ES APPRENTISSAGE ARTIFICIEL

Apprentissage statistique en Bio-Informatique Régularisation et noyaux

Judith Abécassis
Centre for Computational Biology, Mines ParisTech
judith.abecassis@mines-paristech.fr

Material by Chloé-Agathe Azencott

Centre for Computational Biology

- Centre de recherche des MINES
- Associé à un centre de recherche cancer (Institut Curie)
- Thématique : développement et mise en œuvre de méthodes de machine learning pour la bioinformatique du cancer

Supervised machine learning in bioinformatics

- Quelles données ?
  - Images biologiques
  - Données génomiques
    - Reads de séquençage ADN ou ARN, mutations d'une seule paire de base (SNPs), méthylation, etc
  - Screens de composés chimiques
  - De plus en plus : dossiers patients

Supervised machine learning in bioinformatics

- Quels problèmes ?
- Régression :
  - **Gene expression regulation:** how much of this gene will be expressed?
  - When will this **patient relapse?**
  - **Drug eff cacy:** how well does this drug work on this tumor?
  - What is the **binding aff nity** between these two molecules?
  - How **soluble** is this chemical in water?

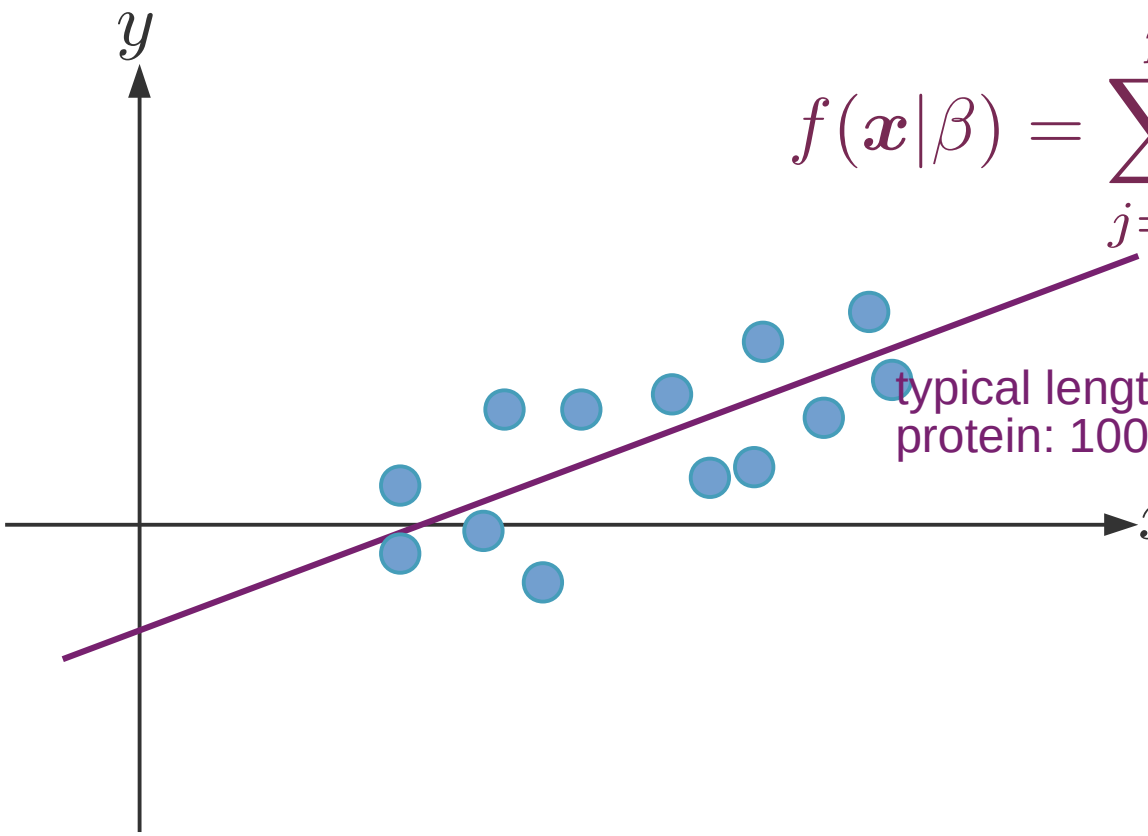
Small n, large p problems

- Few patients, many features (cost of data, rare diseases, invasiveness of data collection)
- Challenges of high-dimension:
  - Curse of dimensionality (intuitions that work in 2D may not work in higher dimensions)
  - Overf tting is more likely
  - Problems become ill-posed

Linear regression

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

$\mathcal{D} = \{\boldsymbol{x}^i, y^i\}_{i=1,\ldots,n}$



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

typical length of a human protein: 100-600 amino acids

Linear regression least-squares f t

- Minimize the **residual sum of squares**

$$\begin{aligned}
\mathrm{RSS}(\beta) &= \sum_{i=1}^{n} (y^i - f(\boldsymbol{x}^i))^2 \\
&= \sum_{i=1}^{n} \left(y^i - \beta_0 - \sum_{j=1}^{p} x_j^i \beta_j \right)^2 \\
&= (y - X\beta)^\top (y - X\beta)
\end{aligned}$$

$$X = \begin{pmatrix}
1 & x_1^1 & x_2^1 & \cdots & x_p^1 \\
1 & x_1^2 & x_2^2 & \cdots & x_p^2 \\
\vdots & \vdots & \cdots & \vdots & \\
1 & x_1^n & x_2^n & \cdots & x_p^n
\end{pmatrix}$$

Linear regression least-squares f t

- Minimize the **residual sum of squares**

$$\begin{aligned} \mathrm{RSS}(\beta) &= \sum_{i=1}^{n} (y^i - f(\boldsymbol{x}^i))^2 \\ &= \sum_{i=1}^{n} \left(y^i - \beta_0 - \sum_{j=1}^{p} x_j^i \beta_j \right)^2 \\ &= (y - X\beta)^\top (y - X\beta) \end{aligned}$$

Historically:

- Carl Friedrich Gauss (to predict the location of Ceres)
- Adrien Marie Legendre

Linear regression least-squares f t

- Minimize the **residual sum of squares**

$$\begin{aligned}\mathrm{RSS}(\beta) &= \sum_{i=1}^{n}(y^i - f(\boldsymbol{x}^i))^2 \\ &= \sum_{i=1}^{n}\left(y^i - \beta_0 - \sum_{j=1}^{p} x_j^i \beta_j\right)^2 \\ &= (y - X\beta)^\top (y - X\beta)\end{aligned}$$

Estimate :

$$\nabla_\beta \mathrm{RSS} = -2X^\top (y - X\beta)$$

$$X^\top (y - X\hat{\beta}) = 0$$

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

Linear regression least-squares f t

- Minimize the **residual sum of squares**

$$\begin{aligned} \mathrm{RSS}(\beta) &= \sum_{i=1}^{n} (y^i - f(\boldsymbol{x}^i))^2 \\ &= \sum_{i=1}^{n} \left(y^i - \beta_0 - \sum_{j=1}^{p} x_j^i \beta_j \right)^2 \\ &= (y - X\beta)^\top (y - X\beta) \end{aligned}$$

- Assuming **X has full column rank** (and hence $X^\top X$ invertible):

