Foundations of Machine Learning
CentraleSupélec — Fall 2017

7. Regularized linear regression

Chloé-Agathe Azencott
Centre for Computational Biology, Mines ParisTech
chloe-agathe.azencott@mines-paristech.fr

Practical matters

- **Class representatives** (délégués de classe)

HW04

Assume p random variables $X_1, \ldots, X_p$, conditionally independent given Y. Y is a discrete random variable that can take one of K values $\{y_1, \ldots, y_K\}$, corresponding to K classes.

We suppose that each X_j is distributed normally:

$$p(X_j = u | Y = y_k) = \frac{1}{\sigma_{jk}\sqrt{2\pi}} \exp\left(\frac{-(u - \mu_{jk})^2}{2\sigma_{jk}^2}\right).$$

Let us assume that the variance is independent of the class k and the feature j, i.e. $\sigma_{jk} = \sigma$.

We observe n datapoints $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\}$ and their labels $\{y^1, \ldots, y^n\}$. Derive the maximum likelihood estimator for μ_{jk}.

What is known? What are the parameters to be estimated?

HW04

Assume p random variables $X_1, \ldots, X_p$, conditionally independent given Y. Y is a discrete random variable that can take one of K values $\{y_1, \ldots, y_K\}$, corresponding to K classes.

We suppose that each X_j is distributed normally:

variable

parameters to be determined

$$p(X_j = u | Y = y_k) = \frac{1}{\sigma_{jk}\sqrt{2\pi}} \exp\left(\frac{-(u - \mu_{jk})^2}{2\sigma_{jk}^2}\right).$$

Let us assume that the variance is independent of the class k and the feature j, i.e. $\sigma_{jk} = \sigma$.

We observe n datapoints $\boxed{\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\}}$ and their labels $\boxed{\{y^1, \ldots, y^n\}}$. Derive the maximum likelihood estimator for μ_{jk}.

given

Assume p random variables $X_1, \ldots, X_p$, conditionally independent given Y. Y is a discrete random variable that can take one of K values $\{y_1, \ldots, y_K\}$, corresponding to K classes.

We suppose that each X_j is distributed normally:

variable

parameters to be determined

$$p(X_j = u | Y = y_k) = \frac{1}{\sigma_{jk}\sqrt{2\pi}} \exp\left(\frac{-(u - \mu_{jk})^2}{2\sigma_{jk}^2}\right).$$

Let us assume that the variance is independent of the class k and the feature j, i.e. $\sigma_{jk} = \sigma$.

We observe n datapoints $\boxed{\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\}}$ and their labels $\boxed{\{y^1, \ldots, y^n\}}$. Derive the maximum likelihood estimator for μ_{jk}.

given

Maximum likelihood estimation

- **Find θ such that $\mathcal{X}$ is the most likely to be drawn.**
- **Likelihood** of θ given the i.i.d. sample $\mathcal{X}$:
$$\ell(\theta|\mathcal{X}) = p(\mathcal{X}|\theta) = p(\boldsymbol{x}^1|\theta)p(\boldsymbol{x}^2|\theta)\ldots p(\boldsymbol{x}^n|\theta)$$
- **Log likelihood:**
$$\mathcal{L}(\theta|\mathcal{X}) = \log \ell(\theta|\mathcal{X}) = \log p(\boldsymbol{x}^1|\theta) + \cdots + \log p(\boldsymbol{x}^n|\theta)$$
- **Maximum likelihood estimation (MLE):**
$$\theta^* = \arg\max_\theta \mathcal{L}(\theta|\mathcal{X})$$

Assume p random variables $X_1, \ldots, X_p$, conditionally independent given Y. Y is a discrete random variable that can take one of K values $\{y_1, \ldots, y_K\}$, corresponding to K classes.

We suppose that each X_j is distributed normally:

variable

parameters to be determined

$$p(X_j = u | Y = y_k) = \frac{1}{\sigma_{jk}\sqrt{2\pi}} \exp\left(\frac{-(u - \mu_{jk})^2}{2\sigma_{jk}^2}\right).$$

Let us assume that the variance is independent of the class k and the feature j, i.e. $\sigma_{jk} = \sigma$.

We observe n datapoints $\boxed{\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\}}$ and their labels $\boxed{\{y^1, \ldots, y^n\}}$. Derive the maximum likelihood estimator for μ_{jk}.

given

Maximum likelihood estimation

- Find $\boxed{\theta}$ such that $\boxed{\mathcal{X}}$ is the most likely to be drawn.
- Likelihood of θ given the i.i.d. sample $\mathcal{X}$:

$$\ell(\theta|\mathcal{X}) = p(\mathcal{X}|\theta) = p(\boldsymbol{x}^1|\theta)p(\boldsymbol{x}^2|\theta)\ldots p(\boldsymbol{x}^n|\theta)$$

- Log likelihood:

$$\mathcal{L}(\theta|\mathcal{X}) = \log \ell(\theta|\mathcal{X}) = \log p(\boldsymbol{x}^1|\theta) + \cdots + \log p(\boldsymbol{x}^n|\theta)$$

- Maximum likelihood estimation (MLE):

$$\theta^* = \arg\max_\theta \mathcal{L}(\theta|\mathcal{X})$$

Assume p random variables $X_1, \ldots, X_p$, conditionally independent given Y. Y is a discrete random variable that can take one of K values $\{y_1, \ldots, y_K\}$, corresponding to K classes.

We suppose that each X_j is distributed normally:

variable

parameters to be determined

$$p(X_j = u | Y = y_k) = \frac{1}{\sigma_{jk}\sqrt{2\pi}} \exp\left(\frac{-(u - \mu_{jk})^2}{2\sigma_{jk}^2}\right).$$

Let us assume that the variance is independent of the class k and the feature j, i.e. $\sigma_{jk} = \sigma$.

We observe n datapoints $\boxed{\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\}}$ and their labels $\boxed{\{y^1, \ldots, y^n\}}$. Derive the maximum likelihood estimator for μ_{jk}.

given

For feature j, class k

Maximum likelihood estimation

$$\boxed{\mu_{jk}^* = \arg\max_{\mu_{jk}} \sum_{i=1}^{n} \log p(X_j = x_j^i | \mu_{jk}, \sigma)}$$ when $Y=y_k$

- **Log likelihood:**
  $\mathcal{L}(\theta|\mathcal{X}) = \log \ell(\theta|\mathcal{X}) = \log p(\boldsymbol{x}^1|\theta) + \cdots + \log p(\boldsymbol{x}^n|\theta)$
- **Maximum likelihood estimation (MLE):**
  $\theta^* = \arg\max_{\theta} \mathcal{L}(\theta|\mathcal{X})$

Assume p random variables $X_1, \ldots, X_p$, conditionally independent given Y. Y is a discrete random variable that can take one of K values $\{y_1, \ldots, y_K\}$, corresponding to K classes.

We suppose that each X_j is distributed normally:

variable

parameters to be determined

$$p(X_j = u | Y = y_k) = \frac{1}{\sigma_{jk}\sqrt{2\pi}} \exp\left(\frac{-(u - \mu_{jk})^2}{2\sigma_{jk}^2}\right).$$

Let us assume that the variance is independent of the class k and the feature j, i.e. $\sigma_{jk} = \sigma$.

We observe n datapoints $\boxed{\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\}}$ and their labels $\boxed{\{y^1, \ldots, y^n\}}$. Derive the maximum likelihood estimator for μ_{jk}.

given

For feature j, class k

$$\boxed{\mu_{jk}^* = \arg\max_{\mu_{jk}} \sum_{i=1}^{n} \log p(X_j = x_j^i | \mu_{jk}, \sigma)}$$ when $Y=y_k$

$$= \frac{1}{\sigma_{jk}\sqrt{2\pi}} \exp\left(\frac{-(u - \mu_{jk})^2}{2\sigma_{jk}^2}\right)$$

Assume p random variables $X_1, \ldots, X_p$, conditionally independent given Y. Y is a discrete random variable that can take one of K values $\{y_1, \ldots, y_K\}$, corresponding to K classes.

We suppose that each X_j is distributed normally:

variable

parameters to be determined

$$p(X_j = u | Y = y_k) = \frac{1}{\sigma_{jk}\sqrt{2\pi}} \exp\left(\frac{-(u - \mu_{jk})^2}{2\sigma_{jk}^2}\right).$$

Let us assume that the variance is independent of the class k and the feature j, i.e. $\sigma_{jk} = \sigma$.

We observe n datapoints $\boxed{\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\}}$ and their labels $\boxed{\{y^1, \ldots, y^n\}}$. Derive the maximum likelihood estimator for μ_{jk}.

given

For feature j, class k

$$\boxed{\mu_{jk}^* = \arg\max_{\mu_{jk}} \sum_{i=1}^{n} \log \boxed{p(X_j = x_j^i | \mu_{jk}, \sigma)}}$$

when $Y=y_k$

I_{ik}

$$= \frac{1}{\sigma_{jk}\sqrt{2\pi}} \exp\left(\frac{-(x_j^i - \mu_{jk})^2}{2\sigma_{jk}^2}\right)$$

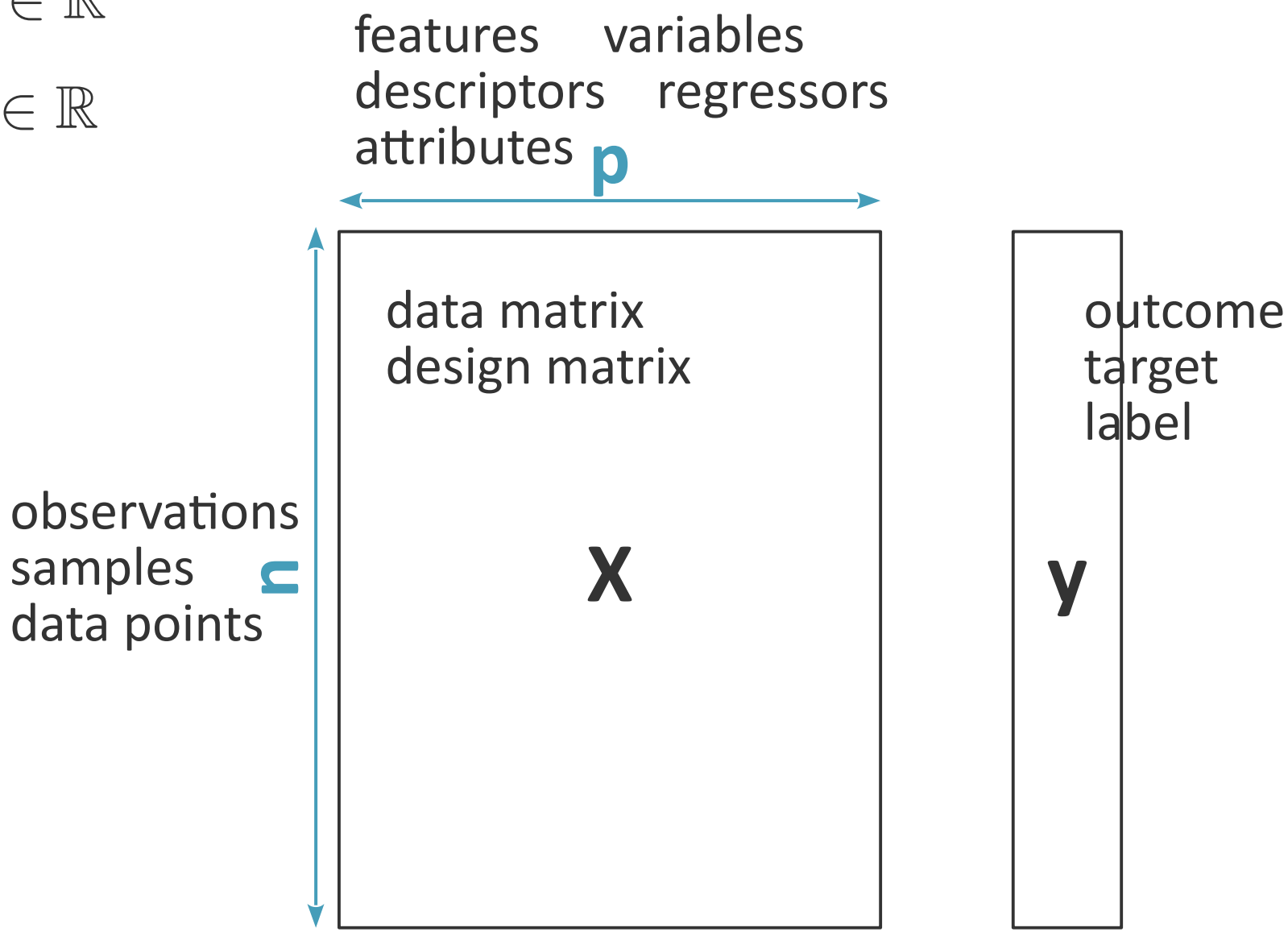
Learning objectives

- Understand **regularization** as a means to control model complexity.
- Define **Lasso**, **ridge regression**, **elastic net.**
- Understand the role of the **l1 and l2 norms** in regularization
- Interpret **solution paths** for Lasso and ridge regression.

