalty $\lambda$

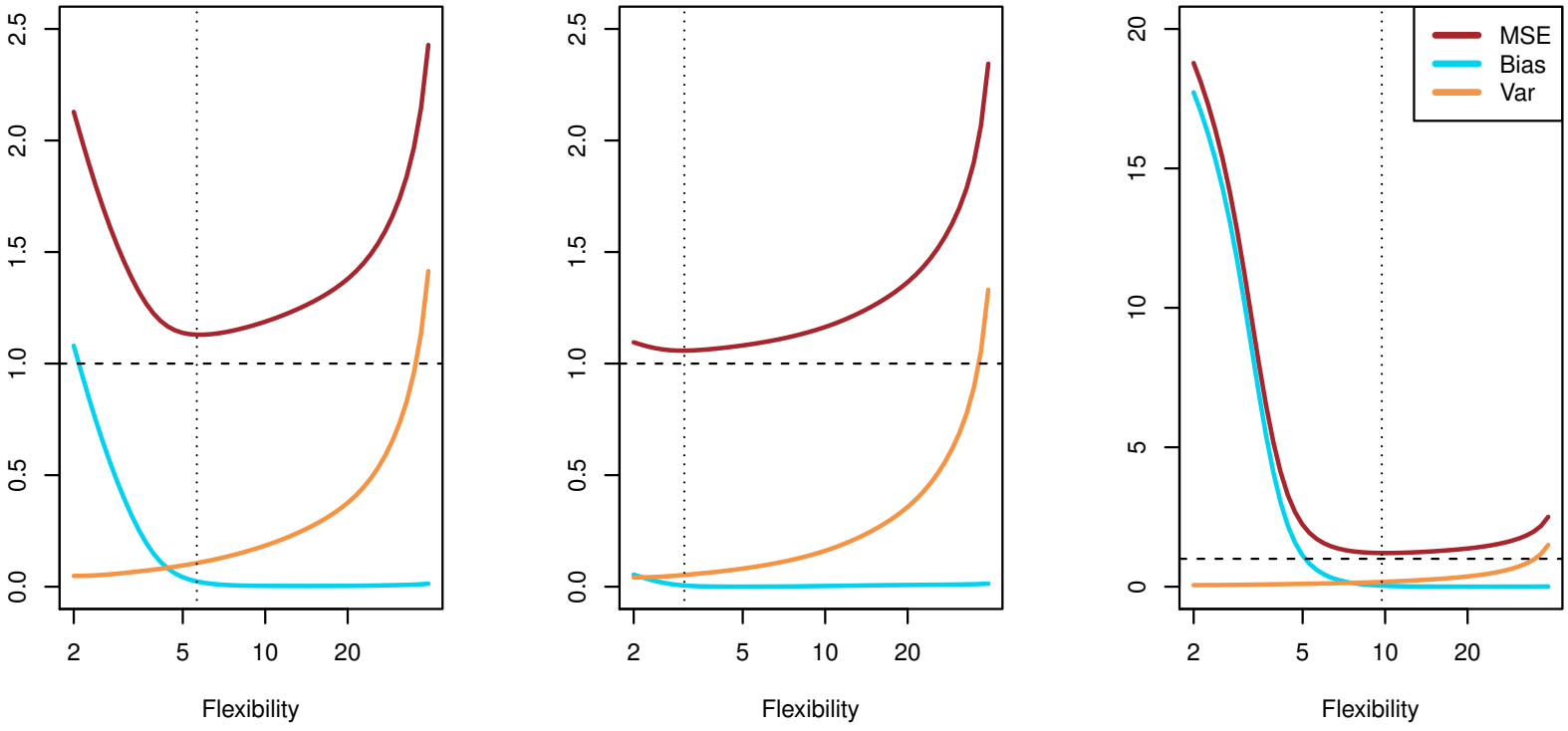
$\hat{\beta}$ Ridge regression with reg. penalty $\infty$ (OLS)

# Bias-Variance trade-off

$$\underbrace{\mathbb{E}(y - \hat{f}(x))^2}_{\text{Expected test MSE}} = \text{Var}(f(x)) + [\text{Bias}(f(x))]^2 + \text{Var}(\epsilon)$$

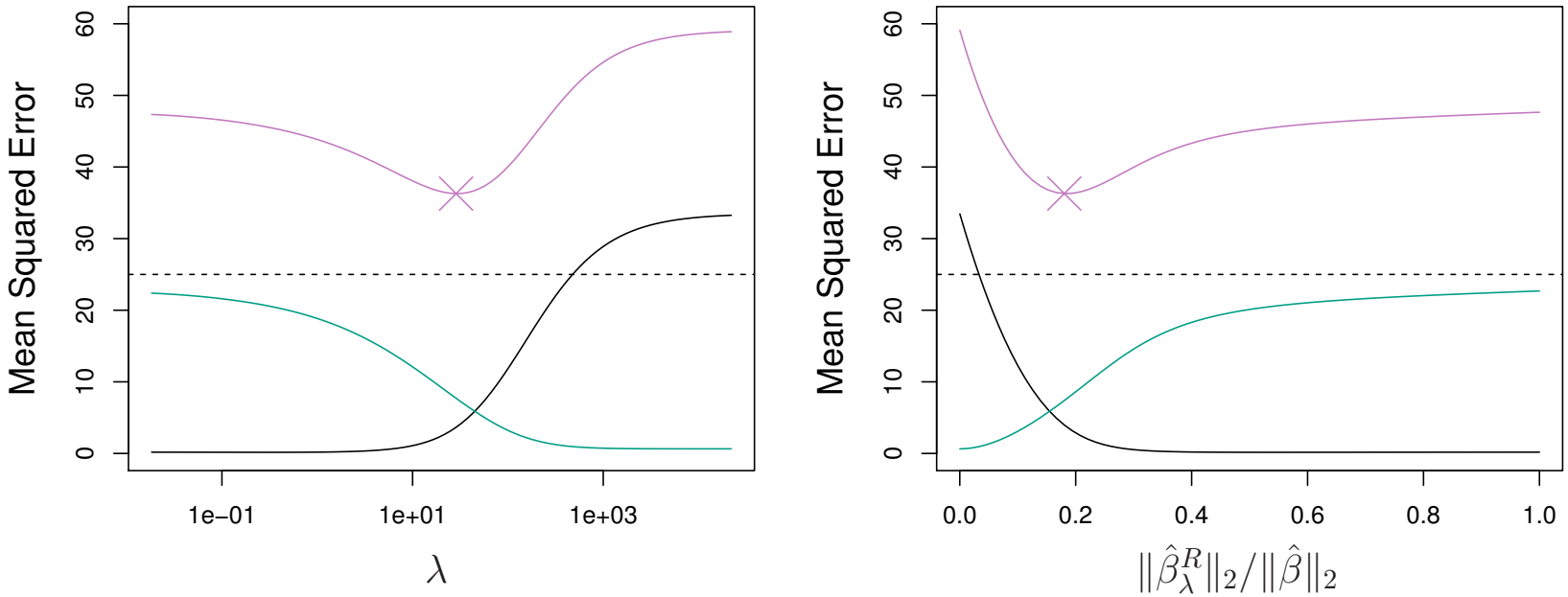
- **Expected MSE**: average test MSE over infinite number of training sets.
- **Variance**: the amount of which $\hat{f}$ would change if we estimated it using a different training set.
- **Bias**: the error introduced by adopting this particular model

# Example: Bias-Variance trade-off




Squared bias (blue curve), variance (orange curve), Var (dashed line), and test MSE (red curve).
Vertical dotted line indicates flexibility level corresponding to smallest test MSE.

# Bias-Variance trade-off




Squared bias (black), variance (green), and test mean squared error (purple) for ridge regression predictions on simulated data set.
Horizontal dashed lines indicate minimum possible MSE.
Purple crosses indicate ridge regression models for which the MSE is smallest.

# $\ell_2^2$-regularized Logistic Regression

## Statistics notation

Training data $(x_1, y_1), \ldots, (x_n, y_n)$ in $\mathbb{R}^d \times \{0, 1\}$.

$$\min_{\beta \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \left\{ y_i \beta^T x_i \log\left(1 + \exp(\beta^T x_i)\right) \right\} + \lambda \|\beta\|_2^2 .$$

## Machine Learning notation

Training data $(x_1, y_1), \ldots, (x_n, y_n)$ in $\mathbb{R}^d \times \{-1, 1\}$.

$$\min_{\beta \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \left\{ \log\left(1 + \exp(-y_i \beta^T x_i)\right) \right\} + \lambda \|\beta\|_2^2 ,$$

# Penalized likelihood interpretation

## Logit Model

$$p(X) := \mathbb{P}(Y = 1|X = x) = \frac{\exp(f_\beta(x))}{1 + \exp(f_\beta(x))} \ ,$$