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

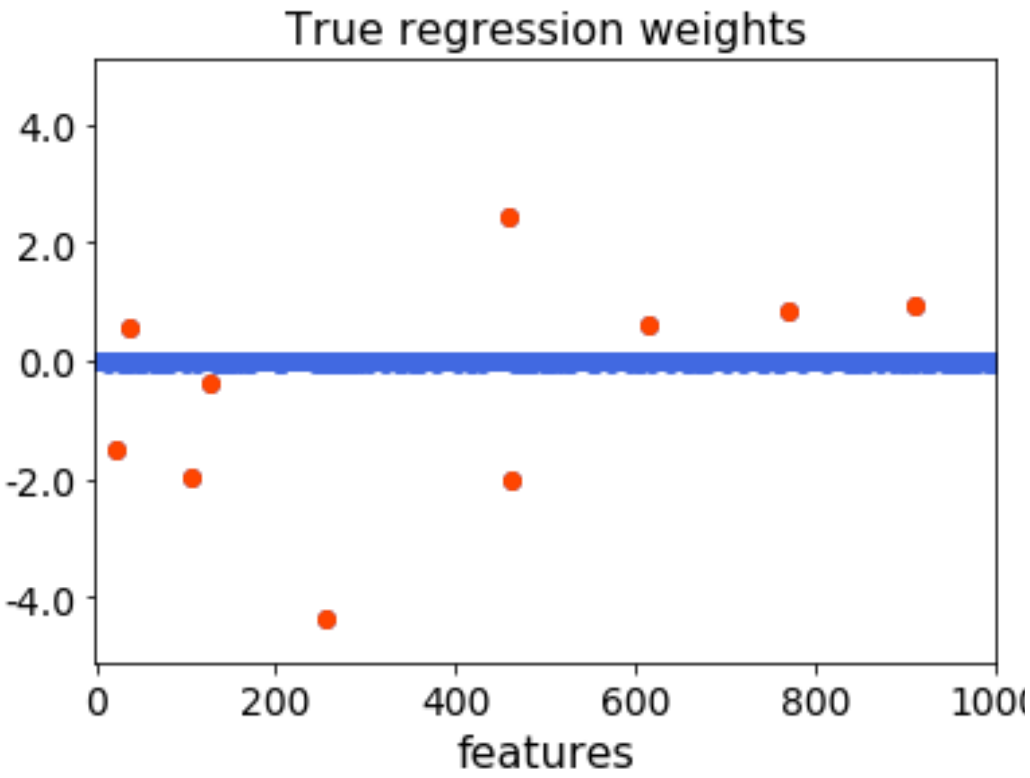
If n << p

- X (dimensions n x p) cannot have full column rank
- Therefore X^TX cannot be inverted
- An infinity of solutions exist
  - One can be found using gradient descent or a pseudo-inverse
  - High variance of the estimator
- Large p reduces the interpretability of the model
  - Very important in many bioinformatics applications, but not only