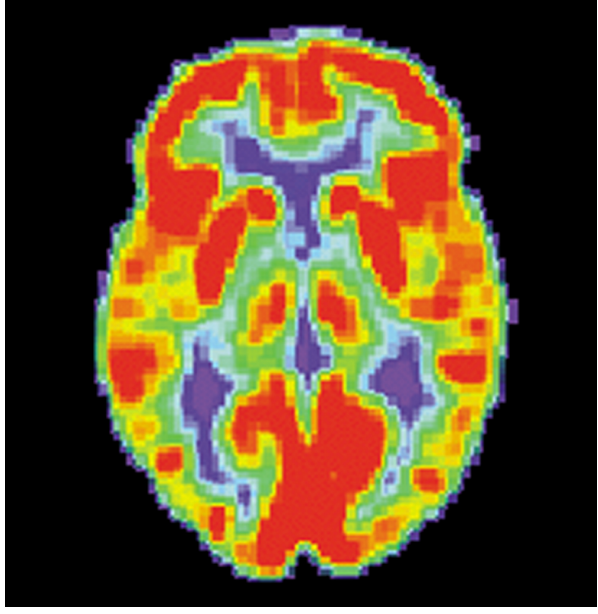
Regression setting

$x_j^i \in \mathbb{R}$

$y^i \in \mathbb{R}$



Large p, small n



- **neuroimaging**

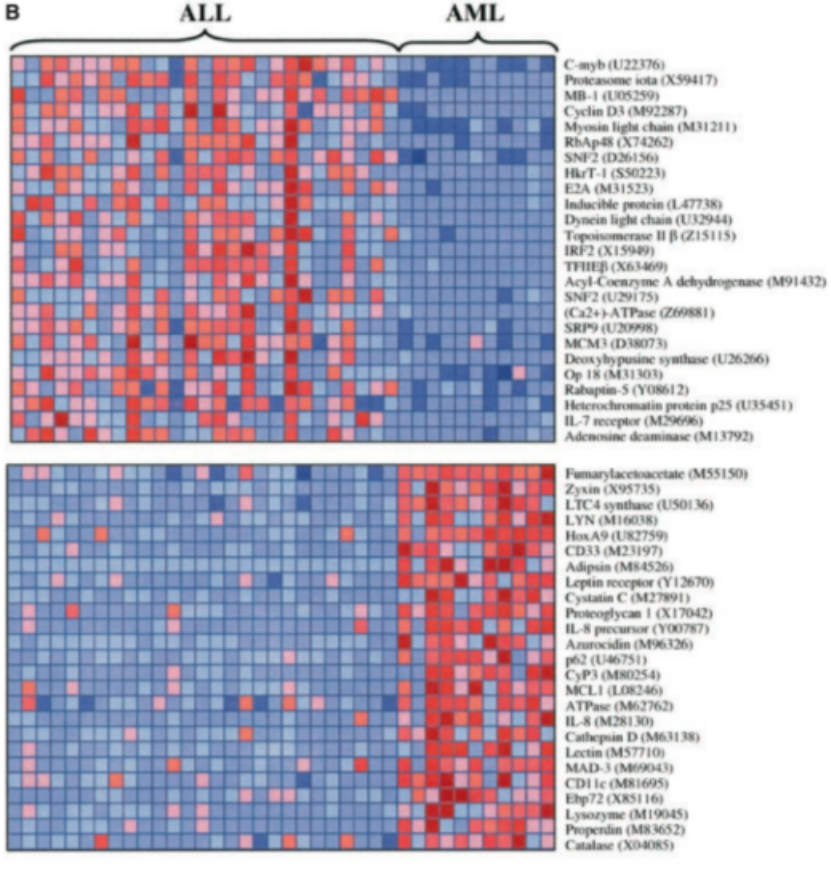
  thousands of brain regions / pixels / voxels

  much fewer patients

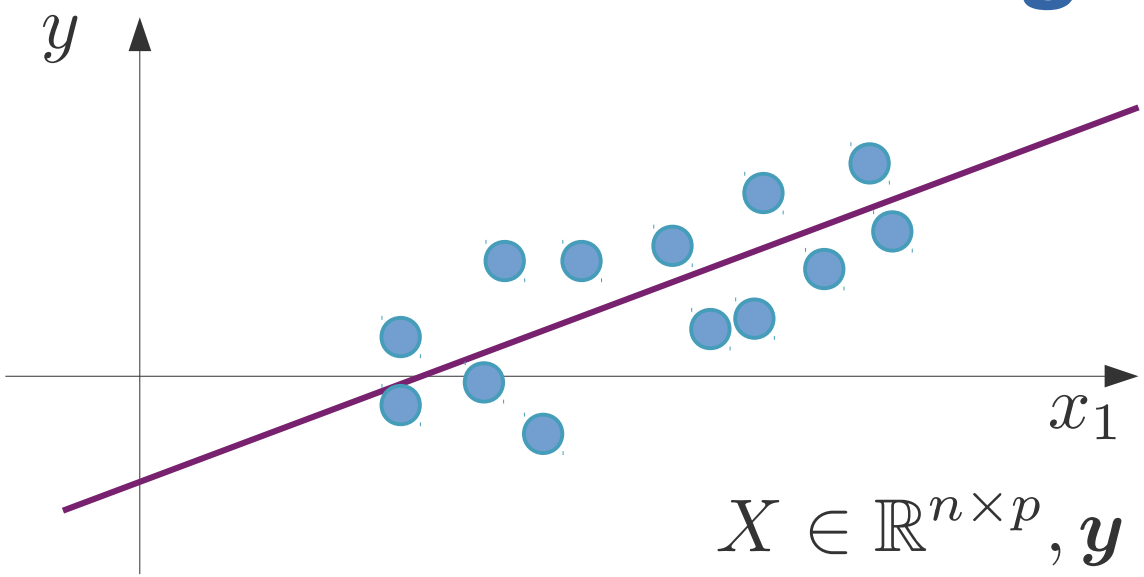
- **genetics and genomics**

  thousands of genes, millions of SNPs..

  usually, at best thousands of patients



Linear regression



$$\boldsymbol{x} \in \mathbb{R}^p$$

$$f(\boldsymbol{x}|\boldsymbol{\beta}) = \sum_{j=1}^{p} \beta_j x_j + \beta_0$$

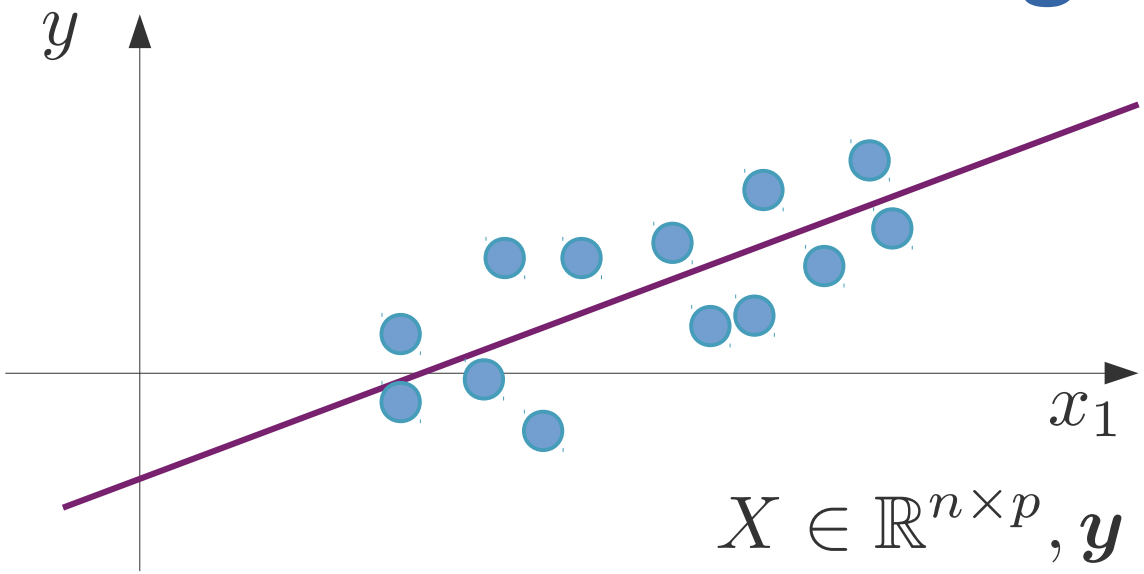
$$X \in \mathbb{R}^{n \times p}, \boldsymbol{y} \in \mathbb{R}^n$$

Least-squares fit (equivalent to MLE under the assumption of Gaussian noise):

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} (\boldsymbol{y} - X\boldsymbol{\beta})^\top (\boldsymbol{y} - X\boldsymbol{\beta}) = (X^\top X)^{-1} X^\top \boldsymbol{y}$$

Solution uniquely defined when ?

Linear regression



$$\boldsymbol{x} \in \mathbb{R}^p$$

$$f(\boldsymbol{x}|\boldsymbol{\beta}) = \sum_{j=1}^{p} \beta_j x_j + \beta_0$$

$$X \in \mathbb{R}^{n \times p}, \boldsymbol{y} \in \mathbb{R}^n$$

Least-squares fit (equivalent to MLE under the assumption of Gaussian noise):

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} (\boldsymbol{y} - X\boldsymbol{\beta})^\top (\boldsymbol{y} - X\boldsymbol{\beta}) = (X^\top X)^{-1} X^\top \boldsymbol{y}$$

Solution uniquely defined when $X^\top X$ **invertible.**

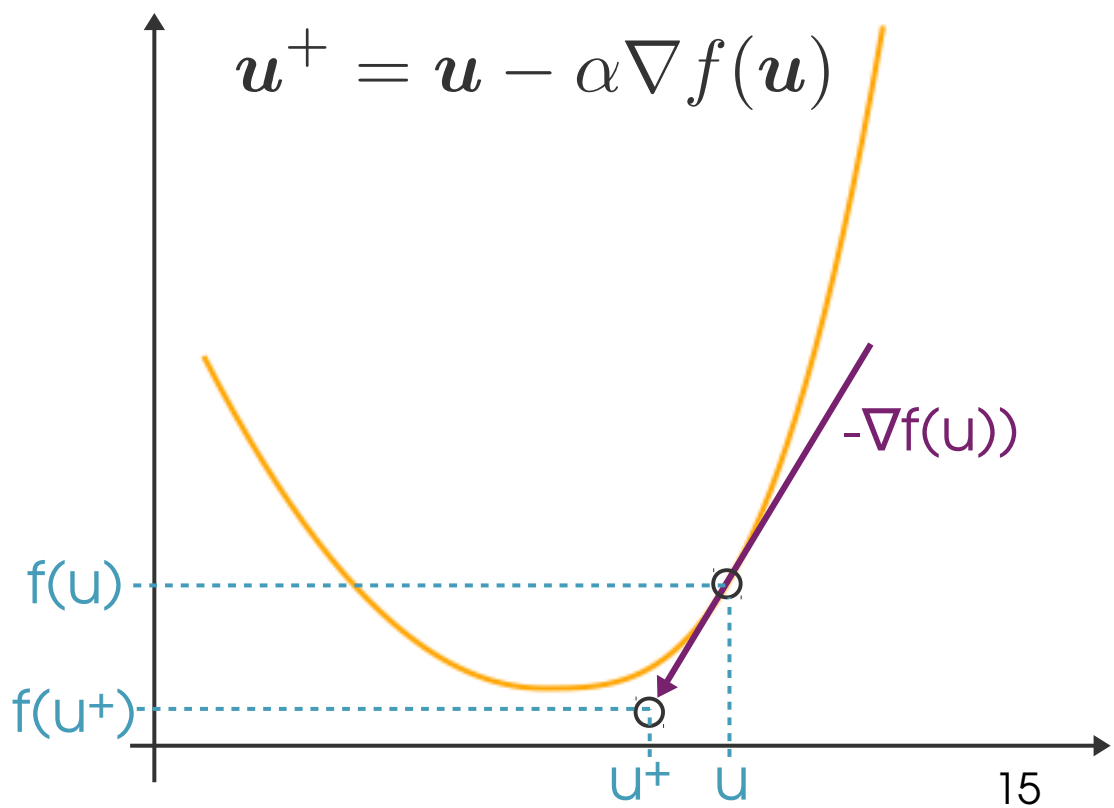
When X^TX not inversible

$$(X^\top X)\hat{\beta} = X^\top y$$

- Pseudo-inverse
- Linear system of p equations:

Numerical methods

- **Gaussian elimination**
- **LU decomposition**
- **Gradient descent**



Properties of the least-squares fit estimate

- **Unbiased** $\mathbb{E}[\hat{\boldsymbol{\beta}}] = \boldsymbol{\beta}$
- **Explicit form** $\hat{\boldsymbol{\beta}} = (X^\top X)^{-1} X^\top \boldsymbol{y}$
- **Computational time** ?

Properties of the least-squares fit estimate

- **Unbiased** $\mathbb{E}[\hat{\boldsymbol{\beta}}] = \boldsymbol{\beta}$
- **Explicit form** $\hat{\boldsymbol{\beta}} = (X^\top X)^{-1} X^\top \boldsymbol{y}$
- **Computational time:** $\mathcal{O}(p^3 + np)$

computation of $(X^\top X)^{-1} = \mathcal{O}(p^3)$

computation of $X^\top \boldsymbol{y} = \mathcal{O}(np)$

When p is large

- **p > n:** $X^\top X$?

When p is large

- **p > n:** $X^\top X$ not invertible

When p is large

- **p > n:** $X^\top X$ not invertible
  - Use a pseudo-inverse M $(X^\top X)M(X^\top X) = (X^\top X)$
  - Multiple possible solutions
  - High variance of the estimator.