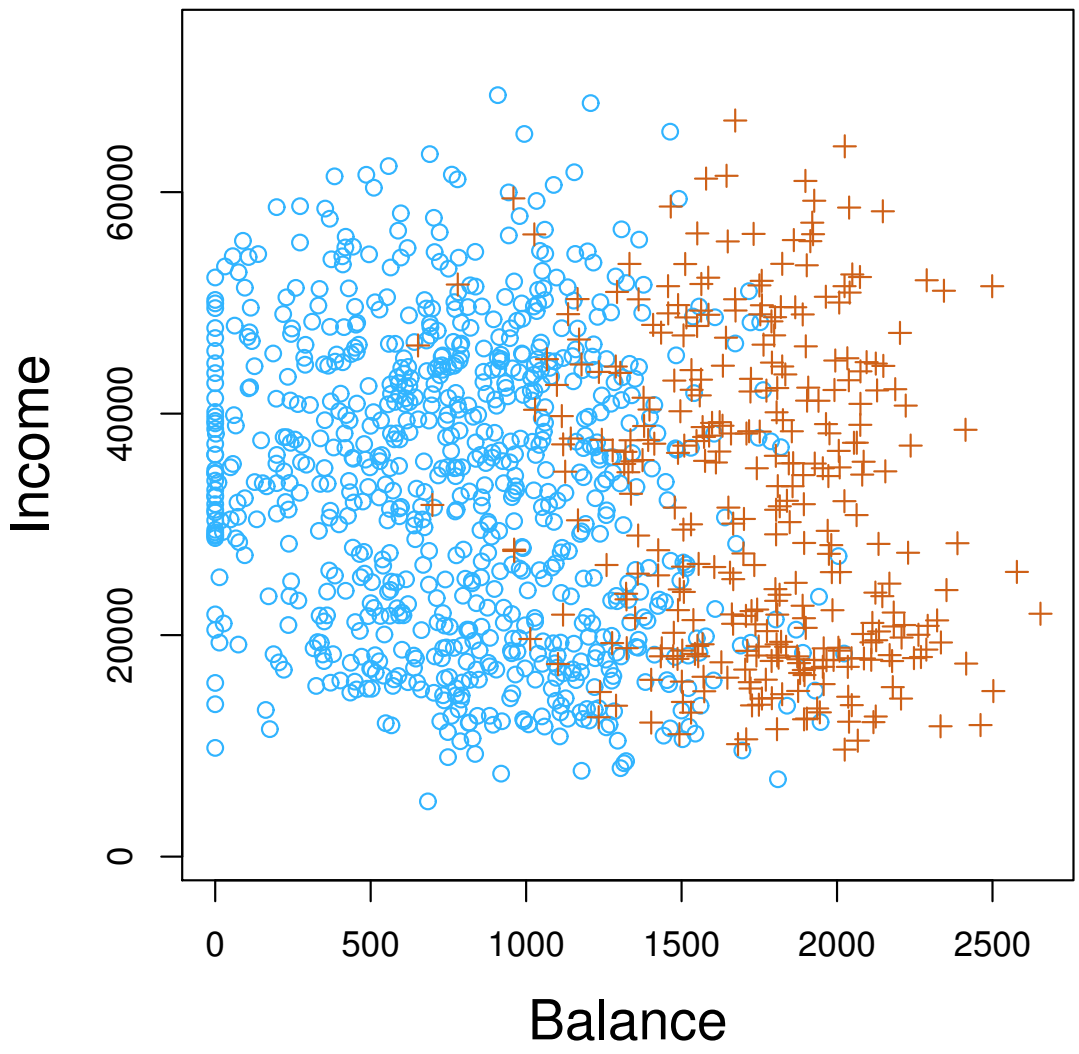
or

$$log\left(\frac{p(X)}{1 - p(X)}\right) = f_\beta(x) \ ,$$

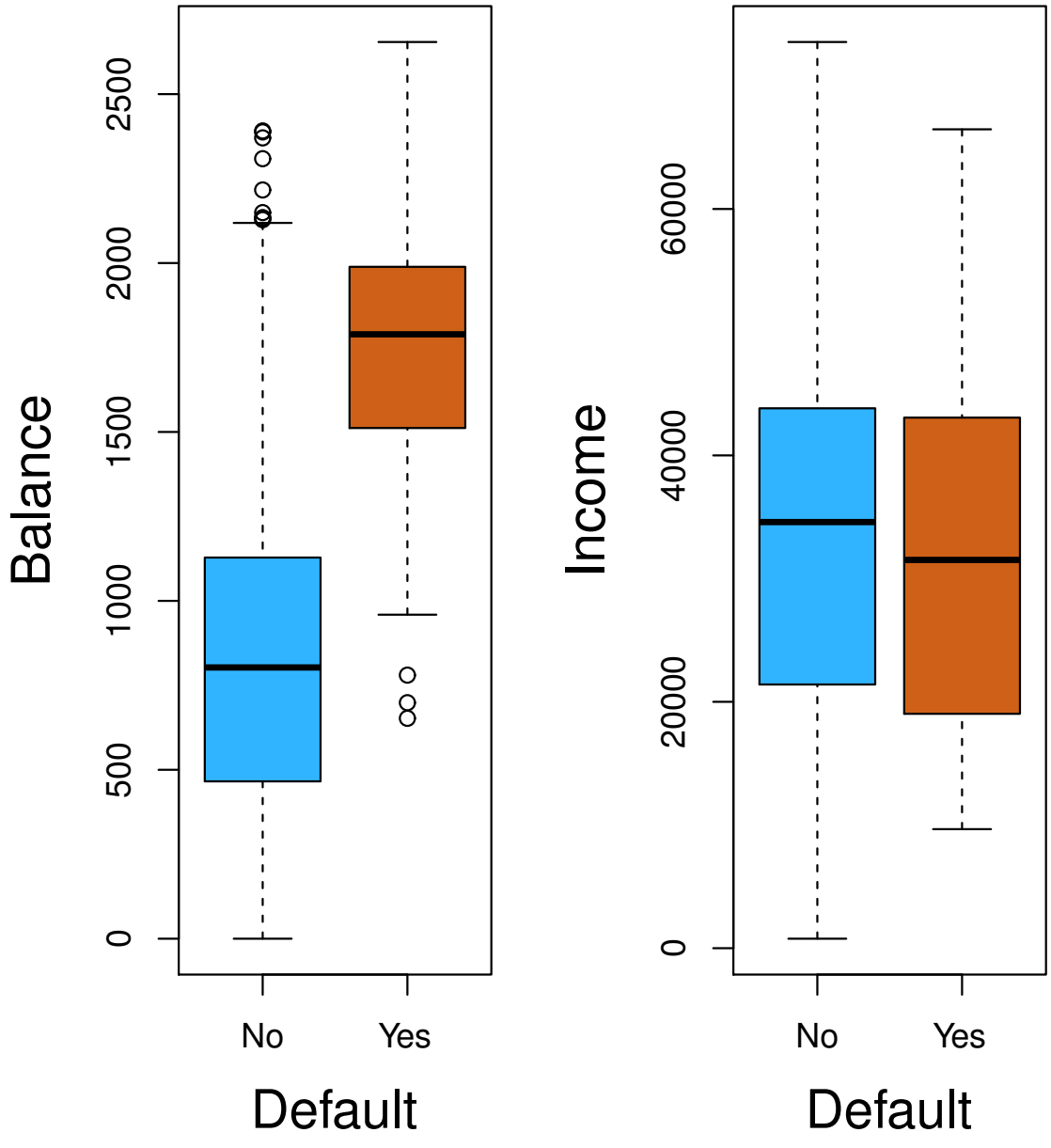
where

$$f_\beta(x) = \beta_0 + \beta^T x \ .$$

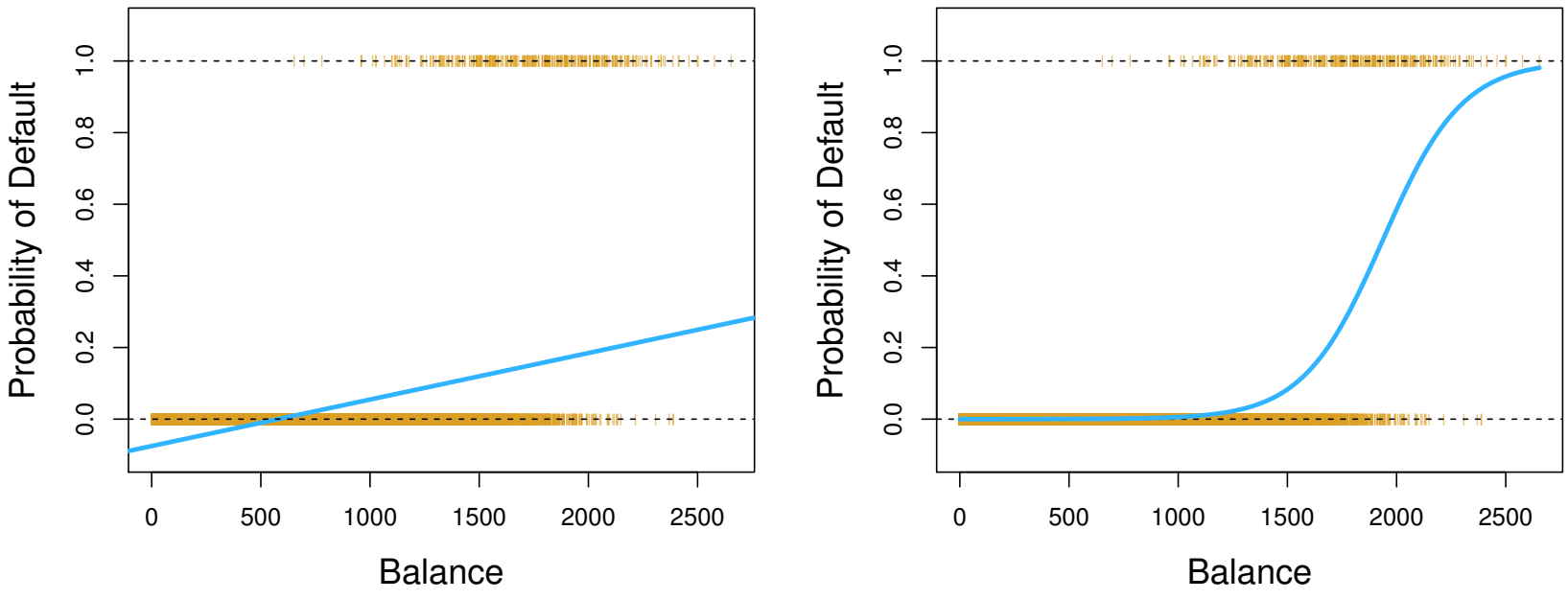
# Example: Credit card default



# Example: Credit card default



# Example: Credit card default




Logistic regression ensures that our estimate for $p(X)$ lies between 0 and 1.

# What about the intercept?

**Standardize the data before you solve the logistic regression problem.**

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{1/n \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)^2}}$$

If there is class imbalance (case-control studies), then the intercept can be adjusted from data for better performance.

# Penalized likelihood interpretation

## Multinomial Logit Model

$$log\left(\frac{\mathbb{P}(Y=1|X=x)}{\mathbb{P}(Y=k|X=x)}\right) = f_{\beta_1}(x)$$
$$log\left(\frac{\mathbb{P}(Y=2|X=x)}{\mathbb{P}(Y=k|X=x)}\right) = f_{\beta_2}(x)$$
$$\cdots\cdots$$
$$log\left(\frac{\mathbb{P}(Y=k-1|X=x)}{\mathbb{P}(Y=k|X=x)}\right) = f_{\beta_{k-1}}(x)$$

where the choice of the reference class is arbitrary.

# Penalized likelihood interpretation

## Multinomial Logit Model

We have for all $\ell = 1, \ldots, k-1$

$$\mathbb{P}(Y = \ell | X = x) = \frac{\exp(\beta_{\ell,0} + \beta_\ell^T x)}{1 + \sum_{m=1}^{k-1} \exp(\beta_{m,0} + \beta_m^T x)}$$

and

$$\sum_{\ell=1}^{k} \mathbb{P}(Y = \ell | X = x) = 1 \, .$$

# Supervised learning

## General objective

Let $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \{1, \ldots, k\}$ be labelled training examples

$$\min_{B \in \mathbb{R}^{d \times k}} \quad \lambda \, \Omega(B) + \frac{1}{n} \sum_{i=1}^{n} L\big(y_i, B^T x_i\big)$$

Large-scale setting

$$n \gg 1, \quad d \gg 1, \quad k \gg 1$$

# Gradient descent

## Gradient descent

Grouping the regularization penalty and the empirical risk in the objective

$$\nabla_B F(B) = \frac{1}{n} \sum_{i=1}^{n} \left\{ n\lambda \, \Omega(B) + L\left(y_i, B^T x_i\right) \right\}$$

# Gradient descent

## Gradient descent

Grouping the regularization penalty and the empirical risk, and expanding the sum onto the examples

$$\nabla_B F(B) = \frac{1}{n} \sum_{i=1}^{n} \left\{ n\lambda \, \Omega(B) + L\left(y_i, B^T x_i\right) \right\}$$

$$= \nabla_B \left\{ \frac{1}{n} \sum_{i=1}^{n} L(B; x_i, y_i) \right\}$$

# Gradient descent

## Gradient descent

- **Initialize**: $B = 0$
- **Iterate**:

$$
\begin{aligned}
B_{t+1} &= B_t - \eta_t \nabla F(B) \\
&= B_t - \eta_t \nabla_B \left\{ \frac{1}{n} \sum_{i=1}^{n} L(B; x_i, y_i) \right\}
\end{aligned}
$$