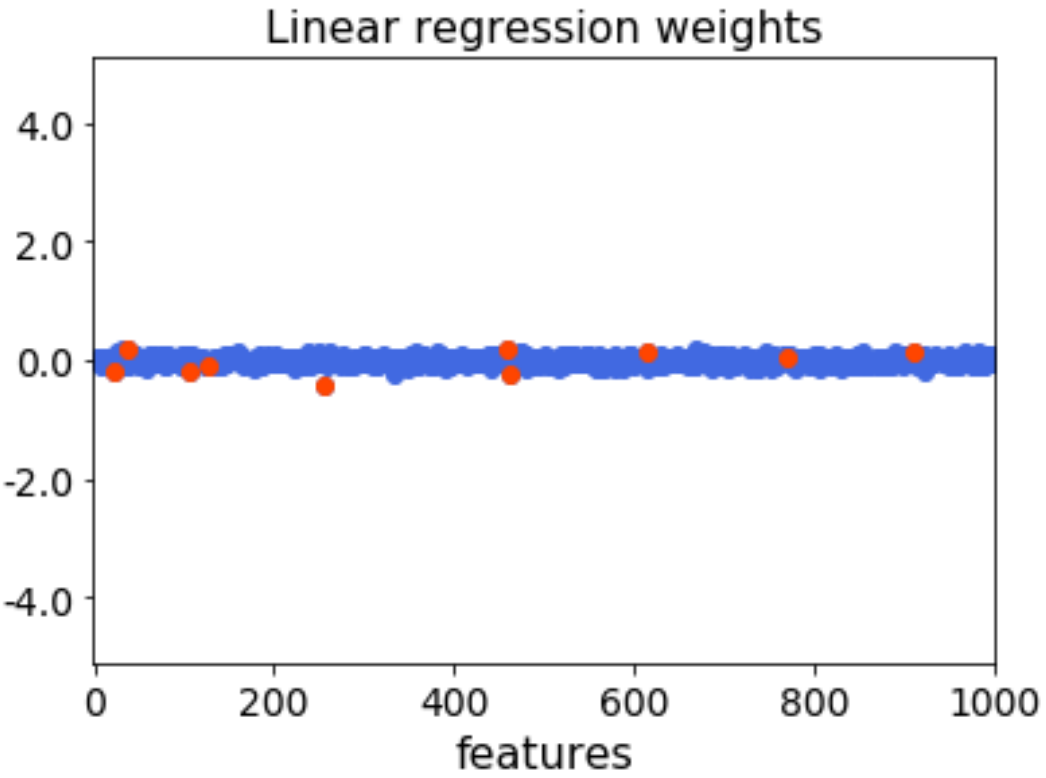
Linear regression when p >> n

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

True coefficients



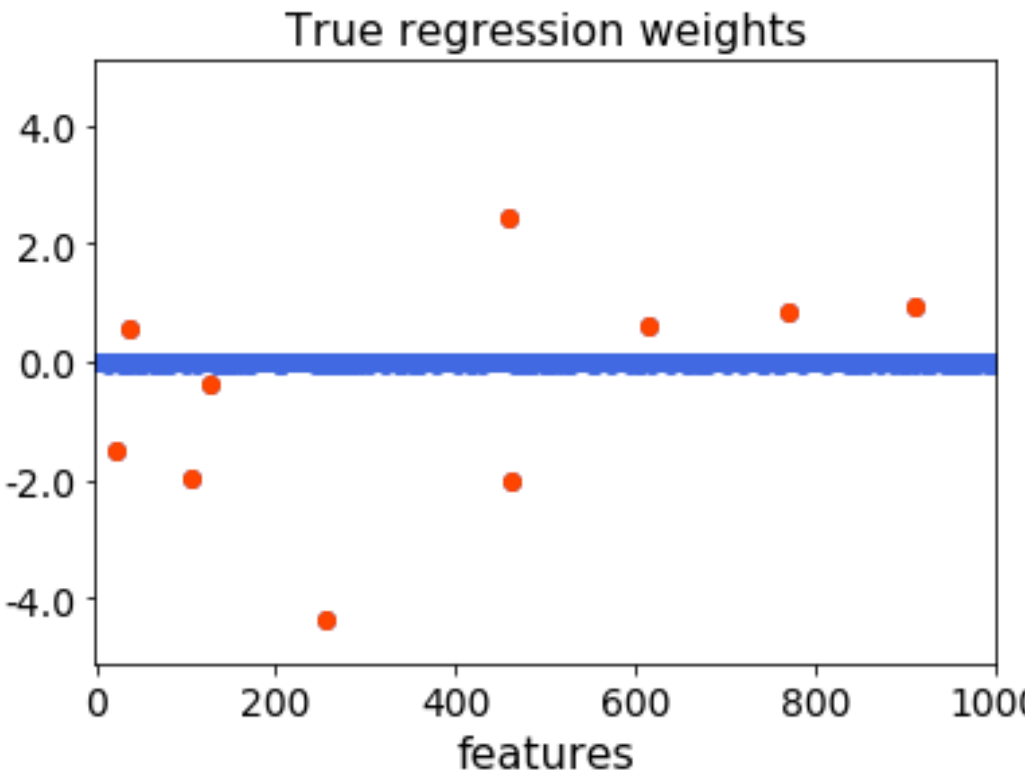
Predicted coefficients



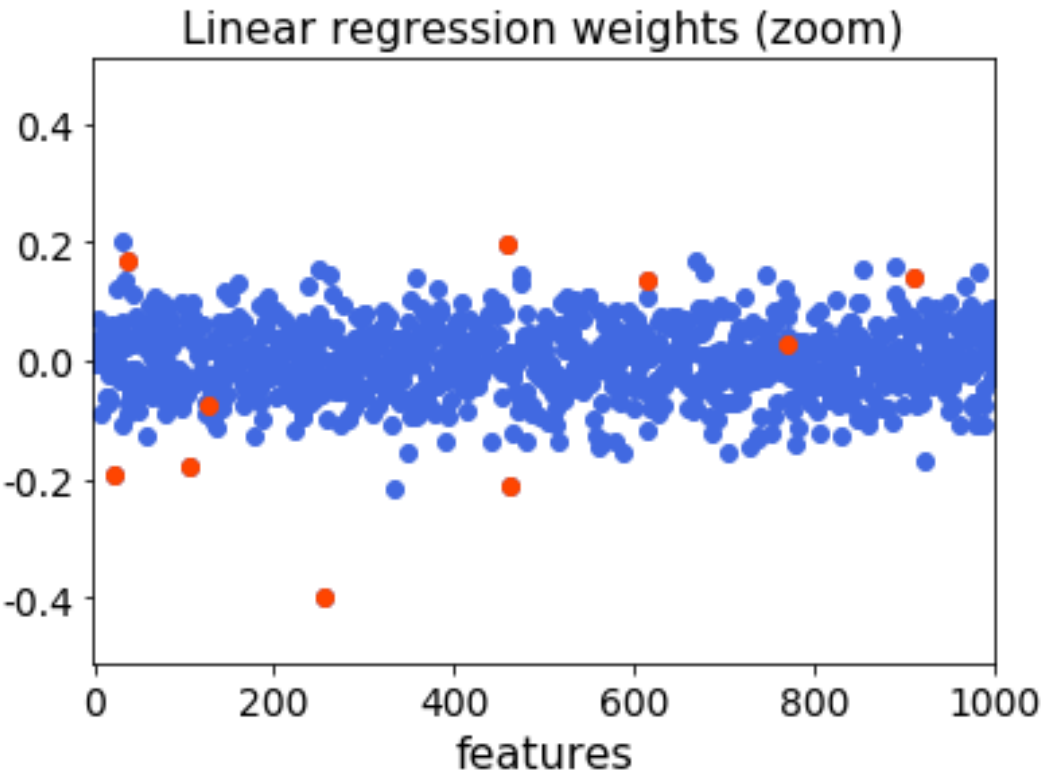
Linear regression when p >> n

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

True coefficients



Predicted coefficients



Regularization

Regularization

- Minimize

  Prediction error + penalty on model complexity
- **Biased estimator** when 0.
- Trade bias for a smaller variance.
- can be set by cross-validation.

- Simpler model fewer parameters

  shrinkage: drive the coeff cients of the parameters towards 0.

Ridge regression

Ridge regression

- **Sum-of-squares penalty**

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

- Equivalent to

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

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

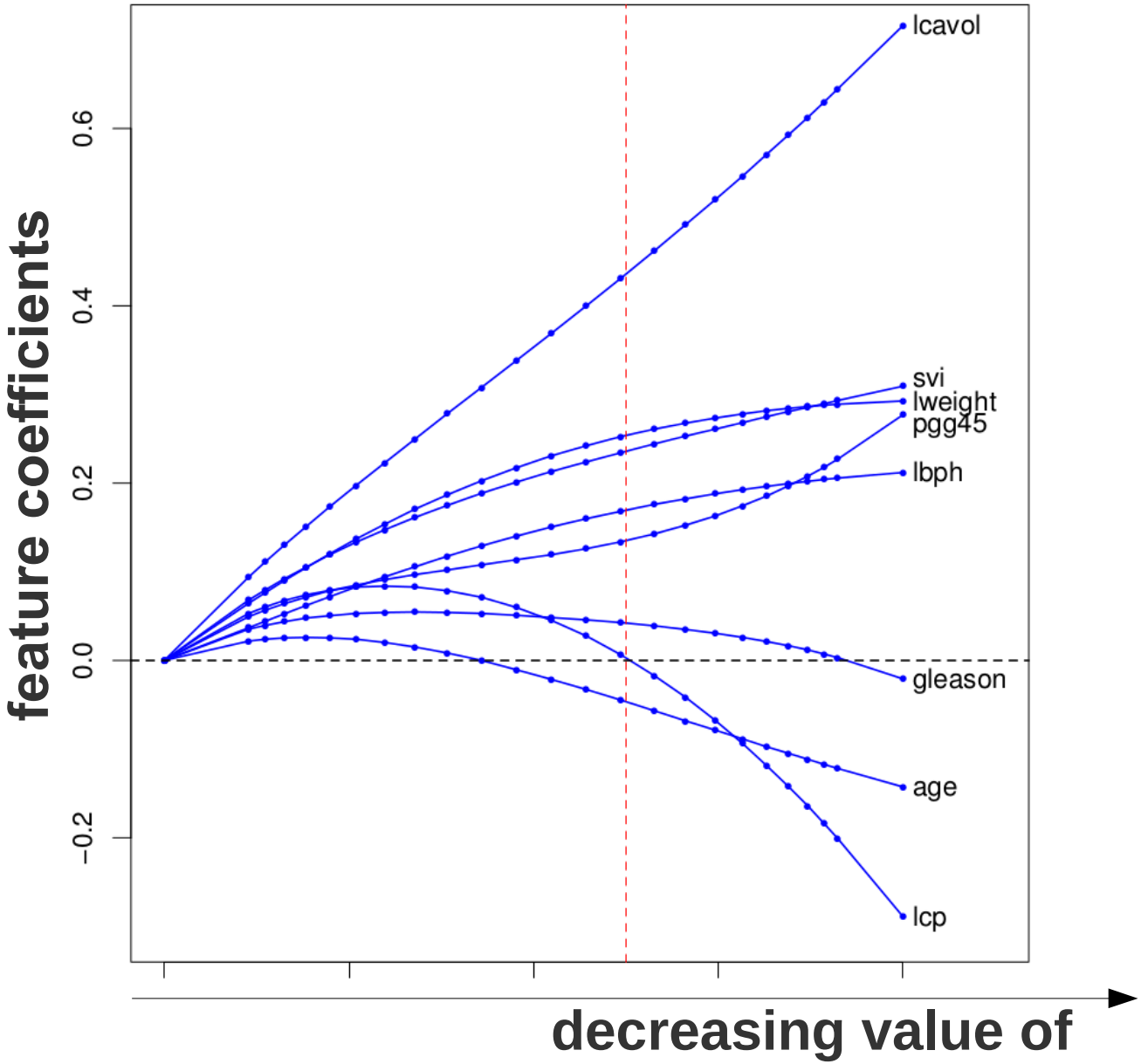
- **Ridge regression estimator:**

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

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

= always!