When p is large

- **p > n:** $X^\top X$ not invertible
  - Use a pseudo-inverse M $\quad (X^\top X)M(X^\top X) = (X^\top X)$
  - Multiple possible solutions
  - High variance of the estimator.
- **Multicollinearity:**

When p is large

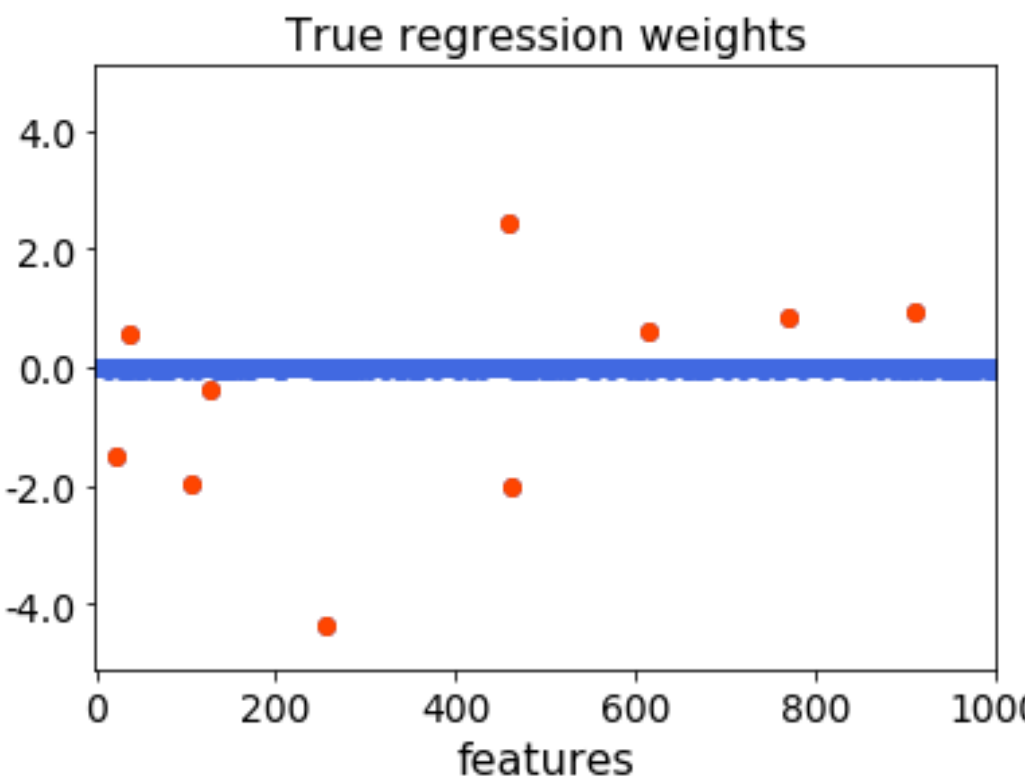
- **p > n:** $X^\top X$ not invertible
  - Use a pseudo-inverse M $(X^\top X)M(X^\top X) = (X^\top X)$
  - Multiple possible solutions
  - High variance of the estimator.
- **Multicollinearity:** $X^\top X$ not invertible
- Large p **reduces interpretability** of the model

Would prefer a small subset of features with strong effects (= large coefficients).

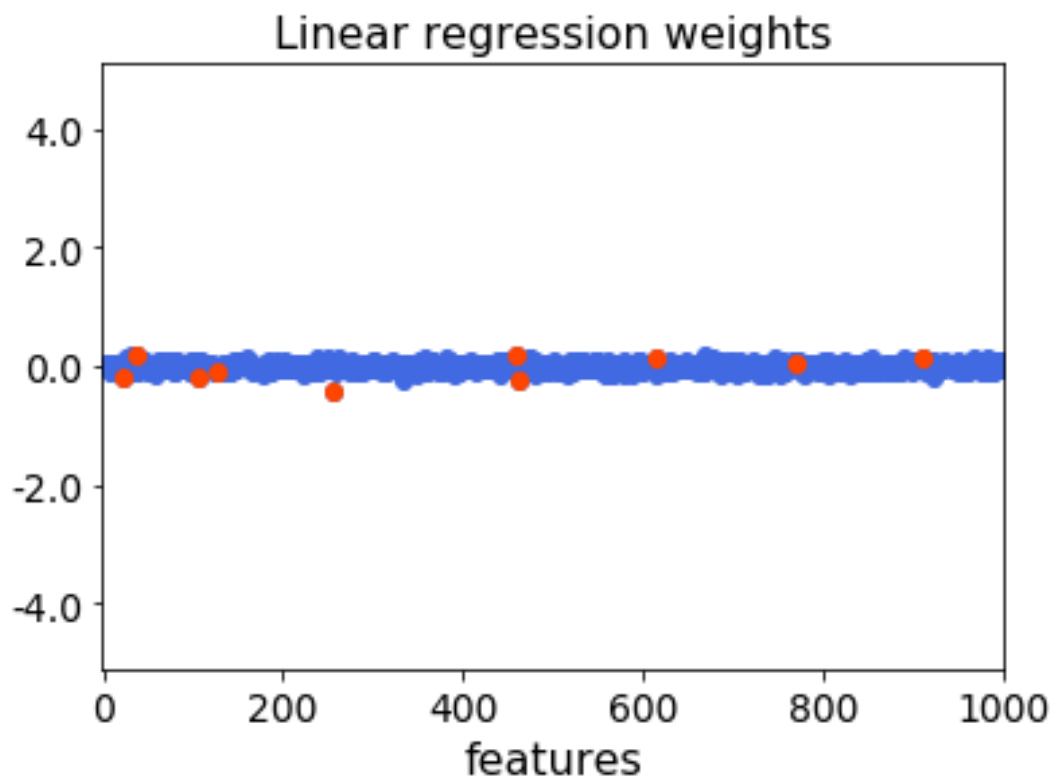
Linear regression when p >> n

Simulated data: p=1000, n=100, 10 causal features

True coefficients



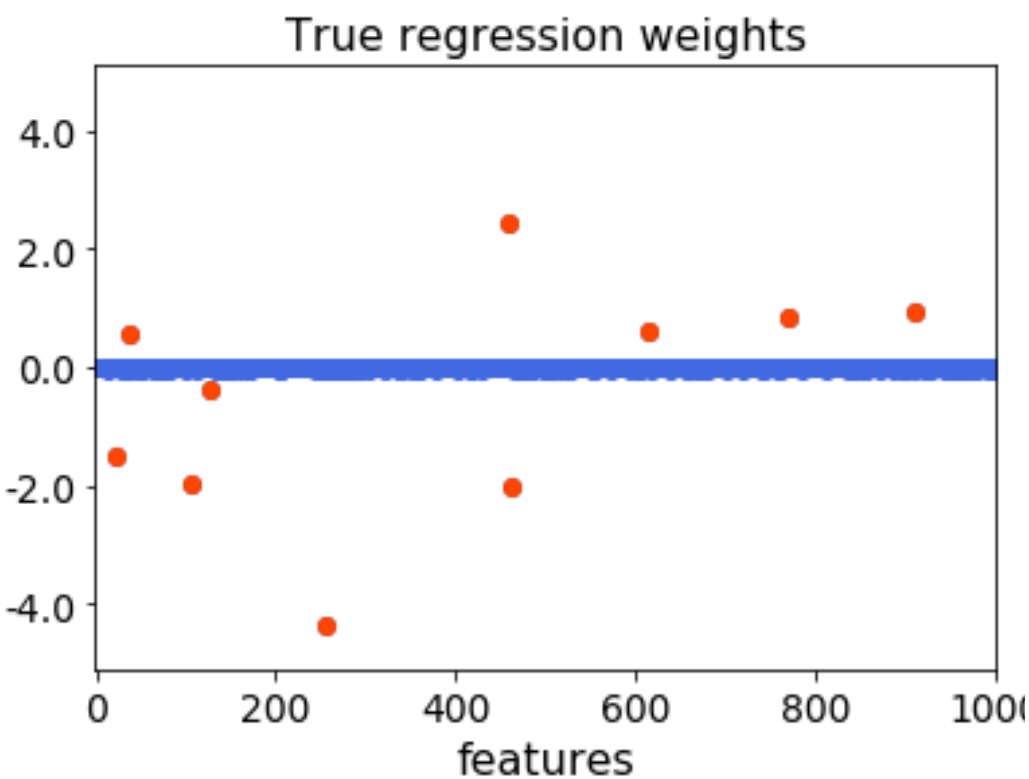
Predicted coefficients



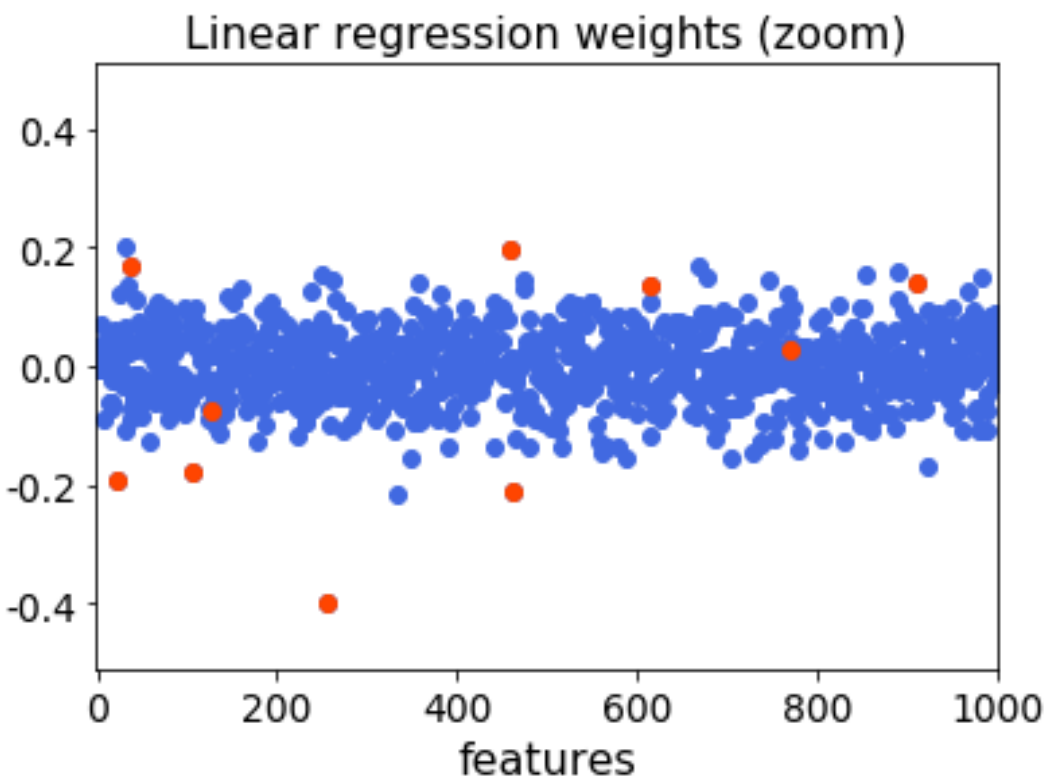
Linear regression when p >> n

Simulated data: p=1000, n=100, 10 causal features

True coefficients



Predicted coefficients



Regularization

Regularization

- Minimize

  Prediction error + **λ** penalty on model complexity

- **Biased estimator** when $\lambda \neq 0$.
- Trade bias for a smaller variance.
- λ can be set by cross-validation.

- Simpler model ≈ fewer parameters

  → **shrinkage:** drive the coefficients of the parameters towards 0.

Ridge regression

Ridge regression

- **Sum-of-squares penalty**

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} ||y - X\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2$$

- **Ridge regression estimator:** ?

Ridge regression

- **Sum-of-squares penalty**

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} ||y - X\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_2^2$$

- **Ridge regression estimator:**

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = (X^\top X + \lambda I)^{-1} X^\top \boldsymbol{y}$$

$$\text{if } (X^\top X + \lambda I) \text{ invertible.}$$

Ridge regression

- **Sum-of-squares penalty**

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} ||y - X\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_2^2$$

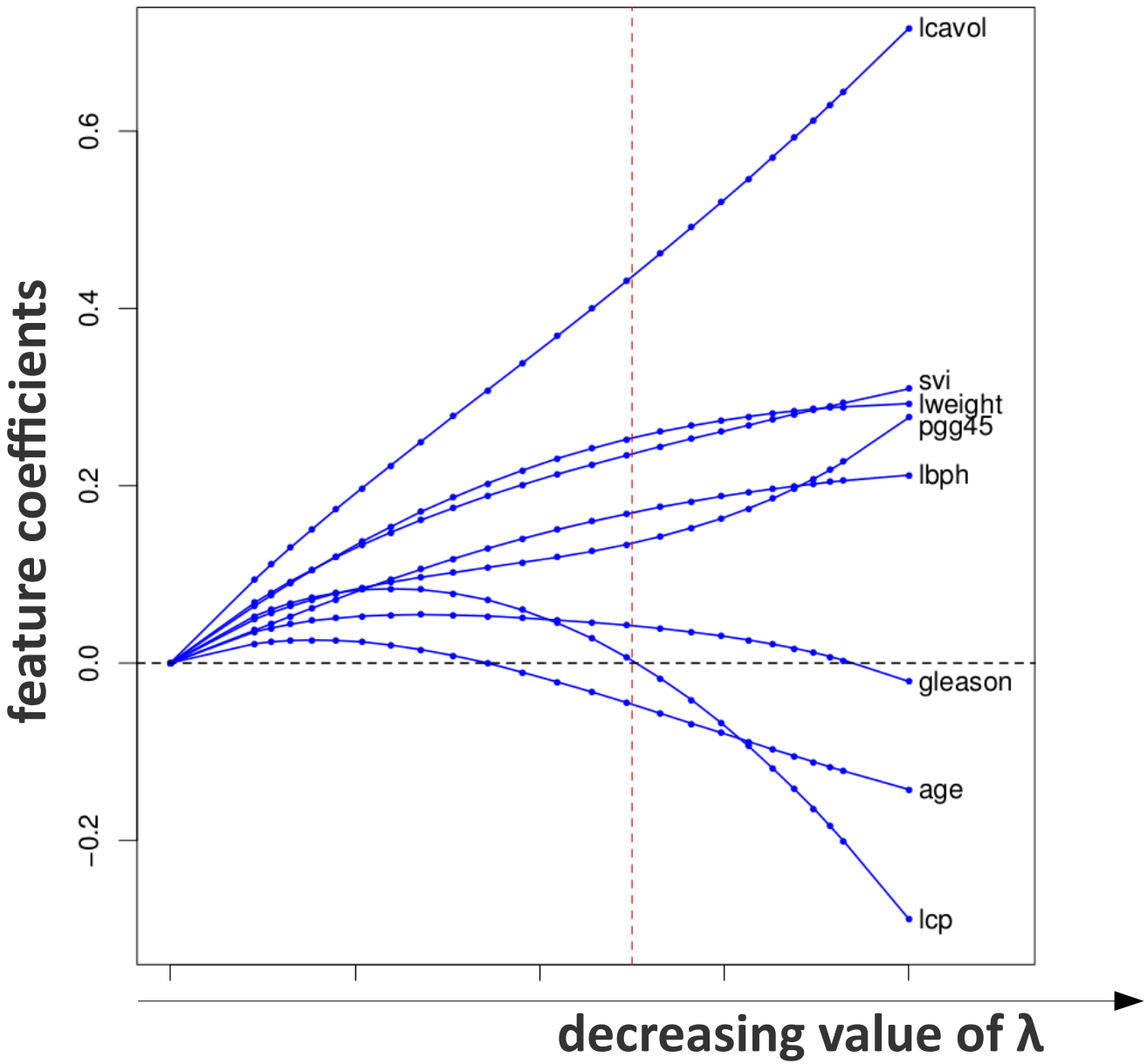
- **Ridge regression estimator:**

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = (X^\top X + \lambda I)^{-1} X^\top \boldsymbol{y}$$

$$\text{if } (X^\top X + \lambda I) \text{ invertible.}$$

Always!