# Gradient descent

## Gradient descent

- **Initialize**: $B = 0$
- **Iterate**:

$$
\begin{aligned}
B_{t+1} &= B_t - \eta_t \nabla_B F(B) \\
&= B_t - \eta_t \nabla_B \left\{ \frac{1}{n} \sum_{i=1}^{n} L(B; x_i, y_i) \right\}
\end{aligned}
$$

# Gradient descent

## Strengths and weaknesses

- Strength: robust to setting of step-size sequence (line-search)
- Weakness: demanding disk/memory requirements

# Tuning the step-size

## Initial step-size estimate

- **standardize the data**
- use the formula

$$\eta_0 \propto \frac{1}{L}\,, \quad \text{where } L = \text{maximum-eigenvalue}\left(\frac{1}{n}X^T X\right) + \lambda$$

## Practical advice

- Subsample the dataset
- Python $\rightarrow$ `largest-eigh`

# Gradient descent with adaptive step-size

- **Initialize**: $B_0 = 0$.
- **Iterate**:

  Find $\eta_t$ with backtracking rule.

$$
\begin{aligned}
B_{t+1} &= B_t - \eta_t \nabla_B F(B) \\
&= B_t - \eta_t \nabla_B \left\{ \frac{1}{n} \sum_{i=1}^{n} L(B; x_i, y_i) \right\}
\end{aligned}
$$

# Fast Gradient Method

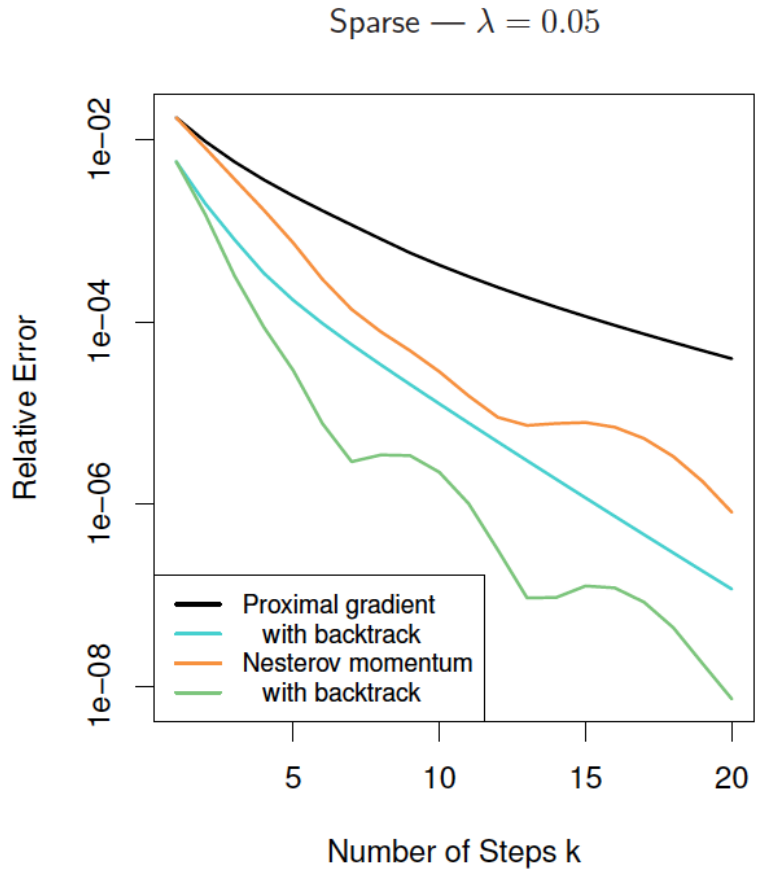
- **Initialize**: $B = 0$ and $\theta_0 = 0$.
- **Iterate**:

  Find $\eta_t$ with backtracking rule.

$$B_{t+1} = \theta_t - \eta_t \nabla_\theta F(\theta)$$

$$\theta_{t+1} = B_{t+1} + \frac{t}{t+3}(B_{t+1} - B_t)$$

# Accelerated Gradient Method




Performance of the gradient descent versus accelerated gradient on a regression problem.

# Large-scale supervised learning

## General form

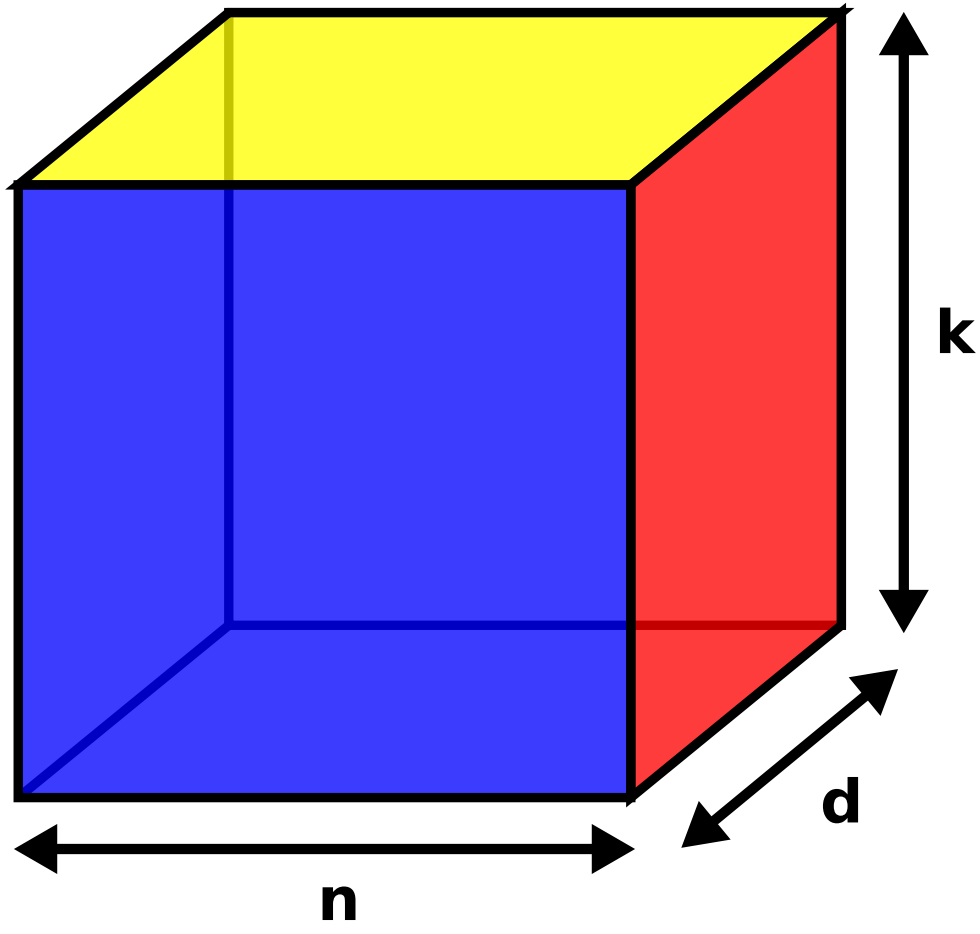
Let $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \{1, \ldots, k\}$ be labelled training examples

$$\min_{B \in \mathbb{R}^{d \times k}} \quad \lambda \, \Omega(B) + \frac{1}{n} \sum_{i=1}^{n} L\big(y_i, B^T x_i\big)$$

Problem: minimizing such objectives in the **large-scale** setting

$$n \gg 1, \quad d \gg 1, \quad k \gg 1$$

# Machine learning cuboid



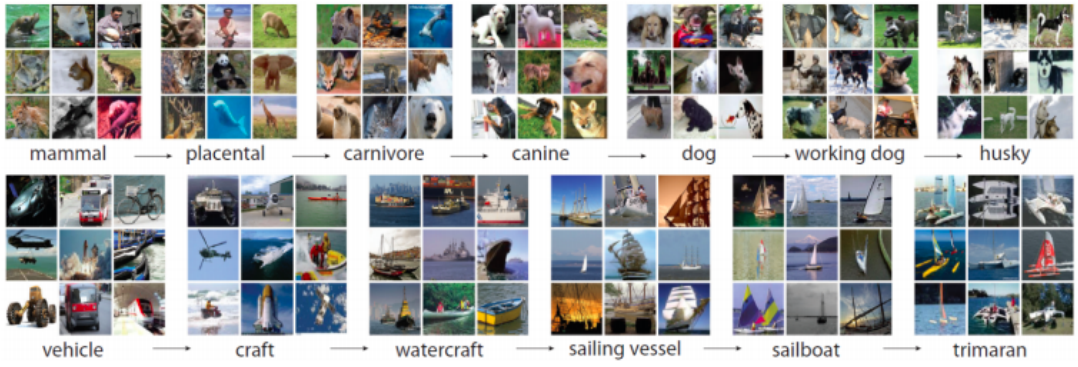
# An example: ImageNet dataset

## ImageNet dataset

- Large number of images/examples: $n = 17,000,000$
- Large number of pixels/image: $d = 200,030$
- Large number of categories: $k = 10,000$

# Zoom on the ImageNet Dataset

Hierarchy of classes:



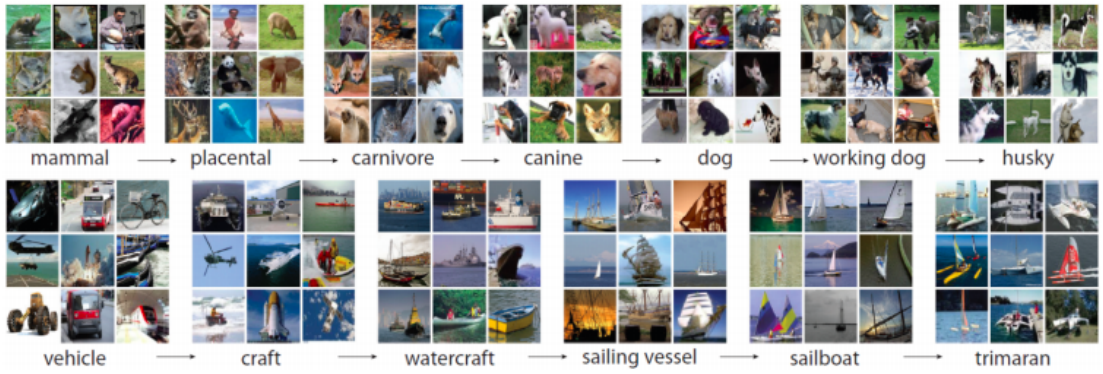

Deng, Dong, Socher, Li, Li and Fei-Fei, "Imagenet: a large-scale hierarchical image database", CVPR'09.

Fine-grained subsets: generally more practical problems


→ Fungus: 134 classes, 90K images

# Zoom on the ImageNet Dataset

Hierarchy of classes:




Deng, Dong, Socher, Li, Li and Fei-Fei, "Imagenet: a large-scale hierarchical image database", CVPR'09.