Ridge regression solution path



Standardization

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

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

  - For **ridge regression:**

    Not so clear, because of the penalization te$\lambda\beta_j^2$

- 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 coeff icients towards 0 but does not result in a **sparse model**.
- **Sparsity:**
  - many coeff icients 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**

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

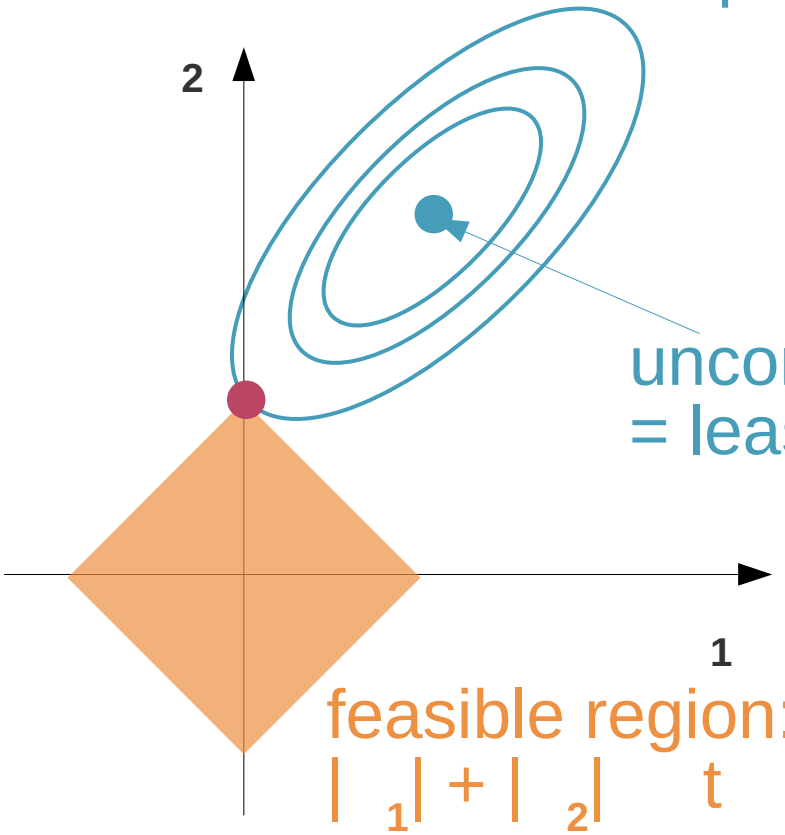
- aka **basis pursuit** (signal processing)
- no closed-form solution
- Equivalent to

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

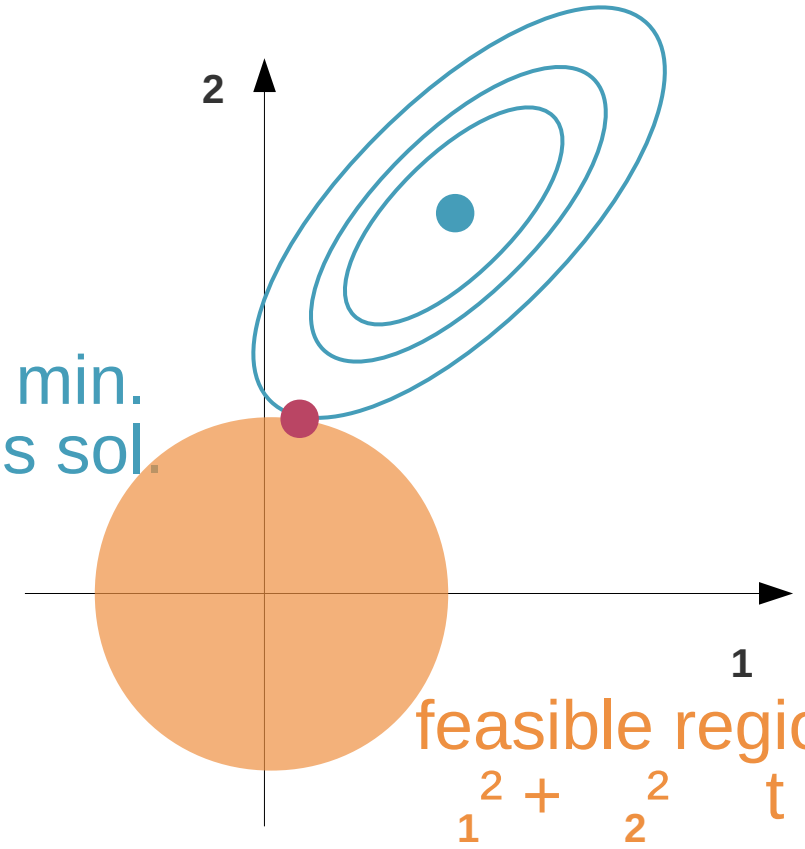
for a unique one-to-one match between t and .

Geometric interpretation

L1 norm

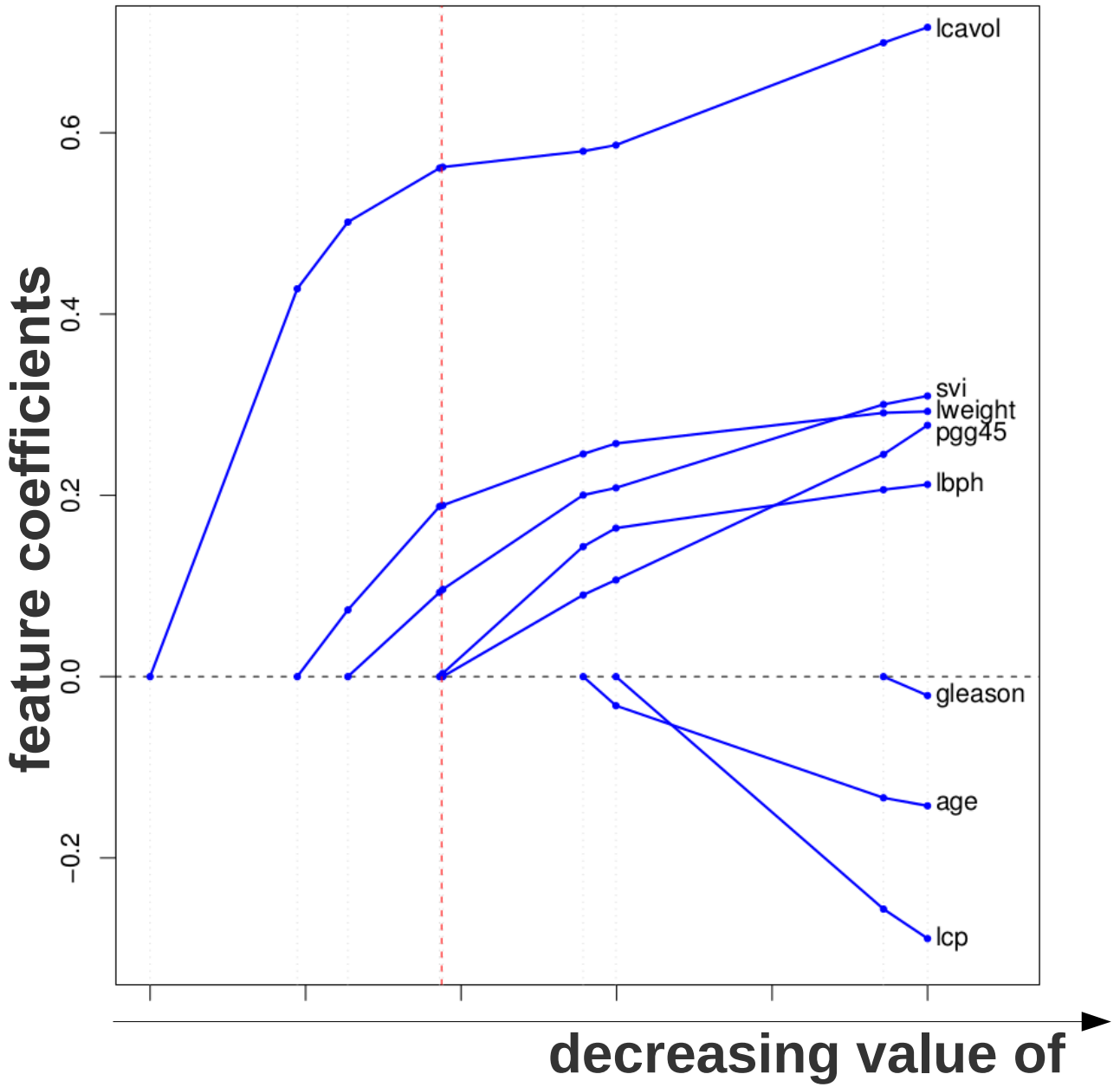


L2 norm



constrained minimum: where the iso-contrours of the error meet the feasible region. Because l1 balls are squares, this is more likely to happen on a corner, where some of the coordinates are 0.