Ridge regression solution path



Standardization

- **Multiply x_j by a constant:**
  - For **standard linear regression**
  - For **ridge regression**

Standardization

- **Multiply x_j by a constant:**
  - For **standard linear regression:**

$$\hat{\beta}_j \rightarrow \frac{1}{c}\hat{\beta}_j$$

  - For **ridge regression:** ?

Standardization

- **Multiply x_j by a constant:**
  - For **standard linear regression:**

$$\hat{\beta}_j \to \frac{1}{c}\hat{\beta}_j$$

  - For **ridge regression:**

    Not so clear, because of the penalization term $\lambda\beta_j^2$
- Need to **standardize** the features

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

average value of x_j

Ridge regression

- **Grouped selection:**
  - correlated variables get similar weights
  - identical variables get identical weights
- Ridge regression shrinks coefficients towards 0 but does not result in a **sparse model**.
- **Sparsity:**
  - many coefficients get a weight of 0
  - they can be eliminated from the model.

Lasso

Lasso

$$||\boldsymbol{\beta}||_1 = \sum_{j=1}^{p} |\beta_p|$$

- **L1 penalty**

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} ||y - X\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_1$$

- aka **basis pursuit** (signal processing)
- no closed-form solution
- Equivalent to

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} ||y - X\boldsymbol{\beta}||_2^2 \quad \text{s.t.} \quad ||\boldsymbol{\beta}||_1 \leq t$$

for a unique one-to-one match between t and λ.

Optimization problem: ?

Lasso

$$||\boldsymbol{\beta}||_1 = \sum_{j=1}^{p} |\beta_p|$$

- **L1 penalty**

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} ||y - X\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_1$$

- aka **basis pursuit** (signal processing)
- no closed-form solution
- Equivalent to

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} ||y - X\boldsymbol{\beta}||_2^2 \quad \text{s.t.} \quad ||\boldsymbol{\beta}||_1 \leq t$$

for a unique one-to-one match between t and λ.

QP: maximize a quadratic form under linear constraints.

Geometric interpretation

Minimize f(β) under the constraint g(β) ≤ 0

$$g(\boldsymbol{\beta}) = ||\boldsymbol{\beta}||_1 - t$$

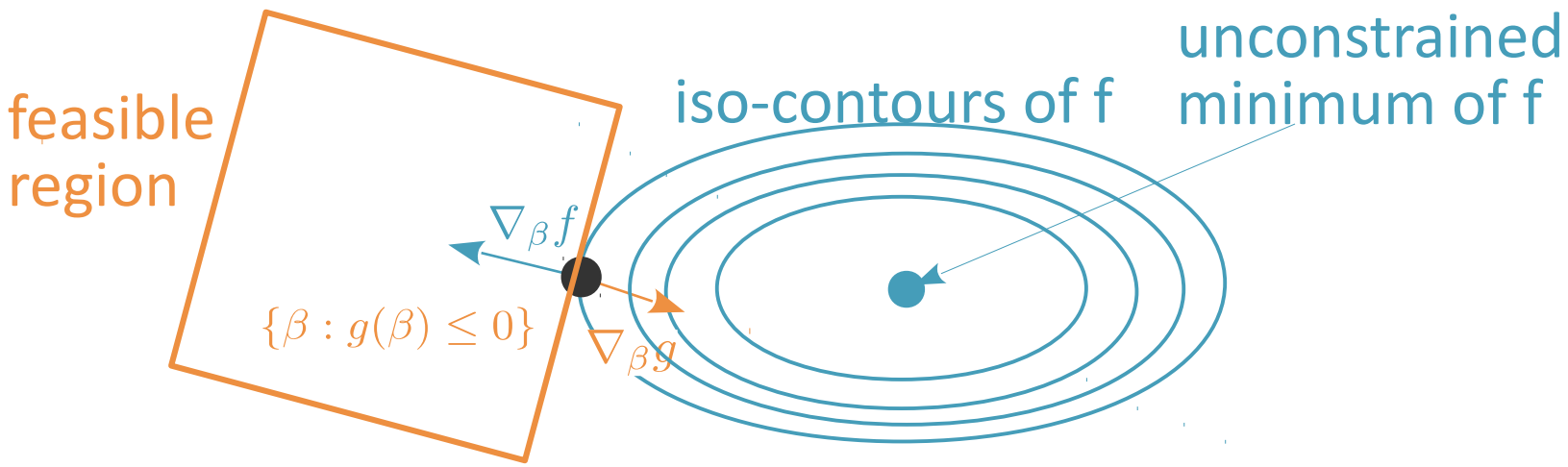
$$f(\boldsymbol{\beta}) = ||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2$$

Geometric interpretation

Minimize f(β) under the constraint g(β) ≤ 0

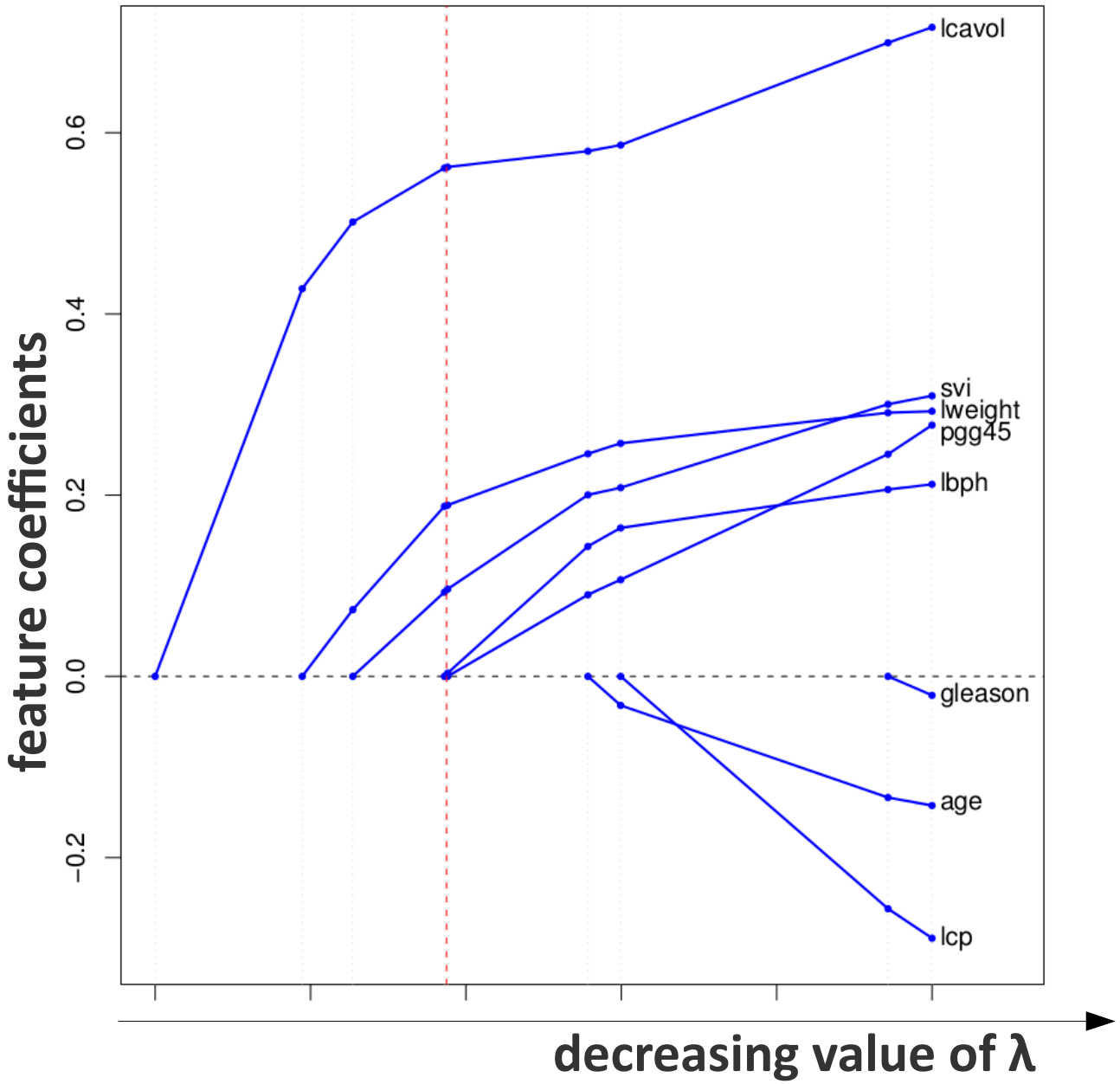
- **Case 1:** the unconstrained minimum lies in the feasible region.
- **Case 2:** it does not.

$$g(\boldsymbol{\beta}) = ||\boldsymbol{\beta}||_1 - t \qquad f(\boldsymbol{\beta}) = ||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2$$



The gradient is orthonormal to the iso-contours and points towards the direction of maximum increase.

Lasso solution path



Forward stepwise regression

- Build model **sequentially**, adding one variable at a time
  - Start with the intercept
  - At each step, add the variable that **most improves the fit**
  - **Stop when** $||\boldsymbol{\beta}||_1 \leq t$
- Greedy solution

Least Angle Regression

At each step, add "only as much of a variable as needed"

1. Standardize the predictors to have mean zero and unit norm. Start with the residual $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \ldots, \beta_p = 0$.

Least Angle Regression

At each step, add "only as much of a variable as needed"

1. Standardize the predictors to have mean zero and unit norm. Start with the residual $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \ldots, \beta_p = 0$.
2. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$.

Least Angle Regression

At each step, add "only as much of a variable as needed"

1. Standardize the predictors to have mean zero and unit norm. Start with the residual $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \ldots, \beta_p = 0$.

2. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$.

3. Move β_j from 0 towards its least-squares coefficient $\langle \mathbf{x}_j, \mathbf{r} \rangle$, until some other competitor $\mathbf{x}_k$ has as much correlation with the current residual as does $\mathbf{x}_j$.

$$\boxed{\begin{aligned} \beta_j &\leftarrow \beta_j + \alpha \frac{1}{\sum_{i=1}^n (x_j^i)^2} \sum_{i=1}^n x_j^i r^i \\ &= \beta_j + \alpha (x_j^\top x_j)^{-1} x_j^\top r \\ &= \beta_j + \alpha \langle x_j^\top, x_j \rangle^{-1} \langle x_j, r \rangle \end{aligned}}$$

$$\boxed{r = (y - \bar{y}) - \beta_j x_j}$$

step size

Least Angle Regression

At each step, add "only as much of a variable as needed"

1. Standardize the predictors to have mean zero and unit norm. Start with the residual $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \ldots, \beta_p = 0$.
2. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$.
3. Move β_j from 0 towards its least-squares coefficient $\langle \mathbf{x}_j, \mathbf{r} \rangle$, until some other competitor $\mathbf{x}_k$ has as much correlation with the current residual as does $\mathbf{x}_j$.
4. Move β_j and β_k in the direction defined by their joint least squares coefficient of the current residual on $(\mathbf{x}_j, \mathbf{x}_k)$, until some other competitor $\mathbf{x}_l$ has as much correlation with the current residual.

$$\boxed{r = (y - \bar{y}) - \beta_j x_j - \beta_k x_k}$$

Least Angle Regression

At each step, add "only as much of a variable as needed"

1. Standardize the predictors to have mean zero and unit norm. Start with the residual $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \ldots, \beta_p = 0$.

2. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$.

3. Move β_j from 0 towards its least-squares coefficient $\langle \mathbf{x}_j, \mathbf{r} \rangle$, until some other competitor $\mathbf{x}_k$ has as much correlation with the current residual as does $\mathbf{x}_j$.