Fine-grained subsets: generally more practical problems

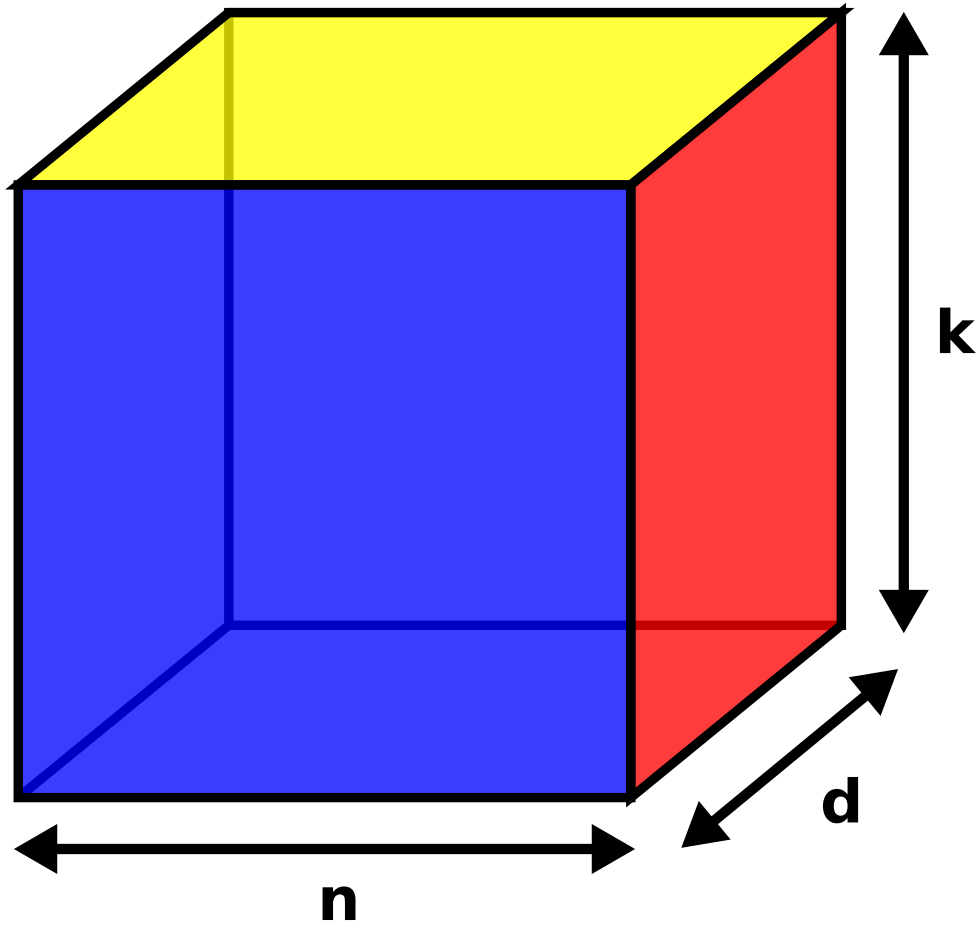

→ Vehicle: 262 classes, 226K images

# Strategies for large-scale problems

## Strategy: divide-and-conquer

Break the large learning problem into small and easy pieces

# Machine learning cuboid



# Aggregation or Decomposition principles

## Aggregation or Decomposition principles

- Decomposition over examples: Stochastic gradient descent
- Decomposition over categories: One-versus-rest strategy
- Decomposition over features: Coordinate descent
- Decomposition over set of image transformations: transformation pursuit

# Decomposition over examples

## Stochastic gradient descent

- Bru, 1890: algorithm to adjust a slant $\theta$ of cannon in order to obtain a specified range $r$ by trial and error, firing one shell after another

$$\theta_t = \theta_{t-1} - \frac{\eta_0}{t}(r - r_t)$$

- Perceptron, Rosenblatt, 1957

$$\begin{aligned} \beta_t &= \beta_{t-1} - \eta_t(y_t \phi(x_t)) && \text{if } y_t \phi(x_t) \leq 0 \\ &= \beta_{t-1} && \text{otherwise} \end{aligned}$$

# Decomposition over examples

## Stochastic gradient descent

- Bru, 1890: algorithm to adjust a slant $\theta$ of cannon in order to obtain a specified range $r$ by trial and error
- Perceptron, Rosenblatt, 1957
- 60s-70s: extensions in learning, optimal control, and adaptive signal processing
- 80s-90s: extensions to non-convex learning problems
- see "Efficient backprop" in *Neural networks: Tricks of the trade*, LeCun et al., 1998, for wise advice and overview on sgd algorithms

# Decomposition over examples

## Stochastic gradient descent

- **Initialize**: $B = 0$
- **Iterate**: pick an example $(x_t, y_t)$

$$B_{t+1} = B_t - \eta_t \underbrace{\nabla_B L(B; x_t, y_t)}_{\text{one example at a time}}$$

## Why?

Where does these update rules come from?

# Theory digest

## Theory digest

- Fixed stepsize $\eta_t \equiv \eta \longrightarrow$ stable convergence
- Decreasing stepsize $\eta_t = \frac{\eta_0}{t+t_0} \longrightarrow$ faster local convergence, **with $\eta_0$ and $t_0$ properly set**
- Note: stochastic gradient descent is an *extreme* decomposition strategy picking *one example* at a time

## In practice

- Pick a random batch of reasonable size, and find best pair $(\eta_0, t_0)$ through cross-validation
- Run stochastic gradient descent with sequence of decreasing stepsize $\eta_t = \frac{\eta_0}{t+t_0}$

# Tricks of the trades

## Life is simpler in large-scale settings

- Shuffle the examples before launching the algorithm, and process the examples in a **balanced manner** w.r.t the categories
- Regularization through early stopping: perform only a few several passes/epochs over the training data, and stop when the accuracy on a held-out validation set does not increase anymore
- Fixed step-size works fine: find best $\eta$ through cross-validation on a small batch

# Stochastic gradient descent

## Put the shoulder to the wheel

Let's try it out!

# Ridge regression

## Ridge regression

Training data: $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$

$$\min_{\beta \in \mathbb{R}^d} \quad \frac{\lambda}{2}\|\beta\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, \beta^T x_i\right)$$

## Key calculations

$$Q\beta; x_i, y_i) := \frac{n\lambda}{2}\|\beta\|_2^2 + (y_i - \beta^T x_i)^2$$

$$\nabla Q(\beta; x_i, y_i) = n\lambda\beta + (y_i - \beta^T x_i)x$$

# Logistic regression

## Logistic regression

Training data: $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$

$$\min_{\beta \in \mathbb{R}^d} \quad \frac{\lambda}{2}\|\beta\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, \beta^T x_i\right)$$

## Key calculations

$$Q(\beta; x_i, y_i) := \frac{n\lambda}{2}\|\beta\|_2^2 + \log\left(1 + \exp(-y_i \beta^T x_i)\right))$$

$$\nabla Q(\beta; x_i, y_i) = n\lambda\beta + -\frac{1}{1 + \exp(y_i \beta^T x_i)} y_i x_i$$

# Linear SVM with linear hinge loss

## Linear SVM with linear hinge loss

Training data: $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$

$$\min_{\beta \in \mathbb{R}^d} \quad \frac{\lambda}{2}\|\beta\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, \beta^T x_i\right)$$

## Key calculations

$$Q(\beta; x_i, y_i) := \frac{n\lambda}{2}\|\beta\|_2^2 + \max\left(0, 1 - y_i \beta^T x_i\right)$$

$$\nabla Q(\beta; x_i, y_i) = \begin{cases} n\lambda\beta - y_i x_i & \text{if } 1 - y_i x_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

# Linear SVM with linear hinge loss

## Linear SVM with linear hinge loss

Training data: $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$

$$\min_{\beta \in \mathbb{R}^d} \quad \frac{\lambda}{2}\|\beta\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, \beta^T x_i\right)$$

## Non-differentiable loss

- Rule: if $L(\beta; x, y)$ has a finite number of a non-differentiable points, then just make **no update**, and pick another example.
- Theoretical justification: the set of a non-differentiable points will have measure zero, and convergence guarantee is still valid

# A quick overview

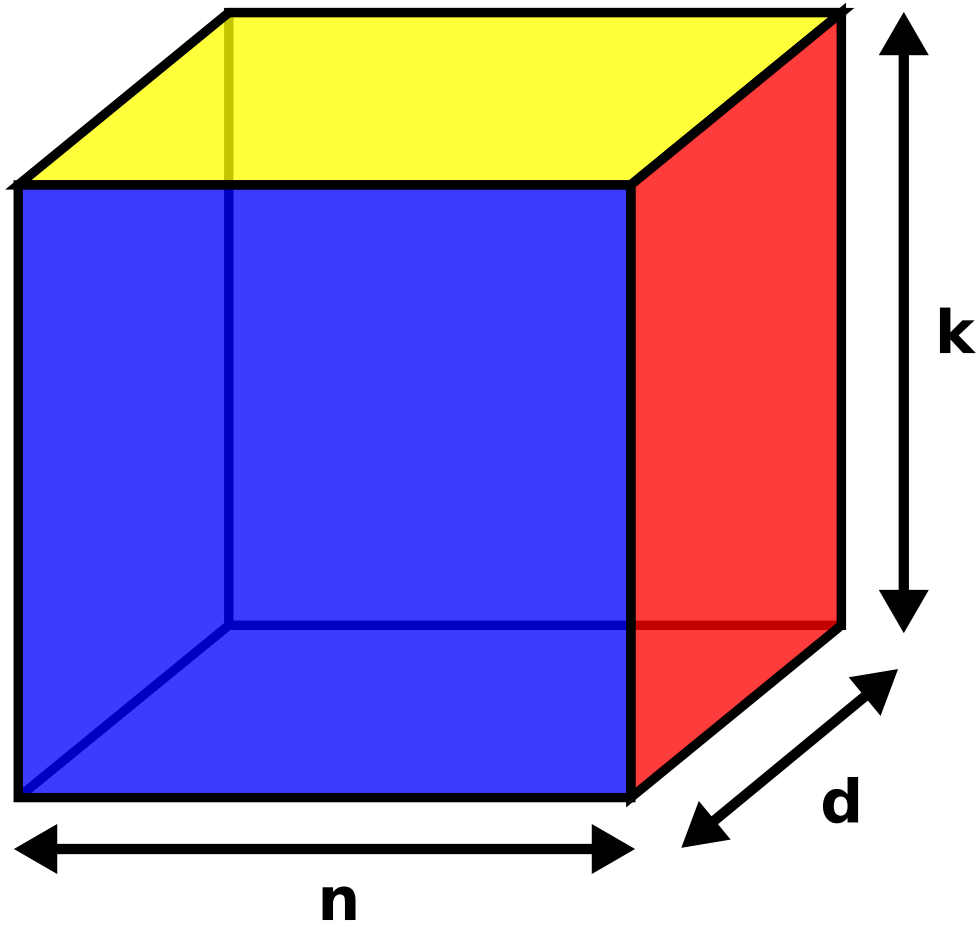
## Convergence guarantees

- Least-square loss: smooth $\rightarrow$ fast and stable convergence
- Logistic loss: smooth $\rightarrow$ fast and stable convergence
- Linear hinge loss: non-smooth $\rightarrow$ slower convergence