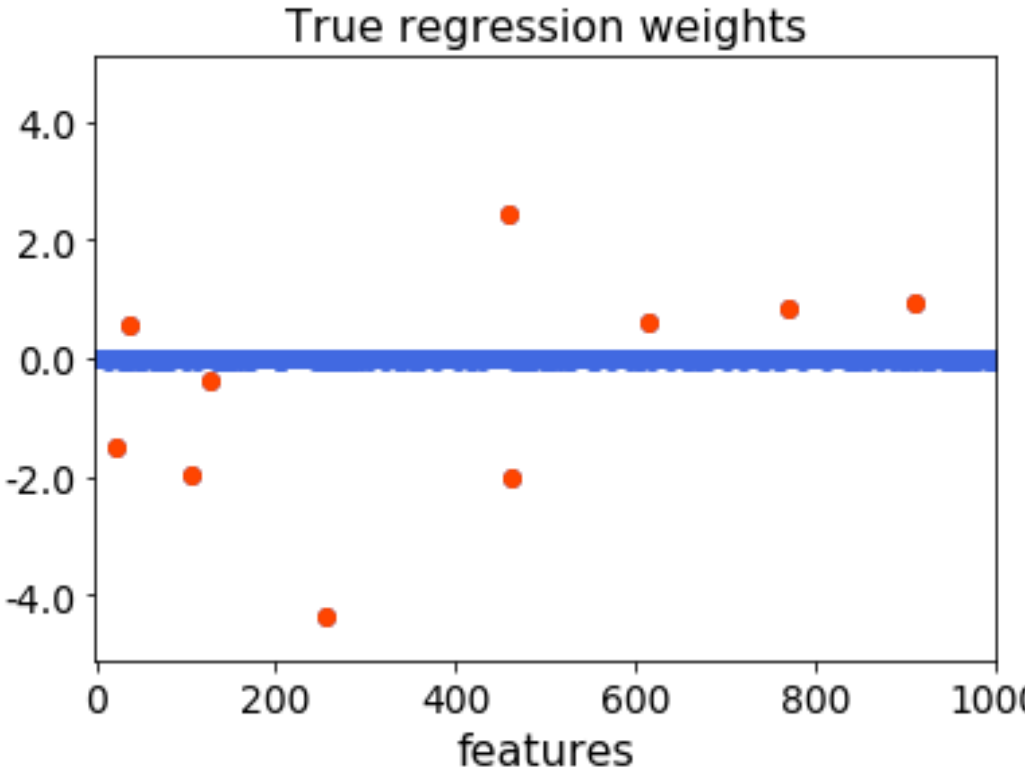
Lasso solution path



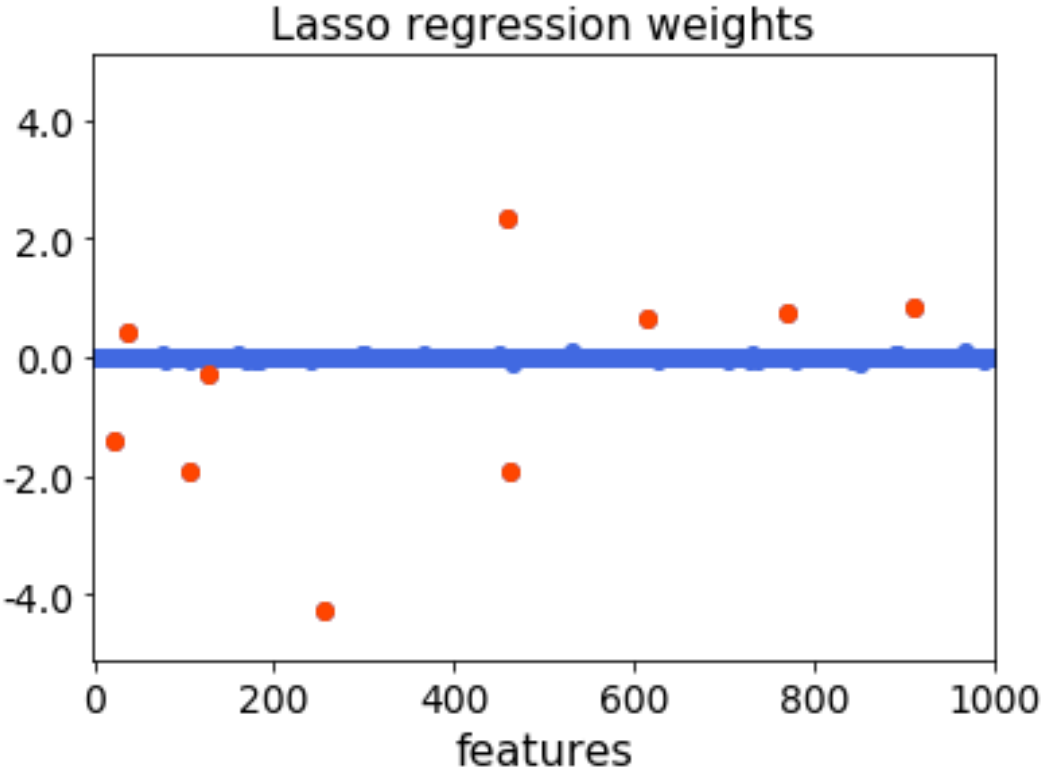
Linear regression when p >> n

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

True coefficients



Predicted coefficients



Elastic Net

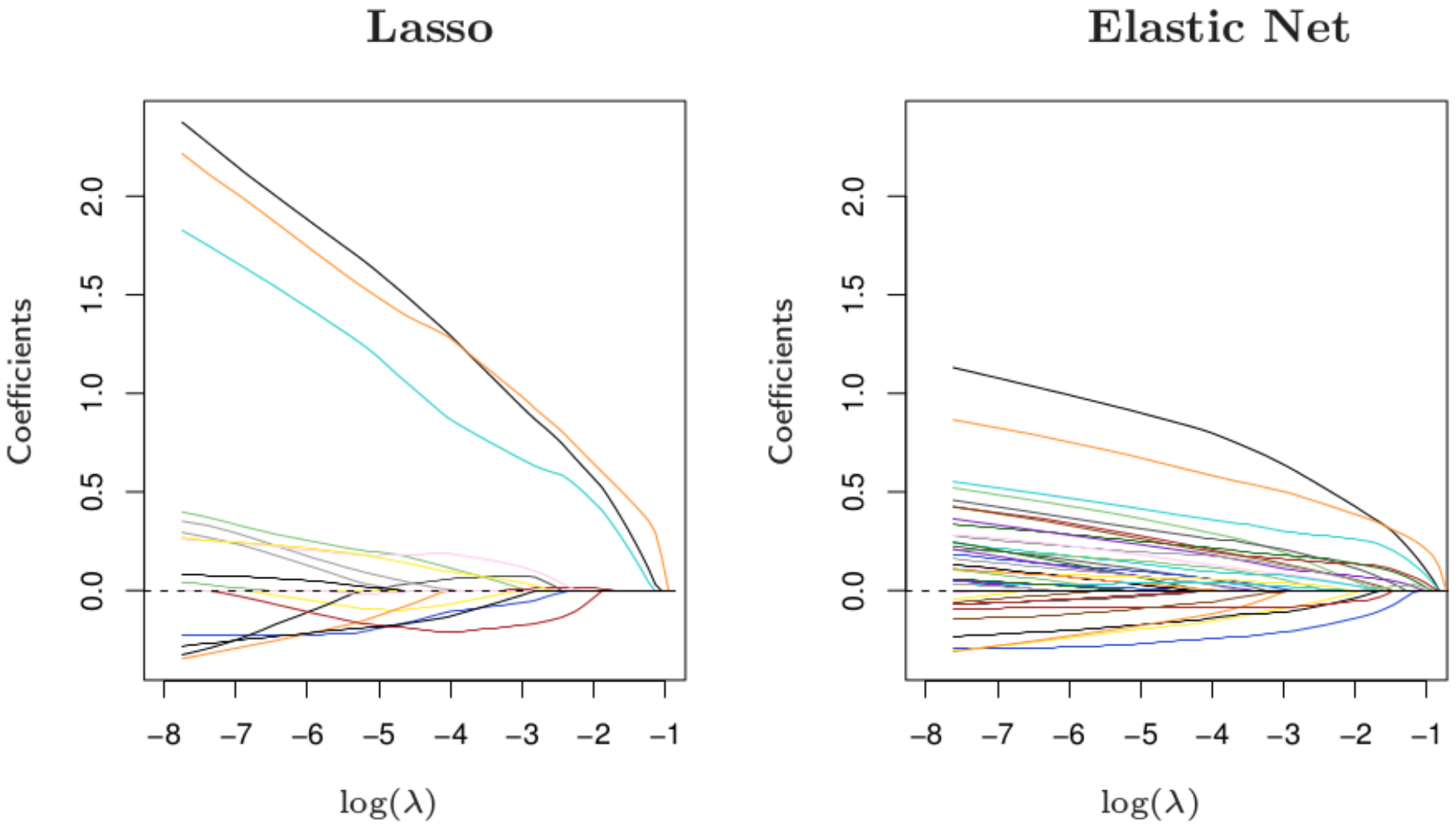
Elastic Net

- **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 coeff cients of correlated variables** like the ridge regression.

E.g. Leukemia data



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

Supervised machine learning in bioinformatics

- Quels problèmes ?
- Classification :
  - What is the **function of this gene?**
  - Is this **DNA sequence** a **micro-RNA?**
  - Does this blood sample come from a **diseased or a healthy individual?**
  - Is this **drug** appropriate for this patient?
  - What **side effects** could this new drug have?

Logistic regression

Classification

- Ridge regression:

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

- Uses the squared loss $\mathcal{L}(y^i, f(\boldsymbol{x}^i)) = \left(f(\boldsymbol{x}^i) - y^i\right)^2$
- For classification, use
  - a different decision/prediction function
  - a different loss

Classif ication

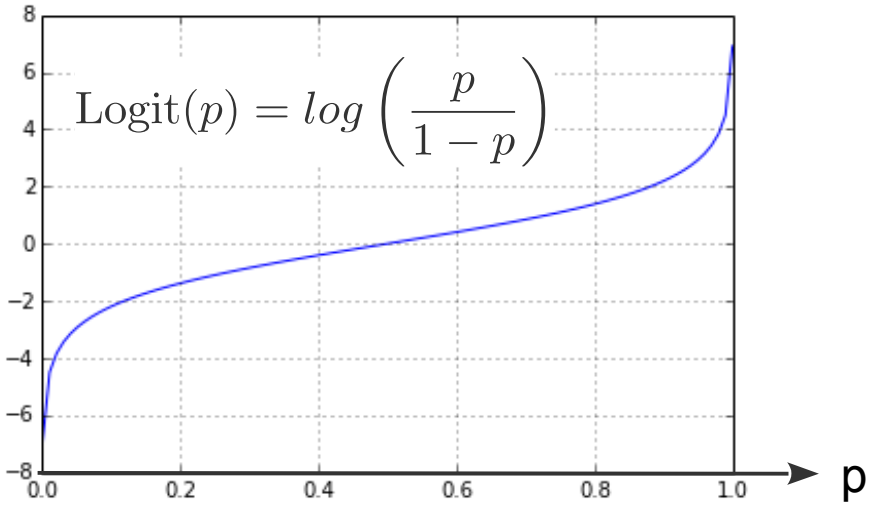
- Ridge regression:

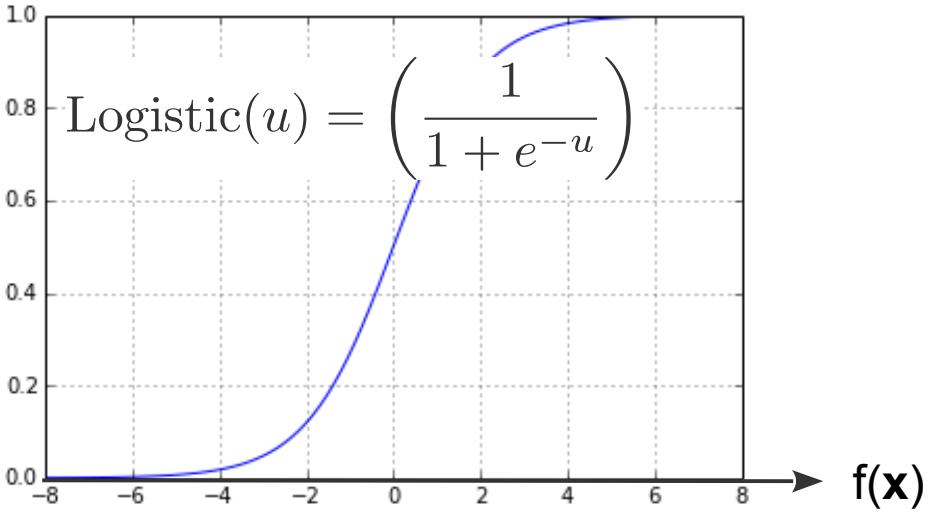
$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} ||y - X\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2$$