4. Move β_j and β_k in the direction defined by their joint least squares coefficient of the current residual on $(\mathbf{x}_j, \mathbf{x}_k)$, until some other competitor $\mathbf{x}_l$ has as much correlation with the current residual.

4a. If a non-zero coefficient hits zero, drop its variable from the active set of variables and recompute the current joint least squares direction.

Least Angle Regression

At each step, add "only as much of a variable as needed"

1. Standardize the predictors to have mean zero and unit norm. Start with the residual $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \ldots, \beta_p = 0$.

2. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$.

3. Move β_j from 0 towards its least-squares coefficient $\langle \mathbf{x}_j, \mathbf{r} \rangle$, until some other competitor $\mathbf{x}_k$ has as much correlation with the current residual as does $\mathbf{x}_j$.

4. Move β_j and β_k in the direction defined by their joint least squares coefficient of the current residual on $(\mathbf{x}_j, \mathbf{x}_k)$, until some other competitor $\mathbf{x}_l$ has as much correlation with the current residual.

4a. If a non-zero coefficient hits zero, drop its variable from the active set of variables and recompute the current joint least squares direction.

5. Continue in this way until all p predictors have been entered.

Least Angle Regression

At each step, add "only as much of a variable as needed"

1. Standardize the predictors to have mean zero and unit norm. Start with the residual $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \ldots, \beta_p = 0$.
2. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$.
3. Move β_j from 0 towards its least-squares coefficient $\langle \mathbf{x}_j, \mathbf{r} \rangle$, until some other competitor $\mathbf{x}_k$ has as much correlation with the current residual as does $\mathbf{x}_j$.
4. Move β_j and β_k in the direction defined by their joint least squares coefficient of the current residual on $(\mathbf{x}_j, \mathbf{x}_k)$, until some other competitor $\mathbf{x}_l$ has as much correlation with the current residual.

4a. If a non-zero coefficient hits zero, drop its variable from the active set of variables and recompute the current joint least squares direction.

5. Continue in this way until all p predictors have been entered.

Maximum number of steps: max(n-1, p)

Approaches to dimensionality reduction

- **Feature selection**

  Choose m < p features, ignore the remaining (p-m)

  - **Filtering** approaches

    Apply a statistical measure to assign a score to each feature (correlation, χ^2-test).

  - **Wrapper** approaches

    Search problem: Find the best set of features for a given predictive model.

  - **Embedded** approaches

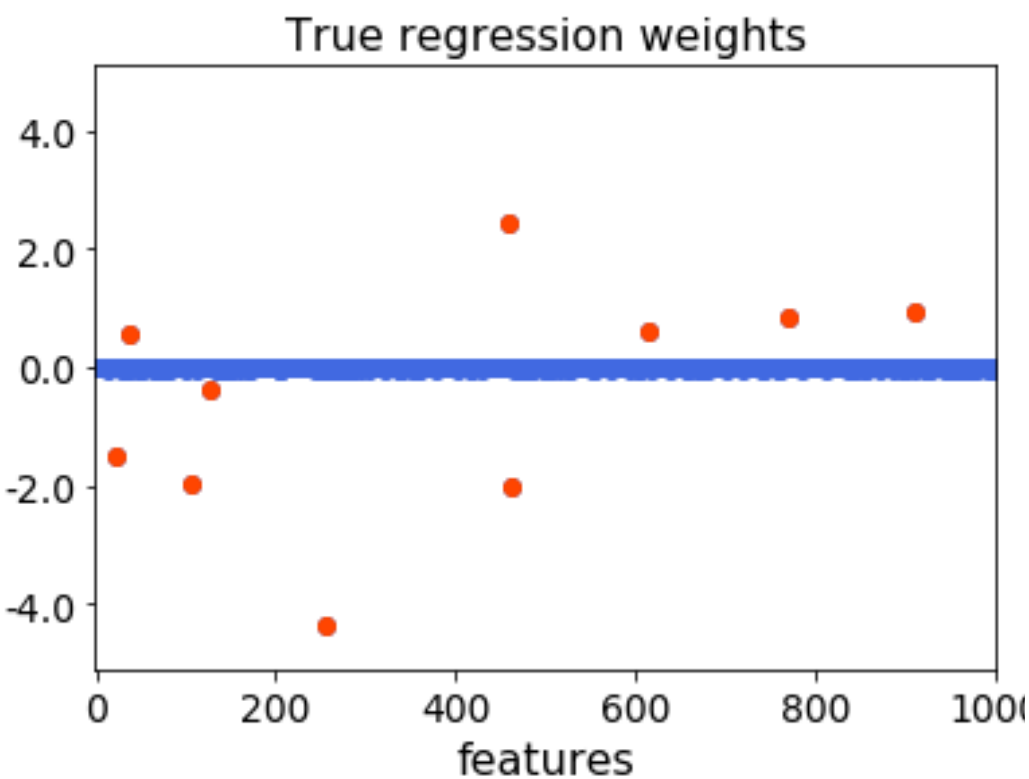
    Simultaneously fit a model and learn which features should be included.

All these feature selection approaches are **supervised.**

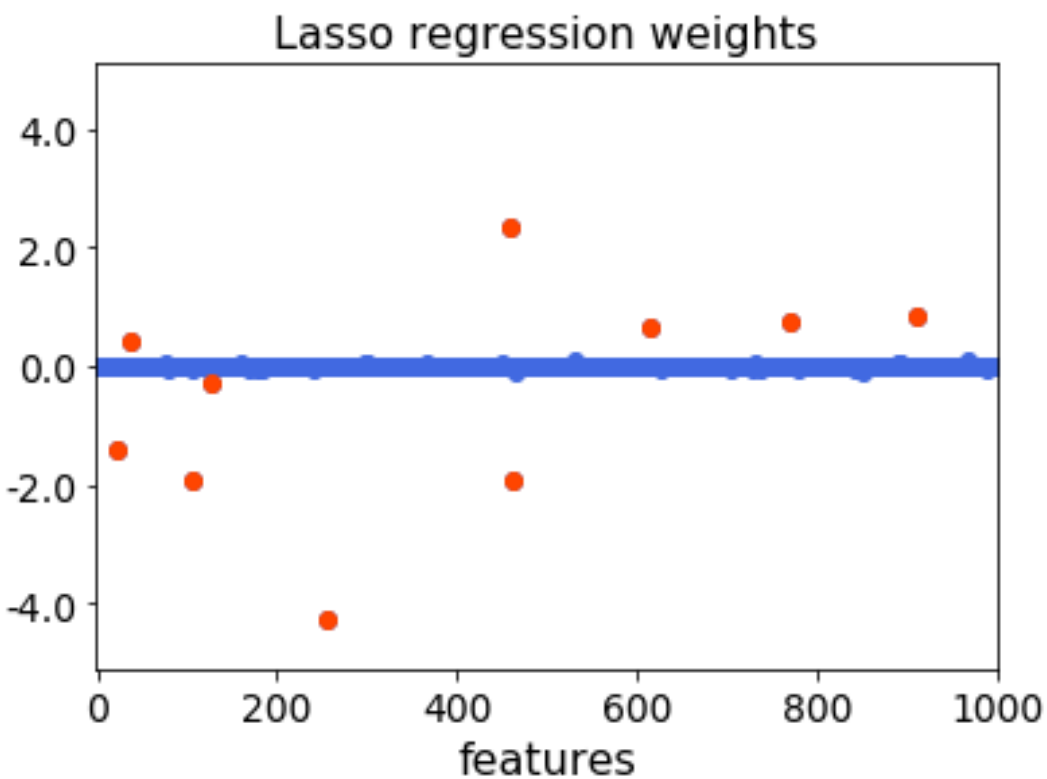
Linear regression when p >> n

Simulated data: p=1000, n=100, 10 causal features

True coefficients



Predicted coefficients



Elastic Net

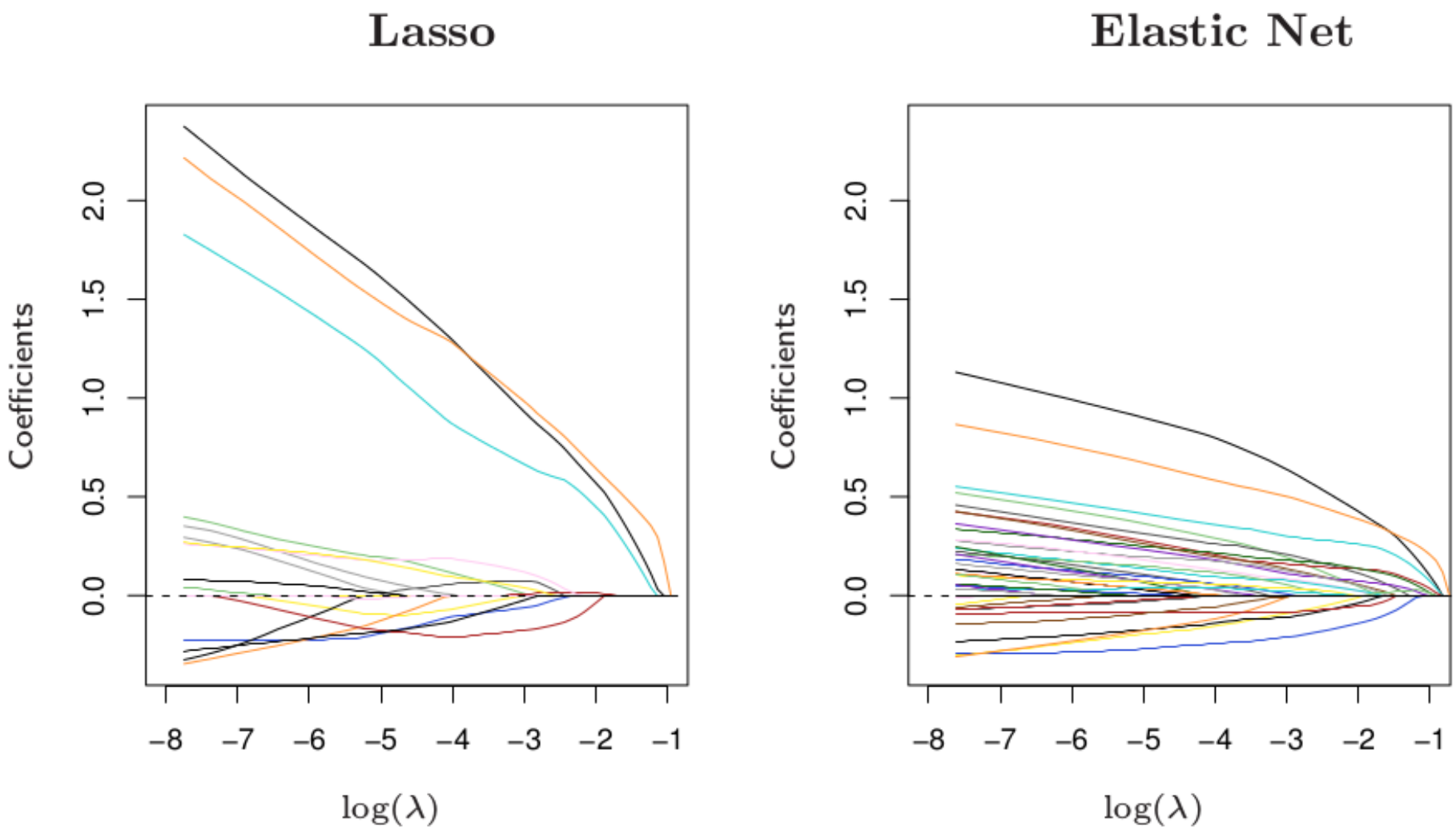
Elastic Net

- **Combine lasso** and **ridge regression**

$$\hat{\boldsymbol{\beta}}_{\text{enet}} = \arg\min_{\boldsymbol{\beta}} ||y - X\boldsymbol{\beta}||_2^2 + \lambda \left(\alpha||\boldsymbol{\beta}||_2^2 + (1-\alpha)||\boldsymbol{\beta}||_1\right)$$

  - **Select variables** like the lasso.
  - **Shrinks together coefficients of correlated variables** like the ridge regression.

E.g. Leukemia data



Elastic Net results in more non-zero coefficients than Lasso, but with smaller amplitudes.