## Convergence guarantees

Take-home message: smooth loss is nicer

# Machine learning cuboid



# Decomposition principle

## Decomposition principle

- Decomposition over examples: Stochastic gradient descent
- **Decomposition over categories: One-versus-rest strategy**
- Decomposition over features: Coordinate Descent
- Decomposition over set of image transformations: transformation pursuit

# Multi-class linear SVM with regular linear hinge loss

## Multi-class linear SVM with regular linear hinge loss

Training data: $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \{0, 1\}^k$

$$\min_{B \in \mathbb{R}^{d \times k}} \quad \lambda \|\beta\|_2^2 + \frac{1}{n} \sum_{i=1}^{n} \text{BinaryLogLoss}_i$$

## One-versus-rest reduction

- Turn original label $y_i \in \{0, 1\}^k$ into binary label $\tilde{y}_i \in \{-1, +1\}$

$$\text{BinaryLogLoss}_i = \max(0, 1 - \tilde{y}_i \beta^T x_i)$$

- Note: any binary loss could do, i.e. also the **binary linear**

# Multi-class linear SVM with regular linear hinge loss

## Multi-class linear SVM with regular linear hinge loss

Training data: $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \{0, 1\}^k$

$$\min_{\beta \in \mathbb{R}^{d \times k}} \quad \sum_{\ell=1}^{k} \lambda_\ell \left\| \beta_\ell \right\|_2^2 + \frac{1}{n} \sum_{\ell=1}^{k} \sum_{\substack{i=1 \\ y_i \equiv \text{class } \ell}}^{n} \text{BinaryLogLoss}_i$$

## Decomposition over categories

Leverage **decomposable structure** over categories

# Multi-class linear SVM with regular linear hinge loss

## Multi-class linear SVM with regular linear hinge loss

Training data: $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \{0, 1\}^k$

$$
\begin{cases}
\min\limits_{\beta_1 \in \mathbb{R}^d} & \lambda_1 \left\|\beta_1\right\|_2^2 + \dfrac{1}{n} \sum\limits_{\substack{i=1 \\ y_i \equiv \text{class } 1}}^{n} \text{BinaryLogLoss}_i \\
\ldots & \\
\min\limits_{\beta_\ell \in \mathbb{R}^d} & \lambda_\ell \left\|\beta_\ell\right\|_2^2 + \dfrac{1}{n} \sum\limits_{\substack{i=1 \\ y_i \equiv \text{class } \ell}}^{n} \text{BinaryLogLoss}_i \\
\ldots & \\
\min\limits_{\beta_k \in \mathbb{R}^d} & \lambda_k \left\|\beta_k\right\|_2^2 + \dfrac{1}{n} \sum\limits_{\substack{i=1 \\ y_i \equiv \text{class } k}}^{n} \text{BinaryLogLoss}_i
\end{cases}
$$

# Multi-class through one-vs-rest

## Multi-class through one-vs-rest

- Overall: simplest multi-class classification algorithm
- Computational strength: easy to optimize by decomposition over classes
- Statistical weakness: no universally consistent loss can be decomposable over classes (do we really care? we'll see)

# Multi-class through one-vs-rest

## In practice

State-of-the-art performance using a **balanced** version of the binary loss, and **learning** the optimal imbalance $\beta$ through cross-validation

$$\text{Empirical risk} = \frac{\beta}{n^+} \sum_{i \in \text{positive examples}} \text{BinaryLogLoss}_i + \frac{1-\beta}{n^-} \sum_{i \in \text{negative examples}} \text{BinaryLogLoss}_i$$

# Multi-class with non-decomposable loss functions

## Other multi-class loss functions

- Multinomial logistic loss
- Crammer & Singer multi-class loss

$$R_{\text{MUL}} = \frac{1}{n} \sum_{i=1}^{n} \left\{ \max_{y} (\Delta(y_i, y) + \beta_y^T x_i) - \beta_{y_i}^T x_i \right\}$$

The multi-class binary hinge loss and the multi-class logistic loss are the only **decomposable** losses.

# Multi-class with non-decomposable loss functions

## More sophisticated losses

Loss functions tailored to optimize a nice surrogate to top-$k$ accuracy

$$\text{Accuracy}_{\text{top}-k} = \frac{\#\text{ ex. with correct label lies in top-k scores}}{\text{Total number of ex.}}$$

- Ranking losses

$$R_{\text{RNK}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{y=1}^{k}\max_{y}\left(0, \Delta(y_i, y) - (\beta_{y_i}^T - \beta_y)^T x_i\right)$$

- Weighted ranking losses, and other variations

# Lasso Regression

## Lasso Regression (Penalized Form)

The lasso regression solution for regularization parameter $\lambda \geq 0$ is

$$\hat{\beta} = \underset{\beta \in \mathbf{R}^d}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \left\{ \beta^T x_i - y_i \right\}^2 + \lambda \|\beta\|_1,$$

where $\|\beta\|_1 = |\beta_1| + \cdots + |\beta_d|$ is the $\ell_1$-norm.

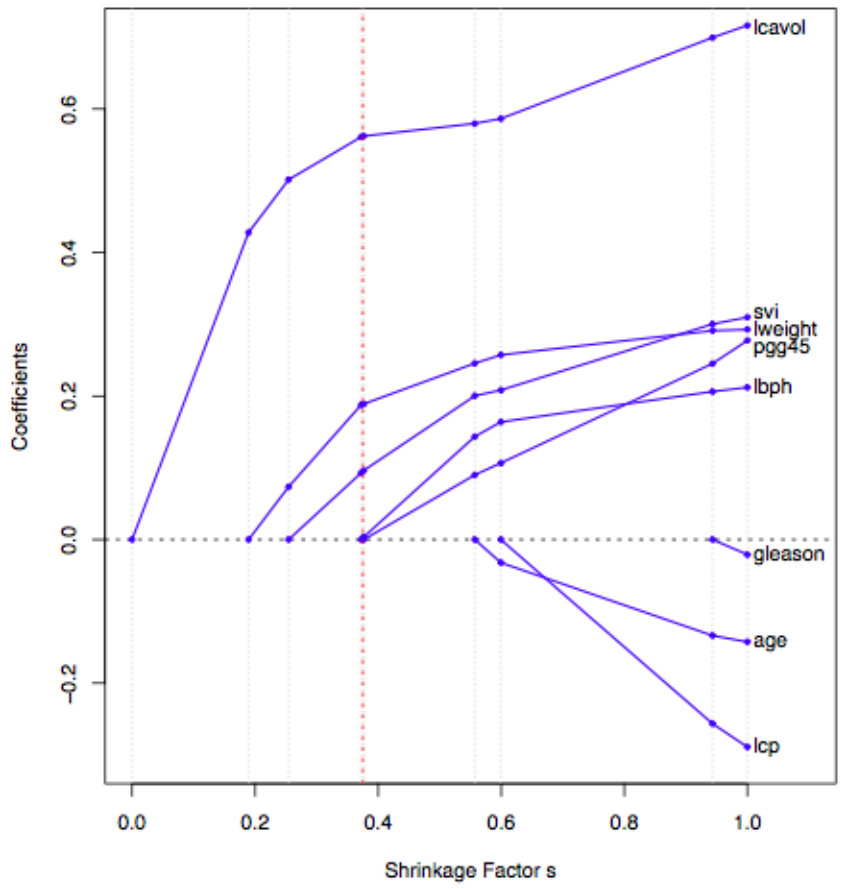
# Lasso Regression

## Lasso Regression (Constrained form)

The lasso regression solution for regularization parameter $r \geq 0$ is

$$\hat{\beta} = \underset{\|\beta\|_1 \leq r}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \left\{ \beta^T x_i - y_i \right\}^2 .$$

# Lasso Regression: Regularization Path




Shrinkage Factor $s = r / \left|\hat{\beta}\right|_1$, where $\hat{\beta}$ is the ERM (the unpenalized fit).

# Lasso Gives Feature Sparsity

- Time/expense to compute/buy features

# Lasso Gives Feature Sparsity

- Time/expense to compute/buy features
- Memory to store features (e.g. real-time deployment)

# Lasso Gives Feature Sparsity

- Time/expense to compute/buy features
- Memory to store features (e.g. real-time deployment)
- Identifies the important features

# Lasso Gives Feature Sparsity

- Time/expense to compute/buy features
- Memory to store features (e.g. real-time deployment)
- Identifies the important features
- Better prediction? sometimes

# Lasso Gives Feature Sparsity

- Time/expense to compute/buy features
- Memory to store features (e.g. real-time deployment)
- Identifies the important features
- Better prediction? sometimes
- As a feature-selection step for training a slower non-linear model

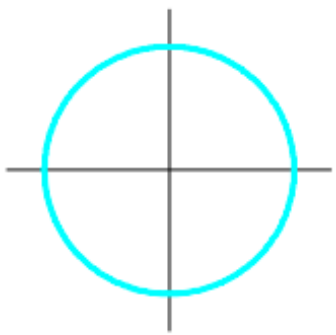
# Constrained and Penalized Equivalent?

- For ridge regression and lasso regression,
  - the Constrained and Penalized formulations are equivalent
- We will use whichever form is most convenient.

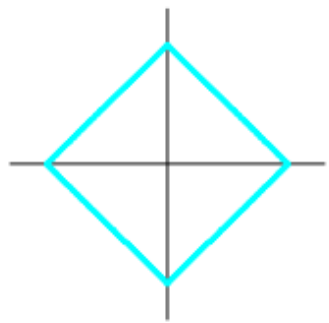
# The $\ell_1$ and $\ell_2$ Norm Constraints

- For visualization, restrict to 2-dimensional input space
- $\mathcal{F} = \{f(x) = \beta_1 x_1 + \beta_2 x_2\}$ (linear hypothesis space)
- Represent $\mathcal{F}$ by $\{(\beta_1, \beta_2) \in \mathbf{R}^2\}$.

- $\ell_2$ contour: $\beta_1^2 + \beta_2^2 = r$



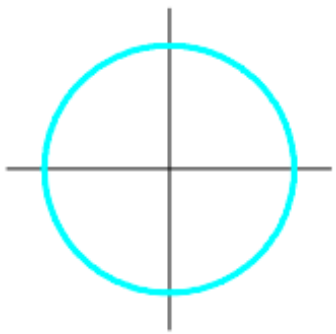
- $\ell_1$ contour: $|\beta_1| + |\beta_2| = r$



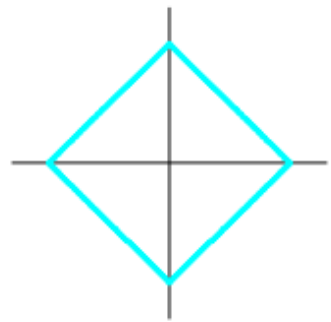
# The $\ell_1$ and $\ell_2$ Norm Constraints

- For visualization, restrict to 2-dimensional input space
- $\mathcal{F} = \{f(x) = \beta_1 x_1 + \beta_2 x_2\}$ (linear hypothesis space)
- Represent $\mathcal{F}$ by $\{(\beta_1, \beta_2) \in \mathbf{R}^2\}$.