- Uses the squared loss $\mathcal{L}(y^i, f(\boldsymbol{x}^i)) = \left(f(\boldsymbol{x}^i) - y^i\right)^2$
- For classif ication, use
  - **a different decision function**
  - a different loss

Classif cation

- **Model P(Y=1|x) as a linear function?**
  - Problem: P(Y=1|**x**) **must be between 0 and 1.**
  - Use a **logit transformation**





Logistic regression.

$$\log \frac{P(y=1|\boldsymbol{x})}{1-P(y=1|\boldsymbol{x})} = \beta^\top \boldsymbol{x} + \beta_0 = f(\boldsymbol{x})$$

Classif ication

- Ridge regression:

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

- Uses the squared loss $\mathcal{L}(y^i, f(\boldsymbol{x}^i)) = \left(f(\boldsymbol{x}^i) - y^i\right)^2$
- For classif ication, use
  - a different decision function $\log \frac{P(y=1|\boldsymbol{x})}{1-P(y=1|\boldsymbol{x})} = \beta^\top \boldsymbol{x} + \beta_0 = f(\boldsymbol{x})$
  - **a different loss**

Classification

- Ridge regression:

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

- Uses the squared loss $\mathcal{L}(y^i, f(\boldsymbol{x}^i)) = \left(f(\boldsymbol{x}^i) - y^i\right)^2$
- For classification, use
  - a different decision function $\log \frac{P(y=1|\boldsymbol{x})}{1-P(y=1|\boldsymbol{x})} = \beta^\top \boldsymbol{x} + \beta_0 = f(\boldsymbol{x})$
  - **a different loss:** the **logistic loss**

$$\mathcal{L}(y^i, f(\boldsymbol{x}^i)) = \log\left(1 + \exp(-y^i f(\boldsymbol{x}^i))\right)$$

$$y \in \{-1, +1\}$$

Classification

- Ridge regression:

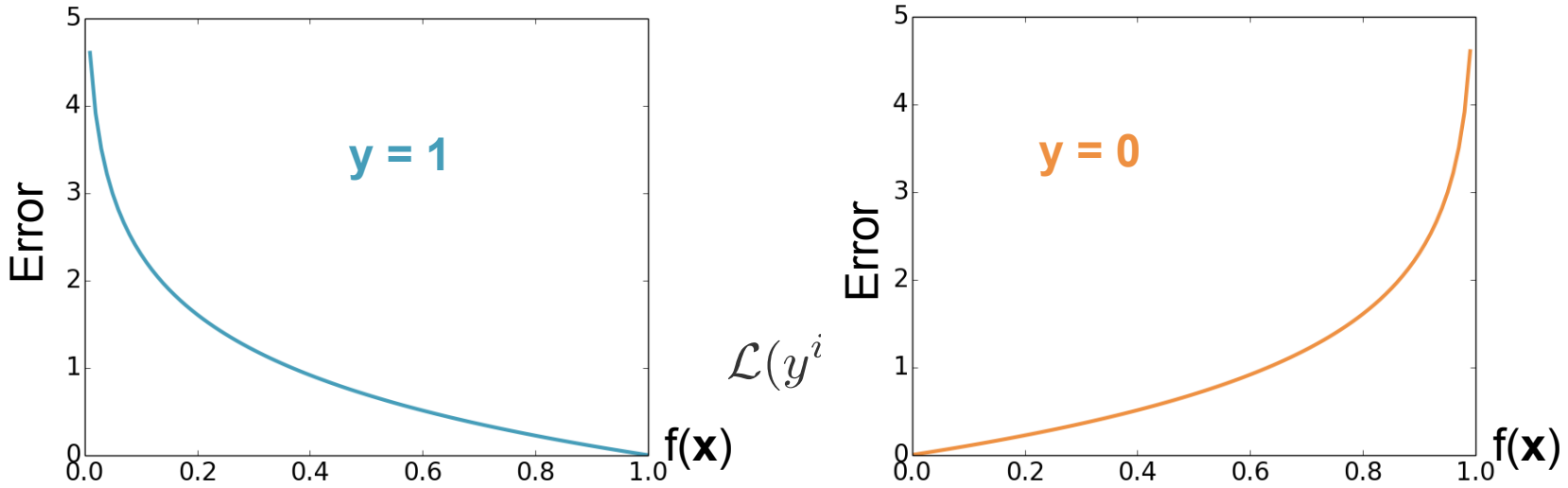
$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} ||y - X\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2$$

- Uses the squared loss $\mathcal{L}(y^i, f(\boldsymbol{x}^i)) = \left(f(\boldsymbol{x}^i) - y^i\right)^2$
- For classification, use
  - a different decision function $\log \frac{P(y=1|\boldsymbol{x})}{1-P(y=1|\boldsymbol{x})} = \beta^\top \boldsymbol{x} + \beta_0 = f(\boldsymbol{x})$
  - **a different loss:** the **logistic loss**

$$\mathcal{L}(y^i, f(\boldsymbol{x}^i)) = \log\left(1 + \exp(-y^i f(\boldsymbol{x}^i))\right)$$

$$y \in \{-1, +1\}$$

Classif cation



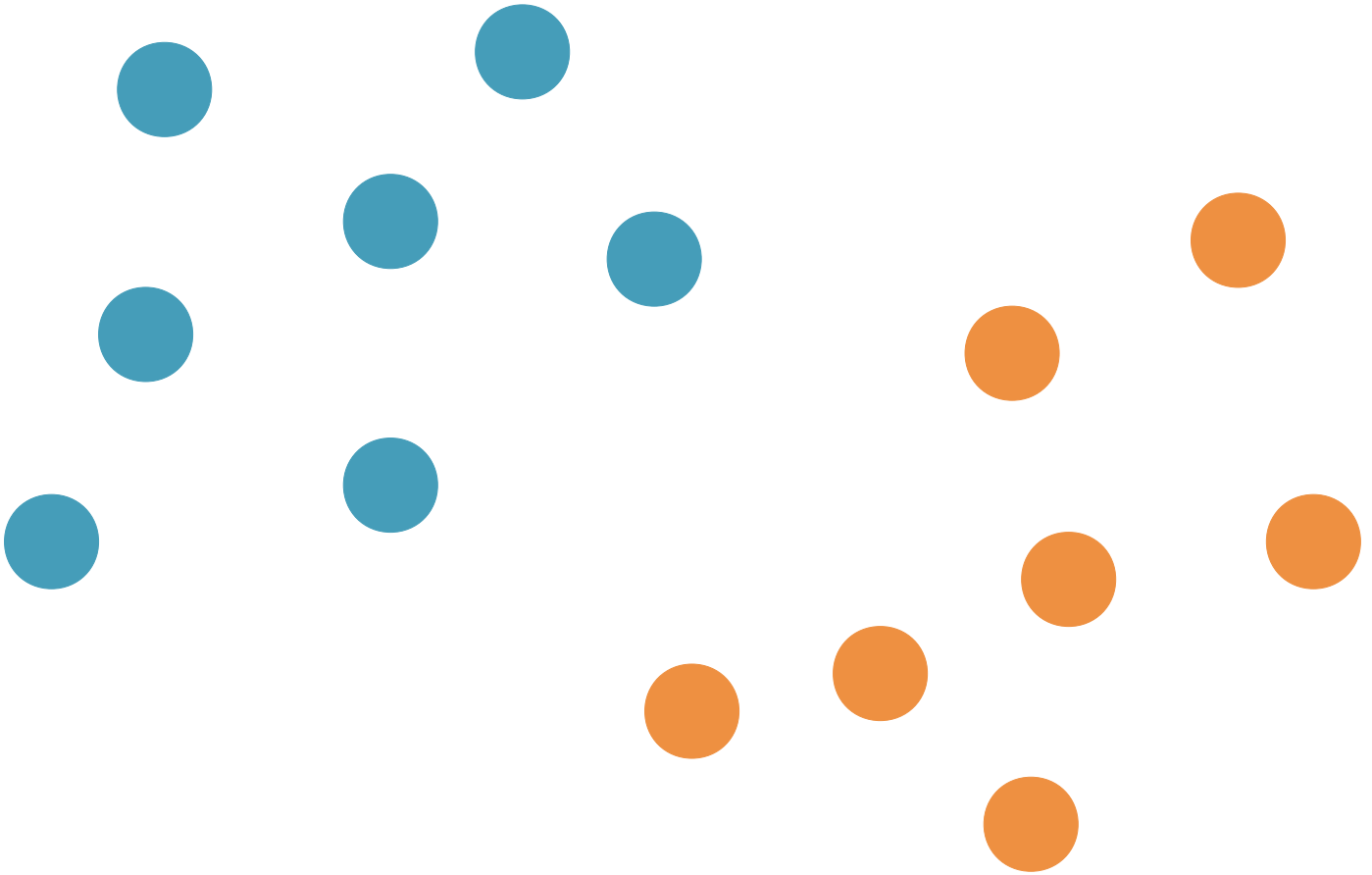
- a different decision function $\log \frac{P(y=1|\boldsymbol{x})}{1-P(y=1|\boldsymbol{x})} = \beta^\top \boldsymbol{x} + \beta_0 = f(\boldsymbol{x})$
- **a different loss:** the **logistic loss**

$$\mathcal{L}(y^i, f(\boldsymbol{x}^i)) = \log\left(1 + \exp(-y^i f(\boldsymbol{x}^i))\right)$$