Lq-norm regularization

Lq-norm regularization

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||Y - X\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_q^q \qquad ||\boldsymbol{\beta}||_q = \left(\sum_{j=1}^{p} |\beta_j|^q \right)^{1/q}$$

Equivalently:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} ||Y - X\boldsymbol{\beta}||_2^2 \text{ s. t. } ||\boldsymbol{\beta}||_q^q \leq s$$

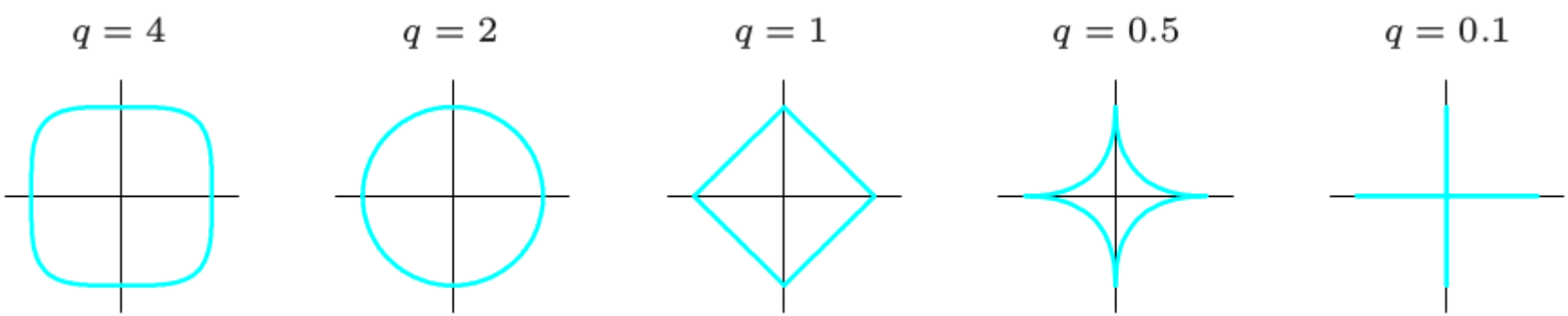
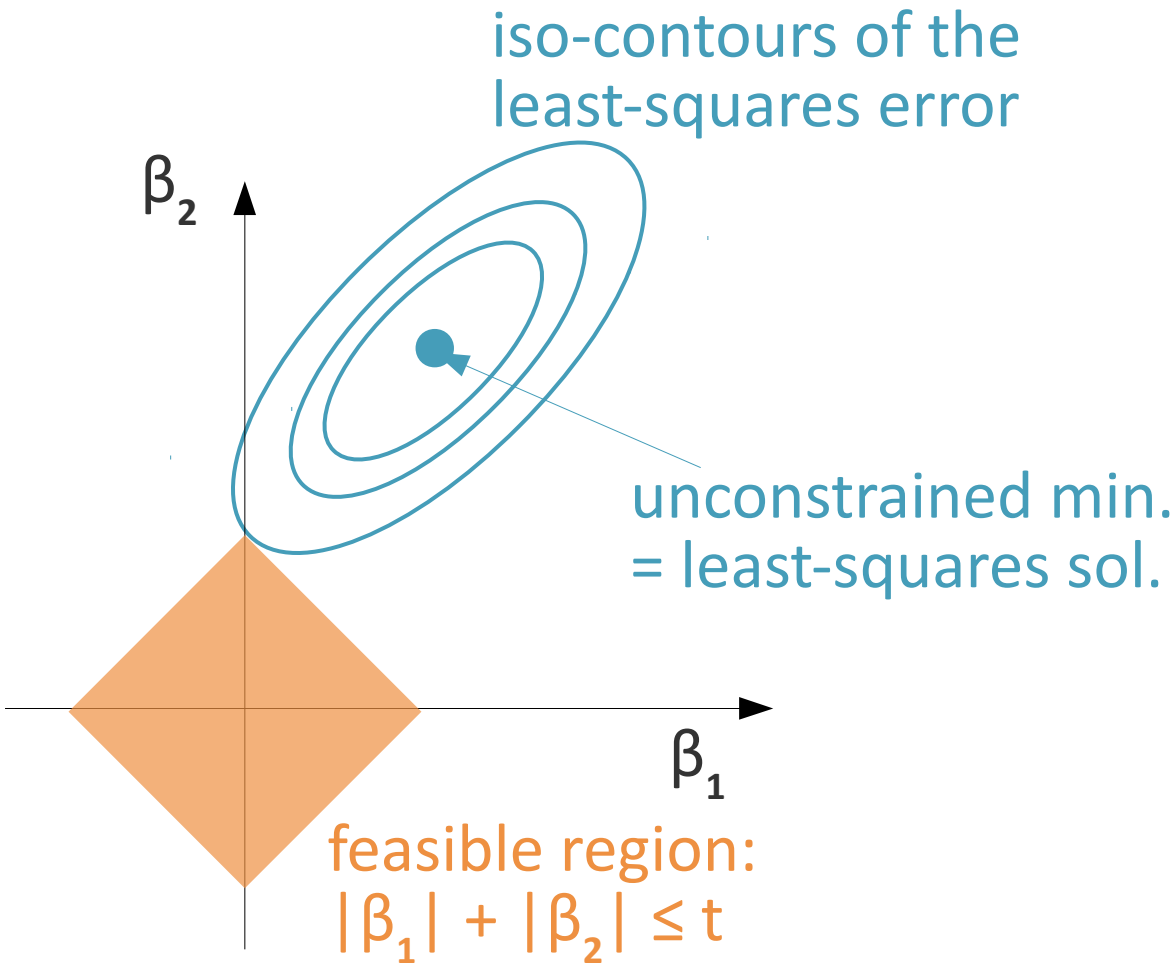


FIGURE 3.12. *Contours of constant value of* $\sum_j |\beta_j|^q$ *for given values of* q.

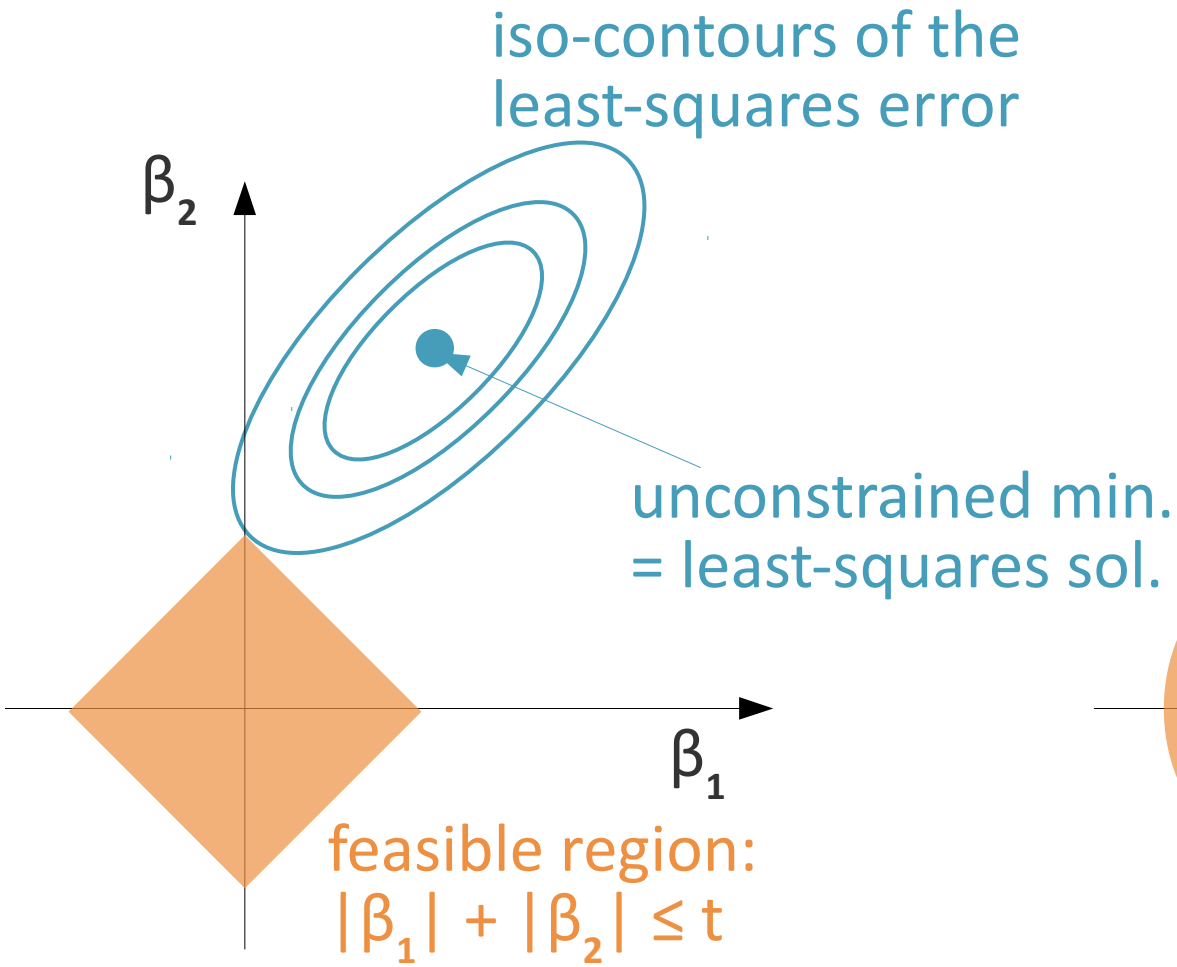
Lasso vs. ridge

L1 norm

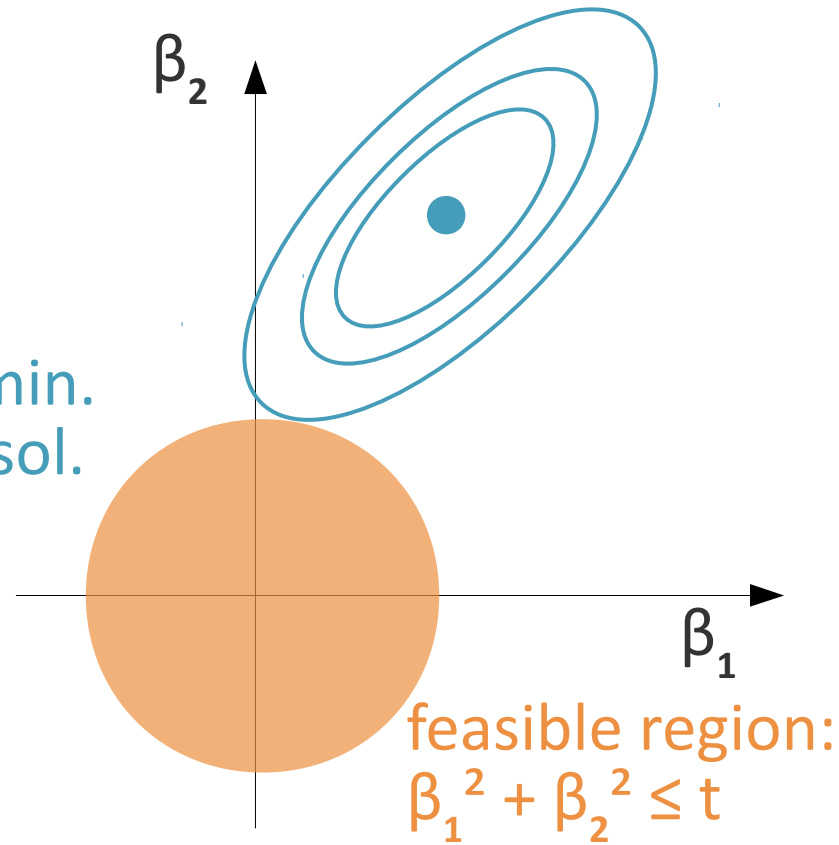


Lasso vs. ridge

L1 norm



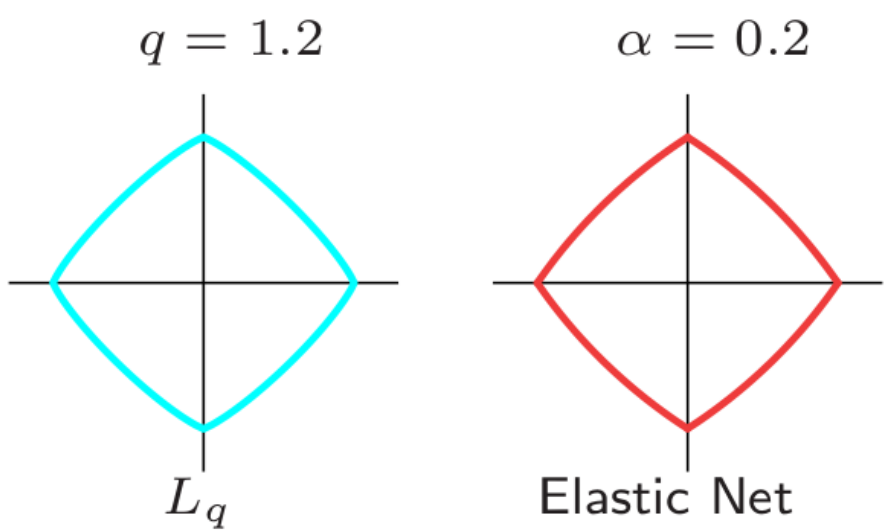
L2 norm



Elastic net

- **Elastic penalty**

$$\hat{\beta} = \arg\min_{\beta} ||y - X\beta||_2^2 + \lambda \left(\alpha||\beta||_2^2 + (1-\alpha)||\beta||_1\right)$$



Structured regularization

Group lasso

Use K predefined groups of variables that are known to "work" together and expected to be either all active or all inactive together.

E.g.: genes belonging to the same biological pathway.

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||y - \sum_{k=1}^{K} X_k \boldsymbol{\beta}_k||_2^2 + \lambda \sum_{k=1}^{K} \sqrt{p_k}\, ||\boldsymbol{\beta}_k||_2$$

Size of group k

Features belonging to group k

Other examples of structured penalties

- **Overlapping groups**

  Jacob et al. (2009). Group lasso with overlap and graph lasso. *ICML.*

- **Graphs**

  Li & Li (2010). Variable selection and regression analysis for graph-structured covariates with an application to genomics. *Ann. App. Stats.*

- **Trees**

  Zhao et al. (2006). Grouped and hierarchical model selection through composite absolute penalties. *Ann. Stat.*

- **Multiple related tasks**

  Obozinski et al. (2006). Multitask feature selection. *Technical Report, UC Berkeley.*

Minimize SSE + λ x regularizer

- **Ridge**
  - gives similar weights to similar variables
  - not very sparse
  - analytical solution
- **Lasso**
  - randomly picks one of several correlated variables
  - sparse
  - LAR algorithm
- **Elastic net**
  - selects variables like the lasso
  - shrinks together the coefficients of correlated variables.
- **Many other regularizers** are possible

  Lp norms, groups, graphs, trees...