- $\ell_2$ contour: $\beta_1^2 + \beta_2^2 = r$



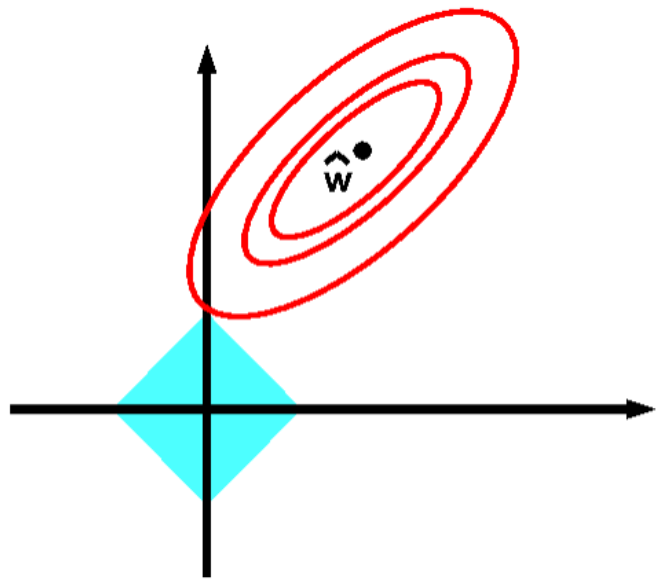
- $\ell_1$ contour: $|\beta_1| + |\beta_2| = r$



Where are the "sparse" solutions?

# The Famous Picture for $\ell_1$ Regularization

- $\min_r f_r =$
  $\arg\min_{\beta \in \mathbf{R}^2} \frac{1}{n} \sum_{i=1}^n \left(\beta^T x_i - y_i\right)^2$ subject to $|\beta_1| + |\beta_2| \leq r$




- Red lines: contours of $\hat{R}_n(\beta) = \sum_{i=1}^n \left(\beta^T x_i - y_i\right)^2$.
- Blue region: Area satisfying complexity constraint:
  $|\beta_1| + |\beta_2| \leq r$

# The Empirical Risk for Square Loss

- Denote the empirical risk of $f(x) = \beta^T x$ by

$$\hat{R}_n(\beta) = \sum_{i=1}^{n} \left(\beta^T x_i - y_i\right)^2 = ||X\beta - y||^2$$

# The Empirical Risk for Square Loss

- Denote the empirical risk of $f(x) = \beta^T x$ by

$$\hat{R}_n(\beta) = \sum_{i=1}^{n} \left(\beta^T x_i - y_i\right)^2 = ||X\beta - y||^2$$

- $\hat{R}_n$ is minimized by $\hat{\beta} = \left(X^T X\right)^{-1} X^T y$, the ordinary (un-regularized) least-squares solution.

# The Empirical Risk for Square Loss

- Denote the empirical risk of $f(x) = \beta^T x$ by

$$\hat{R}_n(\beta) = \sum_{i=1}^{n} \left(\beta^T x_i - y_i\right)^2 = ||X\beta - y||^2$$

- $\hat{R}_n$ is minimized by $\hat{\beta} = \left(X^T X\right)^{-1} X^T y$, the ordinary (un-regularized) least-squares solution.
- What does $\hat{R}_n$ look like around $\hat{\beta}$?

# The Empirical Risk for Square Loss

- By completing the quadratic form, we can show for any $\beta \in \mathbf{R}^d$:

$$\hat{R}_n(\beta) = R_{\text{ERM}} + \left(\beta - \hat{\beta}\right)^T X^T X \left(\beta - \hat{\beta}\right)$$

where $R_{\text{ERM}} = \hat{R}_n(\hat{\beta})$ is the optimal empirical risk.

# The Empirical Risk for Square Loss

- By completing the quadratic form, we can show for any $\beta \in \mathbf{R}^d$:

$$\hat{R}_n(\beta) = R_{\text{ERM}} + \left(\beta - \hat{\beta}\right)^T X^T X \left(\beta - \hat{\beta}\right)$$

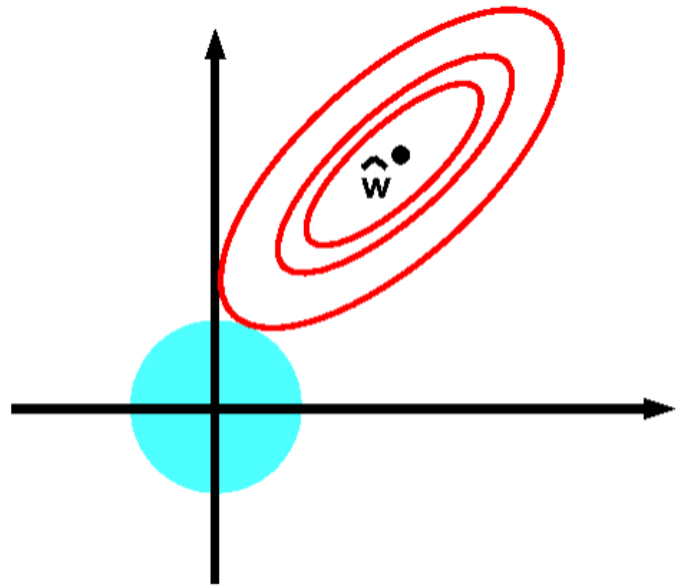
  where $R_{\text{ERM}} = \hat{R}_n(\hat{\beta})$ is the optimal empirical risk.
- Set of $\beta$ with $\hat{R}_n(\beta)$ exceeding $R_{\text{ERM}}$ by $c > 0$ is

$$\begin{aligned}\left\{\beta \mid \hat{R}_n(\beta) = c + R_{\text{ERM}}\right\} \\ = \left\{\beta \mid \left(\beta - \hat{\beta}\right)^T X^T X \left(\beta - \hat{\beta}\right) = c\right\},\end{aligned}$$

  which is an **ellipsoid centered at $\hat{\beta}$.**

# The Famous Picture for $\ell_2$ Regularization

- $\min_r f_r =$
  $\arg\min_{\beta \in \mathbf{R}^2} \frac{1}{n} \sum_{i=1}^{n} \left(\beta^T x_i - y_i\right)^2$ subject to $\beta_1^2 + \beta_2^2 \leq r$




- Red lines: contours of $\hat{R}_n(\beta) = \sum_{i=1}^{n} \left(\beta^T x_i - y_i\right)^2$.
- Blue region: Area satisfying complexity constraint: $\beta_1^2 + \beta_2^2 \leq r$

# How to find the Lasso solution?

- How to solve the Lasso?

$$\min_{\beta \in \mathbf{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(\beta^T x_i - y_i\right)^2 + \lambda \left|\beta\right|_1$$

- $\left|\beta\right|_1$ is not differentiable!

# Coordinate Descent Method

## Coordinate Descent Method

**Goal:** Minimize $L(\beta) = L(\beta_1, \ldots \beta_d)$ over $\beta = (\beta_1, \ldots, \beta_d) \in \mathbf{R}^d$.

- **Initialize** $\beta^{(0)} = 0$
- **while** not converged:
  - Choose a coordinate $j \in \{1, \ldots, d\}$
  - $\beta_j^{\text{new}} \leftarrow \arg\min_{\beta_j} L(\beta_1^{(t)}, \ldots, \beta_{j-1}^{(t)}, \beta_{\mathbf{j}}, \beta_{j+1}^{(t)}, \ldots, \beta_d^{(t)})$
  - $\beta^{(t+1)} \leftarrow \beta^{(t)}$
  - $\beta_j^{(t+1)} \leftarrow \beta_j^{\text{new}}$
  - $t \leftarrow t + 1$

- **randomized coordinate descent**
- **cyclic coordinate descent**

# Coordinate Descent Method for Lasso

- Why mention coordinate descent for Lasso?
- In Lasso, the coordinate minimization is **simple**.

# Coordinate Descent Method for Lasso

## Coordinate Minimization for Lasso

$$\hat{\beta}_j = \underset{\beta_j \in \mathbf{R}}{\arg\min} \sum_{i=1}^{n} \left(\beta^T x_i - y_i\right)^2 + \lambda \left|\beta\right|_1$$

# Coordinate Descent Method for Lasso

## Coordinate Minimization for Lasso

$$\hat{\beta}_j = \underset{\beta_j \in \mathbf{R}}{\arg\min} \sum_{i=1}^{n} \left(\beta^T x_i - y_i\right)^2 + \lambda \left|\beta\right|_1$$

Then

$$\hat{\beta}_j(c_j) = \begin{cases} (c_j + \lambda)/a_j & \text{if } c_j < -\lambda \\ 0 & \text{if } c_j \in [-\lambda, \lambda] \\ (c_j - \lambda)/a_j & \text{if } c_j > \lambda \end{cases}$$

# Coordinate Descent Method for Lasso

## Coordinate Minimization for Lasso

$$\hat{\beta}_j = \arg\min_{\beta_j \in \mathbf{R}} \sum_{i=1}^{n} \left(\beta^T x_i - y_i\right)^2 + \lambda \left|\beta\right|_1$$

Then

$$\hat{\beta}_j(c_j) = \begin{cases} (c_j + \lambda)/a_j & \text{if } c_j < -\lambda \\ 0 & \text{if } c_j \in [-\lambda, \lambda] \\ (c_j - \lambda)/a_j & \text{if } c_j > \lambda \end{cases}$$

$$a_j = 2\sum_{i=1}^{n} x_{i,j}^2 \qquad c_j = 2\sum_{i=1}^{n} x_{i,j}(y_i - \beta_{-j}^T x_{i,-j})$$

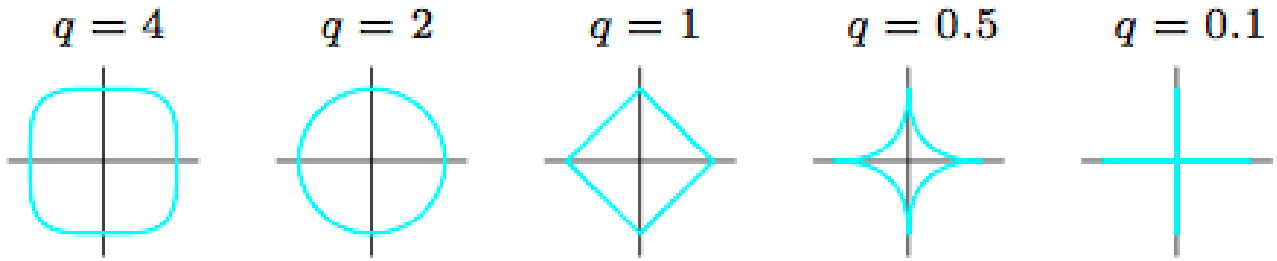
where $\beta_{-j}$ is $\beta$ without component $j$ and similarly for $x_{i,-j}$.

# Coordinate Minimizer for Lasso

$$\hat{\beta}_j(c_j) = \begin{cases} (c_j + \lambda)/a_j & \text{if } c_j < -\lambda \\ 0 & \text{if } c_j \in [-\lambda, \lambda] \\ (c_j - \lambda)/a_j & \text{if } c_j > \lambda \end{cases}$$

# The $(\ell_q)^q$ Norm Constraint

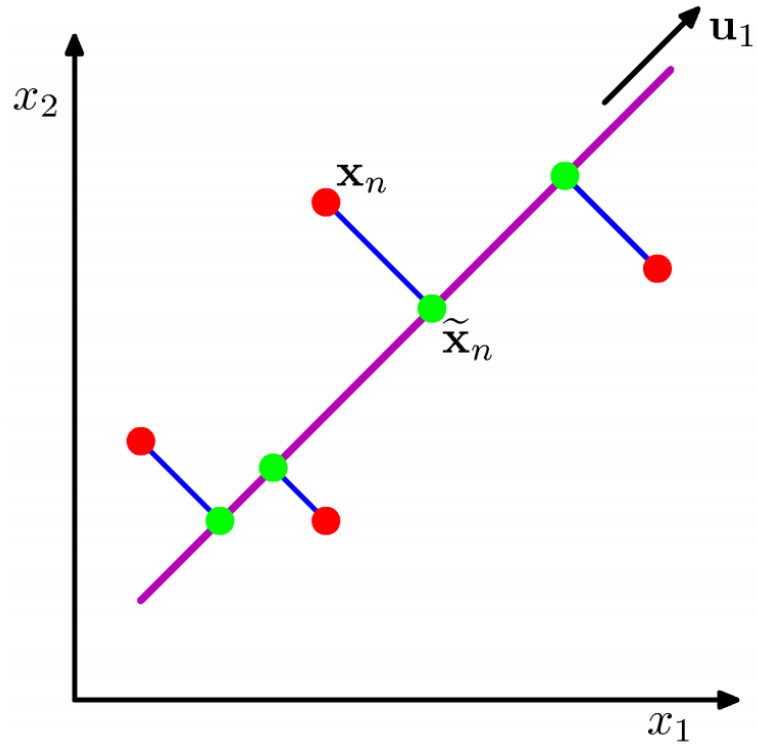
- Generalize to $\ell_q$ norm: $(\|\beta\|_q)^q = |\beta_1|^q + |\beta_2|^q$.
- $\mathcal{F} = \{f(x) = \beta_1 x_1 + \beta_2 x_2\}$.
- Contours of $\|\beta\|_q^q = |\beta_1|^q + |\beta_2|^q$:



# Dimension Reduction: Principal Component Analysis

Goal

- Project data onto a space with dimensional $M < D$
- Maximize the variance of the projected data



# Principal Component Analysis

Goal:

- Maximum variance criterion corresponds to a Rayleigh quotient
- PCA boils down to an eigenvalue problem on the *centered* covariance matrix $\hat{\Sigma}$ of the dataset, that is the principal components $w_1, \ldots, w_d$ are the eigenvectors of $\hat{\Sigma}$ (assuming $n > d$)
- Computational complexity: $O(ndc)$ in time with a *Singular Value Decomposition* (SVD; see `eigs` in Matlab/Octave), with $n$ the number of points, $d$ the dimension, $c$ the number of principal components retained; stochastic approximation version for nonstationary/large-scale datasets.

# Principal Component Analysis

$$\text{Empirical mean} \quad \bar{x} = \frac{1}{N} \sum_{j=1}^{d} x_j$$

$$\text{Empirical covariance} \quad \hat{\Sigma} = \frac{1}{N} \sum_{j=1}^{d} (x_j - \bar{x})(x_j - \bar{x})^T$$

Projection along the direction $w$

- $\text{Proj}_w(x_j) = w^T x_j$, for all $j = 1, \ldots, N$
- $\text{Proj}_w(\bar{x}) = w^T \bar{x}$

# Principal Component Analysis

Projection along the direction $w$

- $\text{Proj}_w(x_j) = w^T x_j$, for all $j = 1, \ldots, N$
- $\text{Proj}_w(\bar{x}) = w^T \bar{x}$

Variance of $\text{Proj}_w(x_j)$

$$\frac{1}{N} \sum_{j=1}^{N} (w^T x_j - w^T \bar{x})^2 = w^T \hat{\Sigma} w \ .$$

# First Principal Component

Projection along the direction $w$

- $\text{Proj}_w(x_j) = w^T x_j$, for all $j = 1, \ldots, N$
- $\text{Proj}_w(\bar{x}) = w^T \bar{x}$

Variance of $\text{Proj}_w(x_j)$

$$\frac{1}{N} \sum_{j=1}^{N} (w^T x_j - w^T \bar{x})^2 = w^T \hat{\Sigma} w \ .$$

# How to compute the top pair of eigenvalue and eigenvector

For a matrix $A$, the Power Iteration algorithm returns the top pair of eigenvalue $\lambda$ and eigenvector $v$ of the matrix $A$.

---

**Algorithm 1** Power Iteration Algorithm

---

**initialization** $v_0$ random vector, and large number $N$ .
**repeat** for $k = 1, 2, 3, \cdots, N$

- Perform update $z_k = Av_{k-1}$ ,
- Perform update $v_k = \frac{z_k}{\|z_k\|_2}, \lambda_k = v_k^T A v_k$ .

**until** the stopping criterion is satisfied.

---

# Variance along a direction and Rayleigh quotients

PCA seeks for directions $w_1, \ldots, w_c$ such that

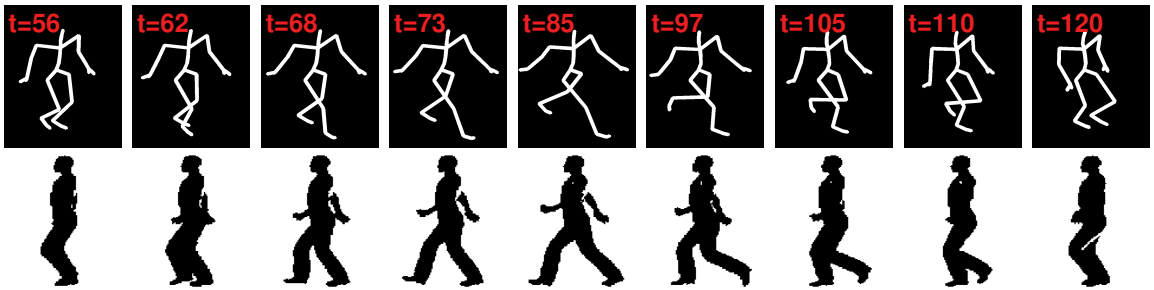
$$
\begin{aligned}
w_j &= \text{argmax}_{w \in \mathbb{R}^d; w_j \perp \{w_1, \ldots, w_{j-1}\}} \text{Var}_x \left\{ \frac{w^T x}{w^T w} \right\} \\
&= \text{argmax}_{w \in \mathbb{R}^d; w_j \perp \{w_1, \ldots, w_{j-1}\}} \frac{1}{m} \sum_{i=1}^{m} \frac{(w^T x_i)^2}{w^T w} \\
&= \text{argmax}_{w \in \mathbb{R}^d; w_j \perp \{w_1, \ldots, w_{j-1}\}} \underbrace{\frac{w^T \hat{\Sigma} w}{w^T w}}_{\text{Rayleigh quotient}} .
\end{aligned}
$$

Principal components $w_1, \ldots, w_c$ are the first $c$ eigenvectors of $\hat{\Sigma}$.

# Low-dimensional representation with PCA

- Walking sequence of length 400 (containing about 3 walking cycles) obtained from the CMU Mocap database
- Data: silhouette images taken at a side view

Human body pose representation (Kim & Pavlovic, 2008). Selected skeleton and silhouette images for a half walking cycle.