References

- *A Course in Machine Learning.* `http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf`
  - **Regularization:** Chap 7.2–7.3
- *The Elements of Statistical Learning.* `http://web.stanford.edu/~hastie/ElemStatLearn/`
  - **Regularization:** Chap 10.12
  - **Ridge regression:** Chap 3.4.1
  - **Lasso:** Chap 3.4.2
  - **LAR:** Chap 3.4.4
  - **Elastic Net:** Chap 4.2

Lab 5

- Logistic regression with no scaling

```
# Logistic regression (no regularization, no scaling)
from sklearn import linear_model
clf_logreg = linear_model.LogisticRegression(C=1e6) # large C = no regularization

# Train the model
clf_logreg.fit(Xtr, ytr)

# Predict on the test set
# Predicted probabilities of belonging to the positive class
pos_idx = list(clf_logreg.classes_).index(1)
ypred_logreg = clf_logreg.predict_proba(Xte)[:, pos_idx]

# Predicted binary labels
ypred_logreg_b = np.where(ypred_logreg > 0.5, 1, 0)

from sklearn import metrics
print("No regularization: accuracy = %.3f" % metrics.accuracy_score(yte, ypred_logreg_b))
print("AUC = %.3f" % (metrics.roc_auc_score(yte, ypred_logreg)))
```

```
No regularization: accuracy = 0.636
AUC = 0.642
```

- Logistic regression with scaling

```
# Logistic regression (no regularization, scaling)
clf_logreg_s = linear_model.LogisticRegression(C=1e6)

# Train the model
clf_logreg_s.fit(Xtr_scaled, ytr)

# Predict on the test set
# Predicted probabilities of belonging to the positive class
pos_idx = list(clf_logreg_s.classes_).index(1)
ypred_logreg_s = clf_logreg_s.predict_proba(Xte_scaled)[:, pos_idx]
# Predicted binary labels
ypred_logreg_s_b = np.where(ypred_logreg_s > 0.5, 1, 0)

print("Scaled, no regularization: accuracy = %.3f" % metrics.accuracy_score(yte, ypred_logreg_s_b))
print("AUC = %.3f" % (metrics.roc_auc_score(yte, ypred_logreg_s)))
```

```
Scaled, no regularization: accuracy = 0.782
AUC = 0.859
```

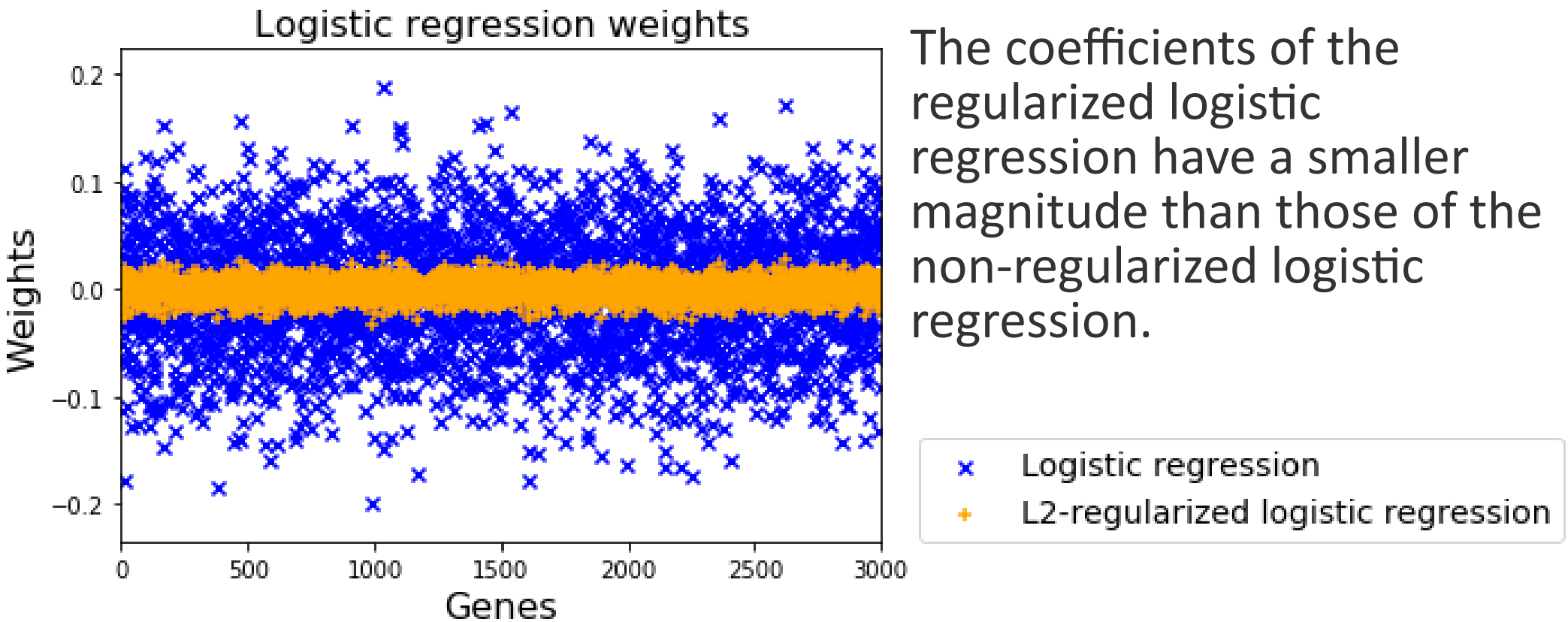
- L2-regularized logistic regression with scaling

```
cvalue = 0.01
clf_logreg_l2_s = linear_model.LogisticRegression(C=cvalue, penalty='l2')

# Train the model
clf_logreg_l2_s.fit(Xtr_scaled, ytr)

# index of positive class
pos_idx = list(clf_logreg_l2_s.classes_).index(1)
# predict probability of being positive
ypred_logreg_l2_s = clf_logreg_l2_s.predict_proba(Xte_scaled)[:, pos_idx]
# predict binary labels
ypred_logreg_l2_s_b = np.where(ypred_logreg_l2_s > 0.5, 1, 0)

print("Scaled, l2 regularization (C=%.2e): accuracy = %.3f" % (cvalue,
                                                                metrics.accuracy_score(yte,
                                                                                       ypred_logreg_l2_s_b)))
print("AUC = %.3f" % (metrics.roc_auc_score(yte, ypred_logreg_l2_s)))
```

```
Scaled, l2 regularization (C=1.00e-02): accuracy = 0.782
AUC = 0.858
```

- The performance did not change.
- Did the regression coefficients change?

```python
# Effect of l2-regularization on the weights
num_features = X_clf.shape[1]
plt.scatter(range(num_features), clf_logreg_s.coef_,
            color='blue', marker='x', label='Logistic regression')
plt.scatter(range(num_features), clf_logreg_l2_s.coef_,
            color='orange', marker='+', label='L2-regularized logistic regression')

plt.xlabel('Genes', fontsize=16)
plt.ylabel('Weights', fontsize=16)
plt.title('Logistic regression weights', fontsize=16)
plt.legend(fontsize=14, loc=(1.05, 0))
plt.xlim([0, num_features])
```



The coefficients of the regularized logistic regression have a smaller magnitude than those of the non-regularized logistic regression.

- Optimization of the parameter C

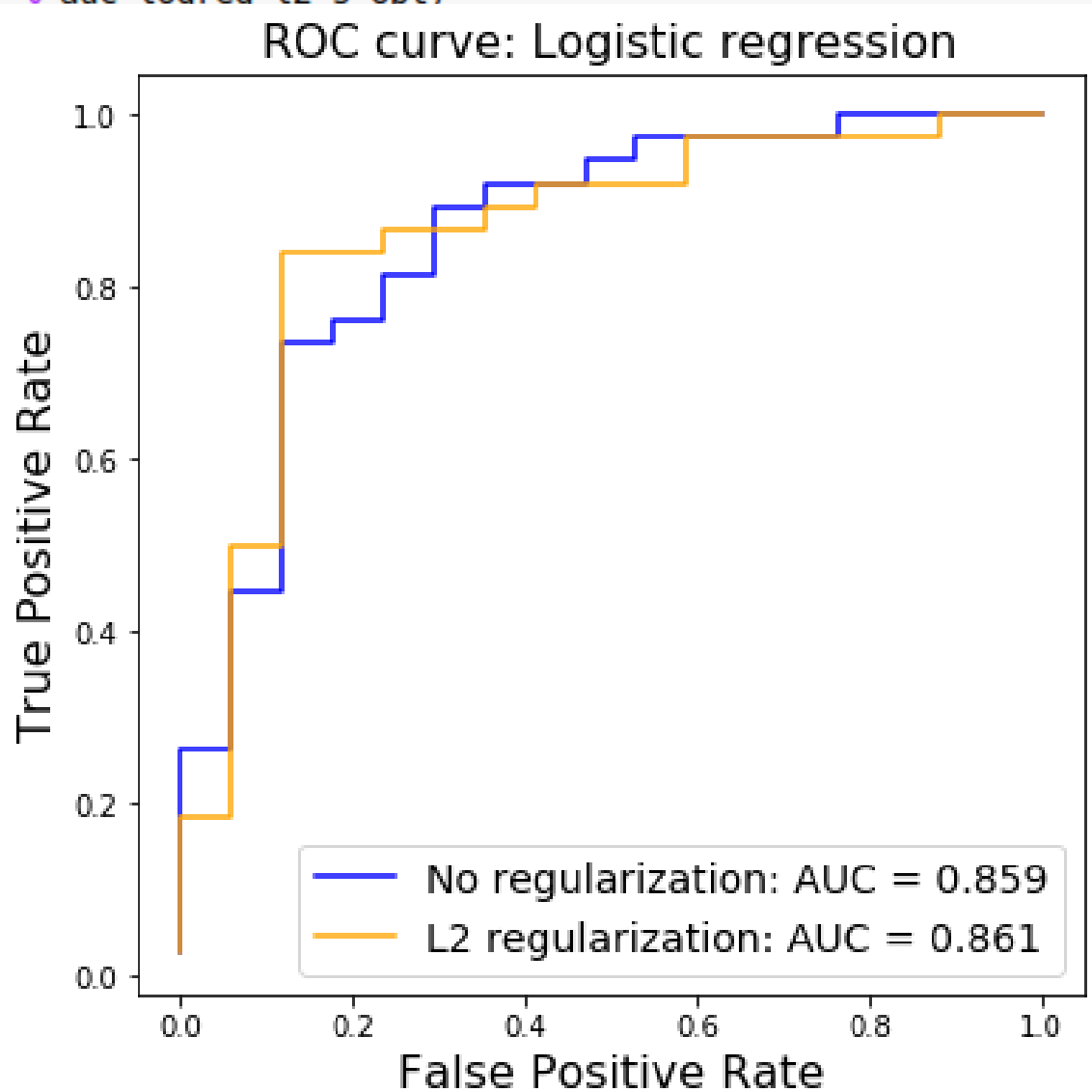
```
# Optimize cvalue
classifier = linear_model.LogisticRegression(penalty='l2')
param_grid = {'C': cvalues_list}
clf_logreg_l2_s_opt = model_selection.GridSearchCV(classifier, param_grid, cv=3)

# Train the model
clf_logreg_l2_s_opt.fit(Xtr_scaled, ytr)

# index of the positive class
pos_idx = list(clf_logreg_l2_s_opt.best_estimator_.classes_).index(1)
# predict probability of being positive
ypred_logreg_l2_s_opt = clf_logreg_l2_s_opt.best_estimator_.predict_proba(Xte_scaled)[:, pos_idx]
# predict binary label
ypred_logreg_l2_s_opt_b = np.where(ypred_logreg_l2_s_opt > 0.5, 1, 0)

cvalue_opt = clf_logreg_l2_s_opt.best_estimator_.C
print("Scaled, l2 regularization (C=%.2e): accuracy = %.3f" % (cvalue_opt,
                                                                metrics.accuracy_score(yte,
                                                                                       ypred_logreg_l2_s_opt_b)))
print("AUC = %.3f" % (metrics.roc_auc_score(yte, ypred_logreg_l2_s_opt)))
```

```
Scaled, l2 regularization (C=3.79e-04): accuracy = 0.818
AUC = 0.861
```