# Low-dimensional representation with PCA

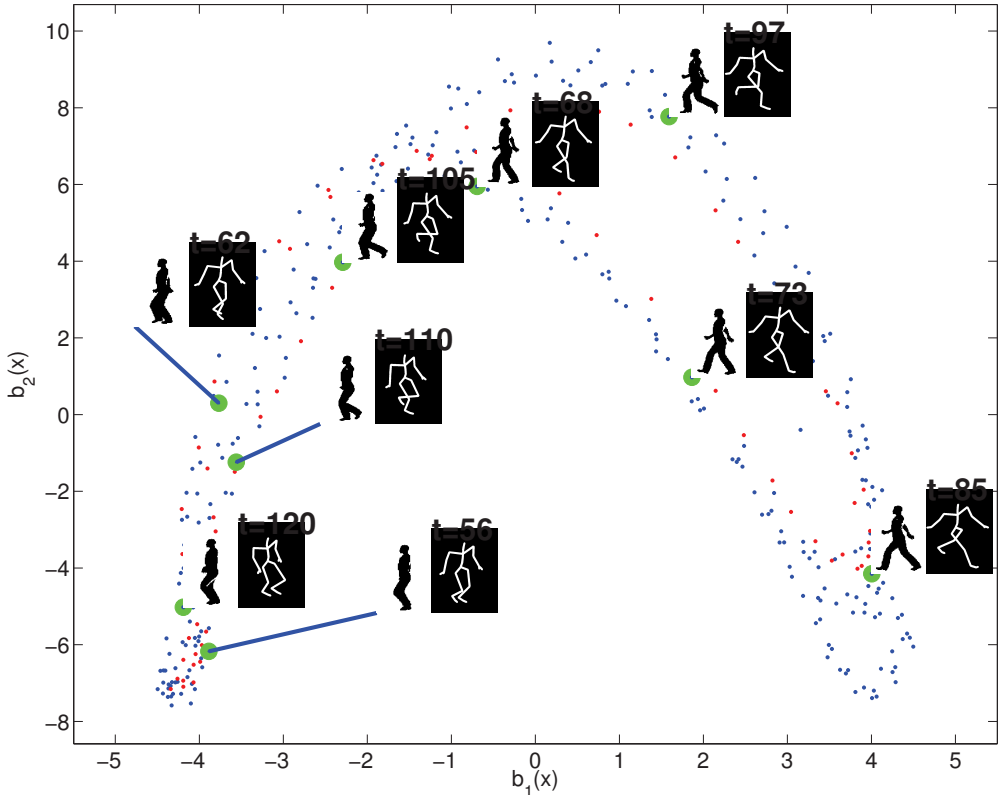



Figure: Central subspaces for silhouette images from walking motion

# Low-dimensional representation with PCA

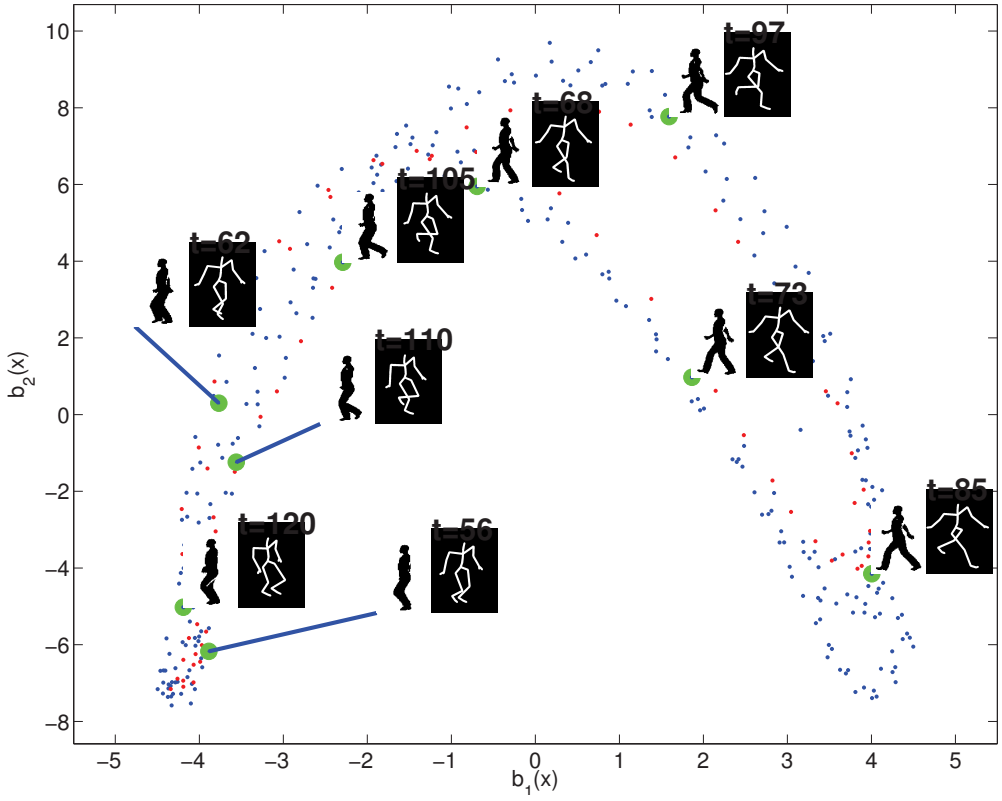



Figure: Central subspaces for silhouette images from walking motion

# Super-resolution with PCA (Kim et al., 2005)

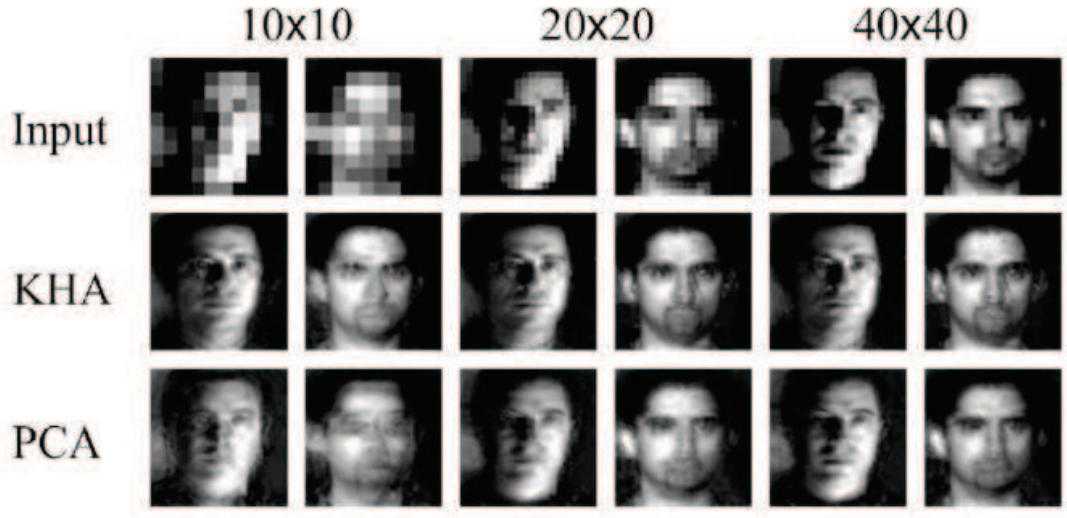



Figure: Super-resolution from low-resolution images of faces

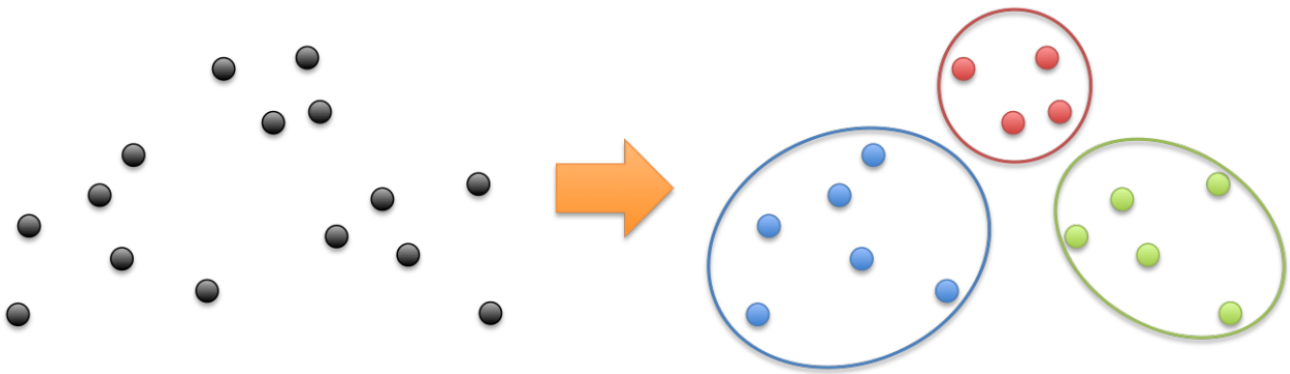
# Applications

- Image denoising (digits, faces, etc.)
- Visualization of bioinformatics data (strings, proteins, etc.)
- Dimension-reduction of high-dimensional features (appearance, interest points, etc.)

# Clustering

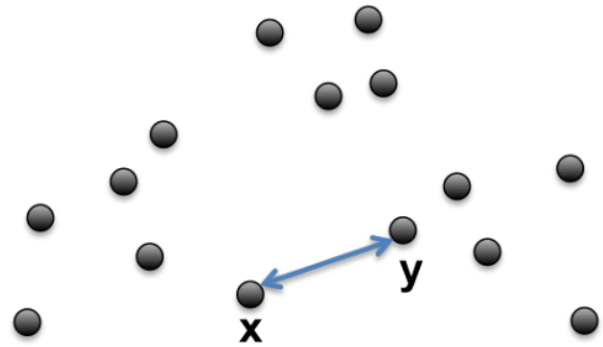
Goal is to group "similar" instances together.

- Given data points $x_i \in \mathbf{R}^d$, $i = 1, \ldots, n$.
- But no labels  unsupervised learning
- Useful for exploratory data analysis



# Clustering

Need measure of similarity (or distance) between $x$ and $y$.




Popular distance metrics:

- Squared Euclidean distance $d(x, y) = \|x - y\|_2^2$
- Cosine similarity $d(x, y) = (x^T y)/(\|x\|\|y\|)$
- Manhattan distance $d(x, y) = \|x - y\|_1$

Clustering results are crucially dependent on the distance metric.

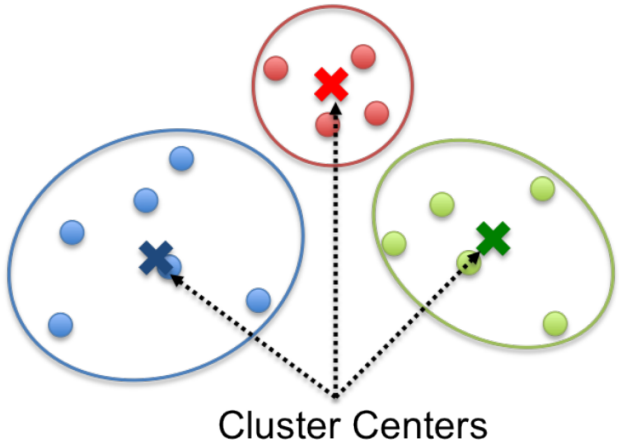
# Clustering

Find $k$ clusters that minimizes the objective:

$$J = \sum_{i=1}^{k} \sum_{x \in \mathcal{C}_i} \|x - m_i\|_2^2$$

where

- $\mathcal{C}_i$ set of points in cluster $i$
- $m_i$ mean of cluster $i$
- Objective is non-convex and problem is NP-hard in general



# Unsupervised Learning: Clustering

**Input:** data points $\boldsymbol{x} \in \mathbb{R}^d$, number of clusters $k$
**Output:** cluster assignment $\mathcal{C}_i$ of data points, $i = 1, 2, \ldots, k$

1: Randomly partition the data into $k$ clusters
2: **while** not converged **do**
3: Compute mean of each cluster $i$

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\boldsymbol{x} \in \mathcal{C}_i} \boldsymbol{x}$$

4: For each $\boldsymbol{x}$, find its new cluster index:

$$\pi(\boldsymbol{x}) = \arg \min_{1 \le i \le k} \|\boldsymbol{x} - \mathbf{m}_i\|_2^2$$

5: Update clusters:

$$\mathcal{C}_i = \{\boldsymbol{x} | \pi(\boldsymbol{x}) = i\}$$

6: **end while**

# Clustering

Let the objective at the $t$-th iteration be

$$J^{(t)} = \sum_{i=1}^{k} \sum_{x \in \mathcal{C}_i^{(t)}} \|x - m_i^{(t)}\|_2^2 .$$

We have

$$\begin{aligned} J^{(t)} &\geq \sum_{i=1}^{k} \sum_{x \in \mathcal{C}_i^{(t)}} \|x - m_{\pi(x)}^{(t)}\|_2^2 \\ &= \sum_{i=1}^{k} \sum_{x \in \mathcal{C}_{i+1}^{(t)}} \|x - m_i^{(t)}\|_2^2 \\ &\geq \sum_{i=1}^{k} \sum_{x \in \mathcal{C}_{i+1}^{(t)}} \|x - m_i^{(t+1)}\|_2^2 \\ &= J^{(t+1)} \end{aligned}$$

# Clustering

- Each step decreases the objective - convergence guarantee
- Convergence to a stationary point, not necessarily the global minimum

# Clustering

**Input:** data points $\boldsymbol{x} \in \mathbb{R}^d$, number of clusters $k$
**Output:** cluster assignment $\mathcal{C}_i$ of data points, $i = 1, 2, \ldots, k$

1: Initialize means $\mathbf{m}_i$ and $n_i = 0$, $i = 1, 2, \ldots, k$
2: **while** not converged **do**
3: Pick a data point $\boldsymbol{x}$ and determine cluster $\pi(\boldsymbol{x})$

$$\pi(\boldsymbol{x}) = \arg \min_{1 \leq i \leq k} \|\boldsymbol{x} - \mathbf{m}_i\|_2^2$$

4: Update mean $\mathbf{m}_{\pi(\boldsymbol{x})}$

$$n_{\pi(\boldsymbol{x})} = n_{\pi(\boldsymbol{x})} + 1 \quad \text{and} \quad \mathbf{m}_{\pi(\boldsymbol{x})} = \mathbf{m}_{\pi(\boldsymbol{x})} + \frac{1}{n_{\pi(\boldsymbol{x})}}(\boldsymbol{x} - \mathbf{m}_{\pi(\boldsymbol{x})})$$

5: **end while**