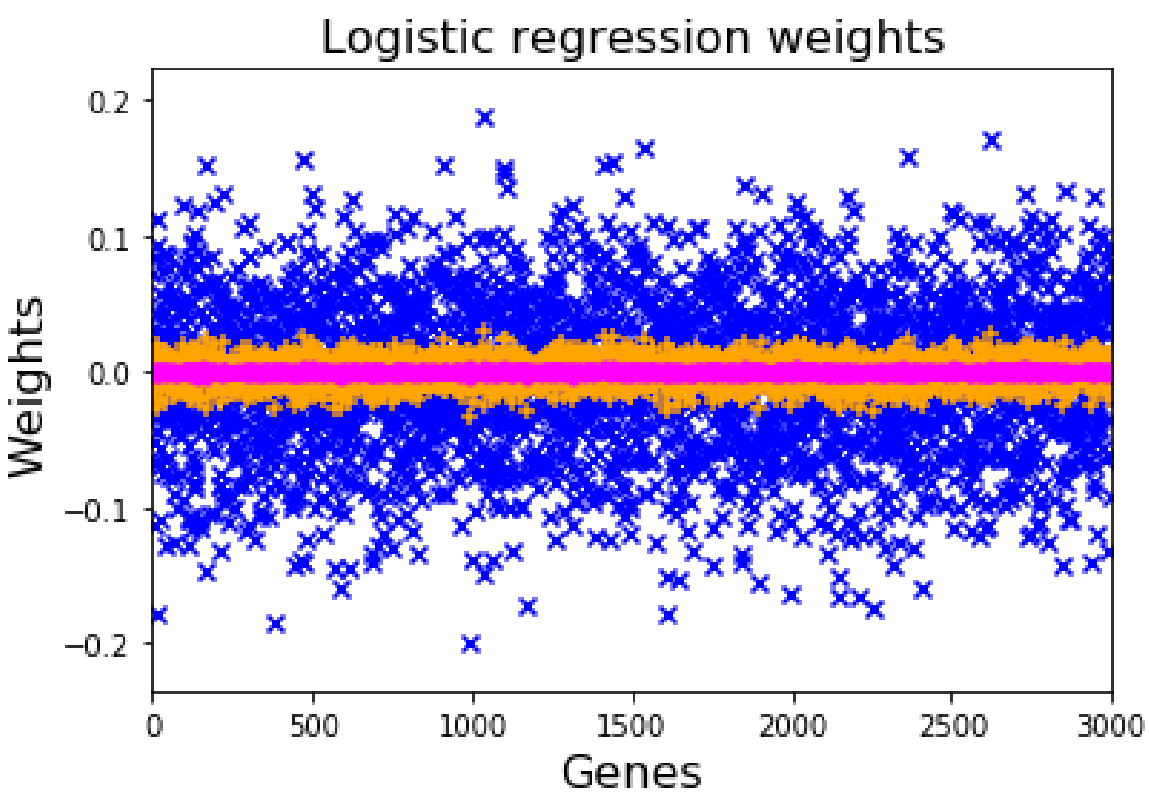
- Now C = 0.000379 (smaller), ie λ has increased. There is *more* regularization. We expect smaller regression coefficients.
- The performance has slightly improved

```
fig = plt.figure(figsize=(6, 6))
fpr_logreg_s, tpr_logreg_s, t = metrics.roc_curve(yte, ypred_logreg_s, pos_label=1)
auc_logreg_s = metrics.auc(fpr_logreg_s, tpr_logreg_s)
plt.plot(fpr_logreg_s, tpr_logreg_s, color='blue',
         label='No regularization: AUC = %0.3f' % auc_logreg_s)

fpr_logreg_l2_s_opt, tpr_logreg_l2_s_opt, t = metrics.roc_curve(yte, ypred_logreg_l2_s_opt, pos_label=1)
auc_logreg_l2_s_opt = metrics.auc(fpr_logreg_l2_s_opt, tpr_logreg_l2_s_opt)
plt.plot(fpr_logreg_l2_s_opt, tpr_logreg_l2_s_opt, color='orange',
         label='L2 regularization: AUC = %0.3f' % auc_logreg_l2_s_opt)

plt.xlabel('False Positive Rate', fontsize=16)
plt.ylabel('True Positive Rate', fontsize=16)
plt.title('ROC curve: Logistic regression', fon
plt.legend(fontsize=14)
```

- The ROC curves are not very different from each other (the AUC are quite close to each other as well).



```python
# Effect of l2-regularization on the weights
num_features = X_clf.shape[1]
plt.scatter(range(num_features), clf_logreg_s.coef_,
            color='blue', marker='x', label='Logistic regression')
plt.scatter(range(num_features), clf_logreg_l2_s.coef_,
            color='orange', marker='+', label='L2-regularized logistic regression')
plt.scatter(range(num_features), clf_logreg_l2_s_opt.best_estimator_.coef_,
            color='magenta', marker='.', label='L2-regularized logistic regression (opt)')

plt.xlabel('Genes', fontsize=16)
plt.ylabel('Weights', fontsize=16)
plt.title('Logistic regression weights', fontsize=16)
plt.legend(fontsize=14, loc=(1.05, 0))
plt.xlim([0, num_features])
```



- As expected, the optimized l2-regularized logistic regression has coefficients of even smaller magnitude than the non-optimized one.

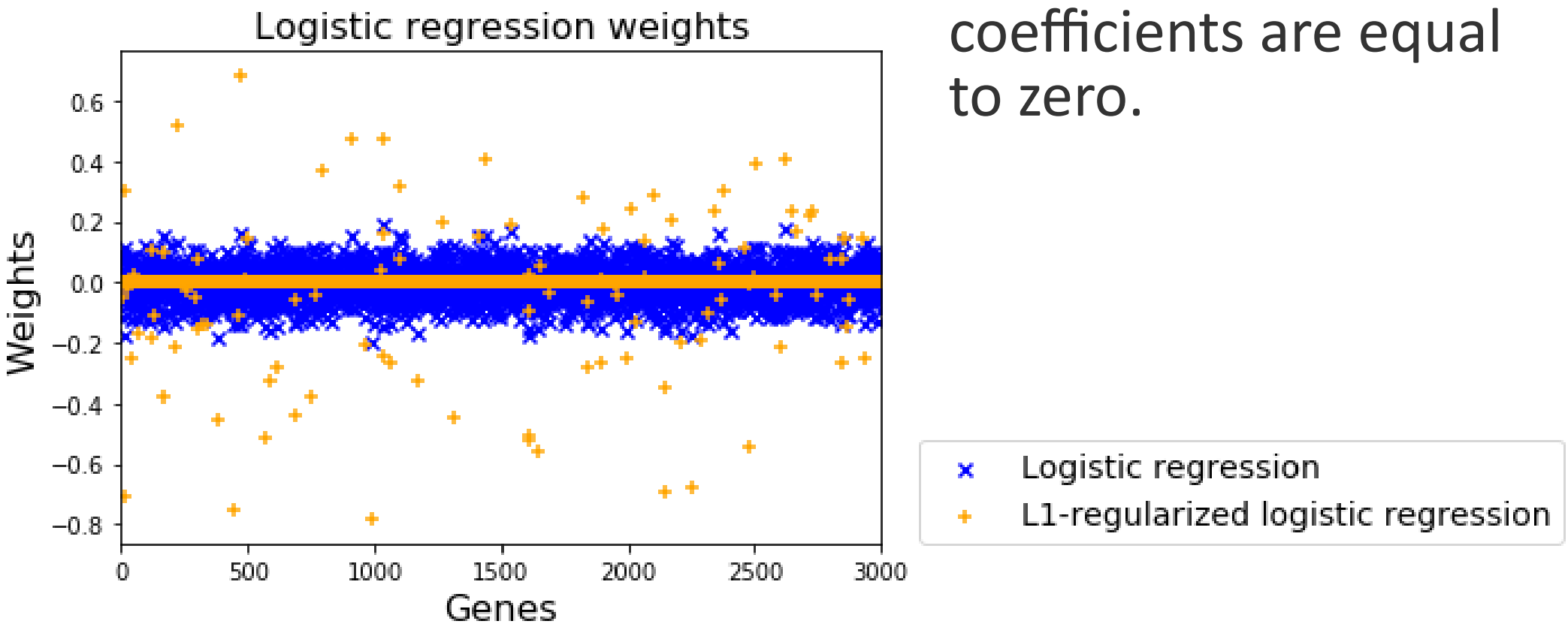
- L1-regularized logistic regression with scaling

```
cvalue = 10.
clf_logreg_l1_s = linear_model.LogisticRegression(C=cvalue, penalty='l1')

# train model
clf_logreg_l1_s.fit(Xtr_scaled, ytr)

# index of the positive class
pos_idx = list(clf_logreg_l1_s.classes_).index(1)
# predict the probability of belonging to the positive class
ypred_logreg_l1_s = clf_logreg_l1_s.predict_proba(Xte_scaled)[:, pos_idx]
# predict binary labels
ypred_logreg_l1_s_b = np.where(ypred_logreg_l1_s > 0.5, 1, 0)

print("Scaled, l1 regularization (C=%.2e): accuracy = %.3f" % (cvalue,
                                                                metrics.accuracy_score(yte,
                                                                                       ypred_logreg_l1_s_b)))
print("AUC = %.3f" % (metrics.roc_auc_score(yte, ypred_logreg_l1_s)))
```

```
Scaled, l1 regularization (C=1.00e+01): accuracy = 0.727
AUC = 0.748
```

- The performance is now noticeably worse!
- What are the regression coefficients like?

```python
num_features = X_clf.shape[1]
plt.scatter(range(num_features), clf_logreg_s.coef_,
            color='blue', marker='x', label='Logistic regression')
plt.scatter(range(num_features), clf_logreg_l1_s.coef_,
            color='orange', marker='+', label='L1-regularized logistic regression')

plt.xlabel('Genes', fontsize=16)
plt.ylabel('Weights', fontsize=16)
plt.title('Logistic regression weights', fontsize=16)
plt.legend(fontsize=14, loc=(1.05, 0))
plt.xlim([0, num_features])
```

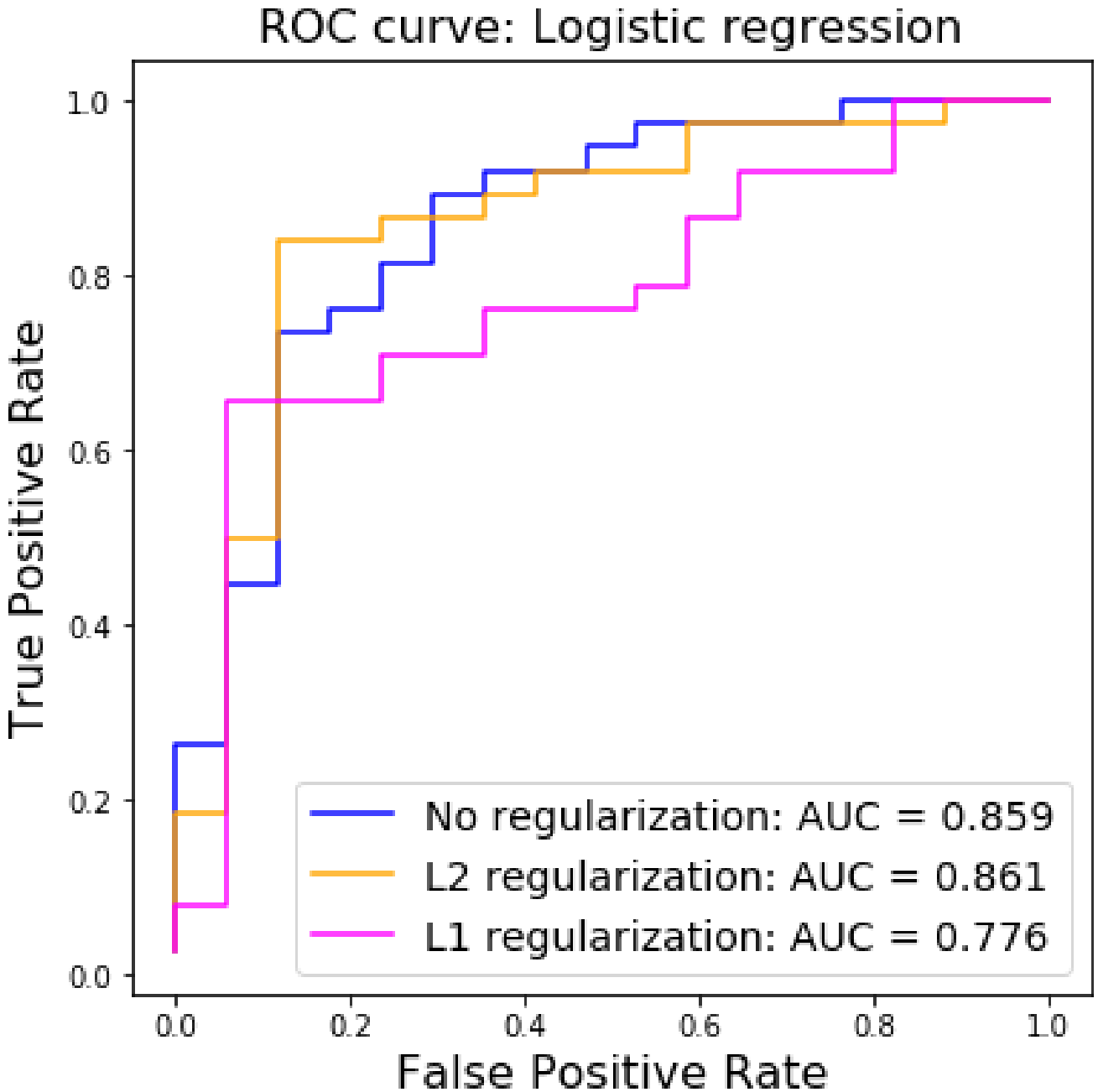


- It looks like many coefficients are equal to zero.

```python
# Number of selected features
print("The non-regularized logistic regression uses %d features" % \
      len(np.where(clf_logreg_s.coef_ !=0)[1]))
print("The L2-regularized logistic regression uses %d features" % \
      len(np.where(clf_logreg_l2_s.coef_ !=0)[1]))
print("The L1-regularized logistic regression uses %d features" % \
      len(np.where(clf_logreg_l1_s.coef_ !=0)[1]))

print("Number of features discarded by the L1-regularization: %d" % \
      (Xtr.shape[1] - len(np.where(clf_logreg_l1_s.coef_ !=0)[1])))
```

```
The non-regularized logistic regression uses 3000 features
The L2-regularized logistic regression uses 3000 features
The L1-regularized logistic regression uses 106 features
Number of features discarded by the L1-regularization: 2894
```

- Indeed, the performance got worse but the l1-regularized model only uses **106** features!
- Notice that if you train the l1-regularized logistic regression you very likely get a different performance and a different number of features: the l1 regularization is **unstable.**



- The ROC curve of the l1-regularized logistic regression is quite below the other curves, confirming a worse performance.

- For me, the optimal value of C is 88.6
- A larger C means a smaller λ, hence less regularization. I am expecting *more* non-zero weights (i.e. selected features).
- Indeed, my l1-regularized model with optimized C uses 289 features. These 289 features include the previous features (remember the lasso regularization path: once a feature is included it stays included).