$$y \in \{-1, +1\}$$

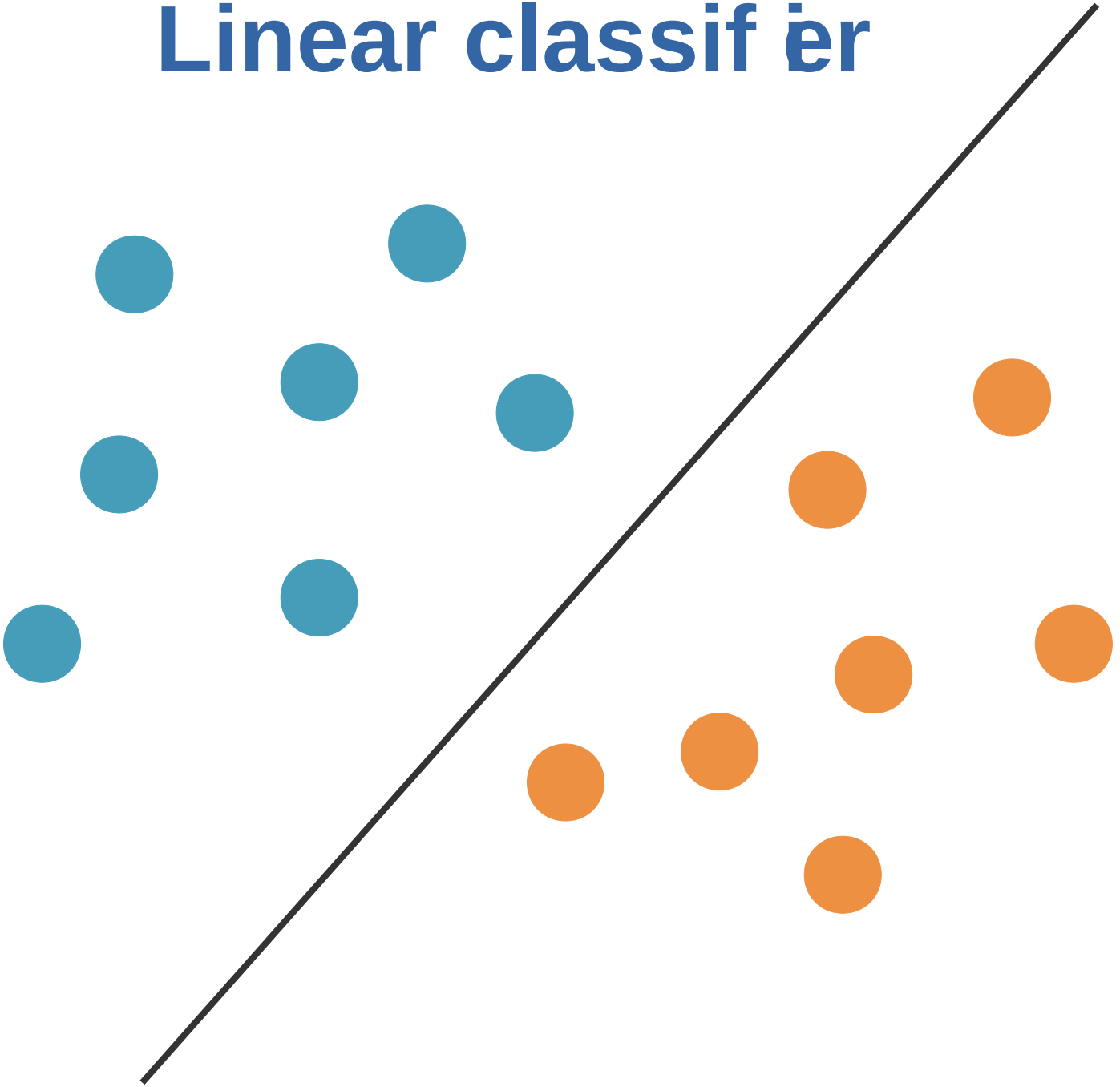
Support vector machines

Linear classifier

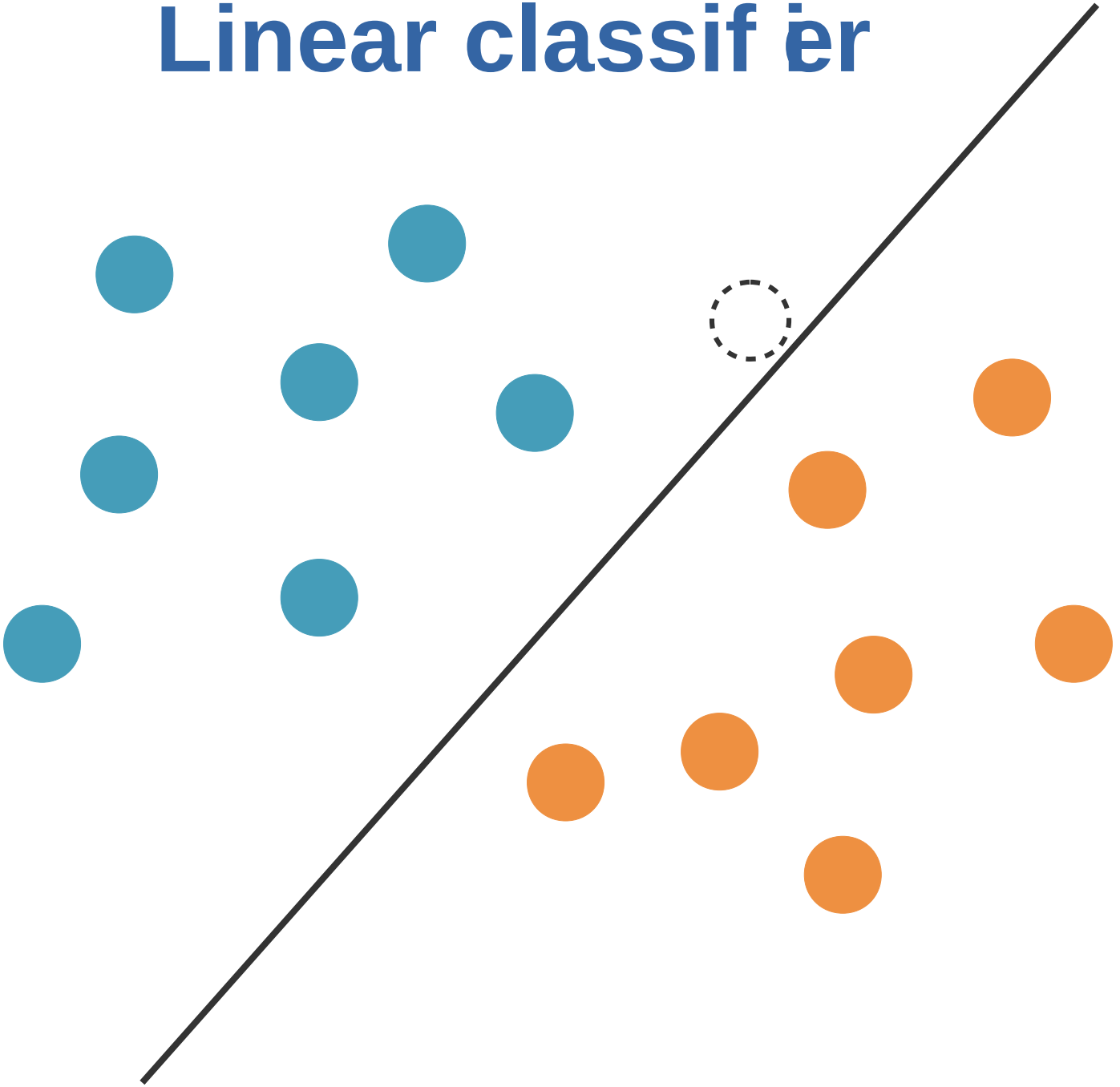


Assume data is **linearly separable**:
there exists a line that separates + from -

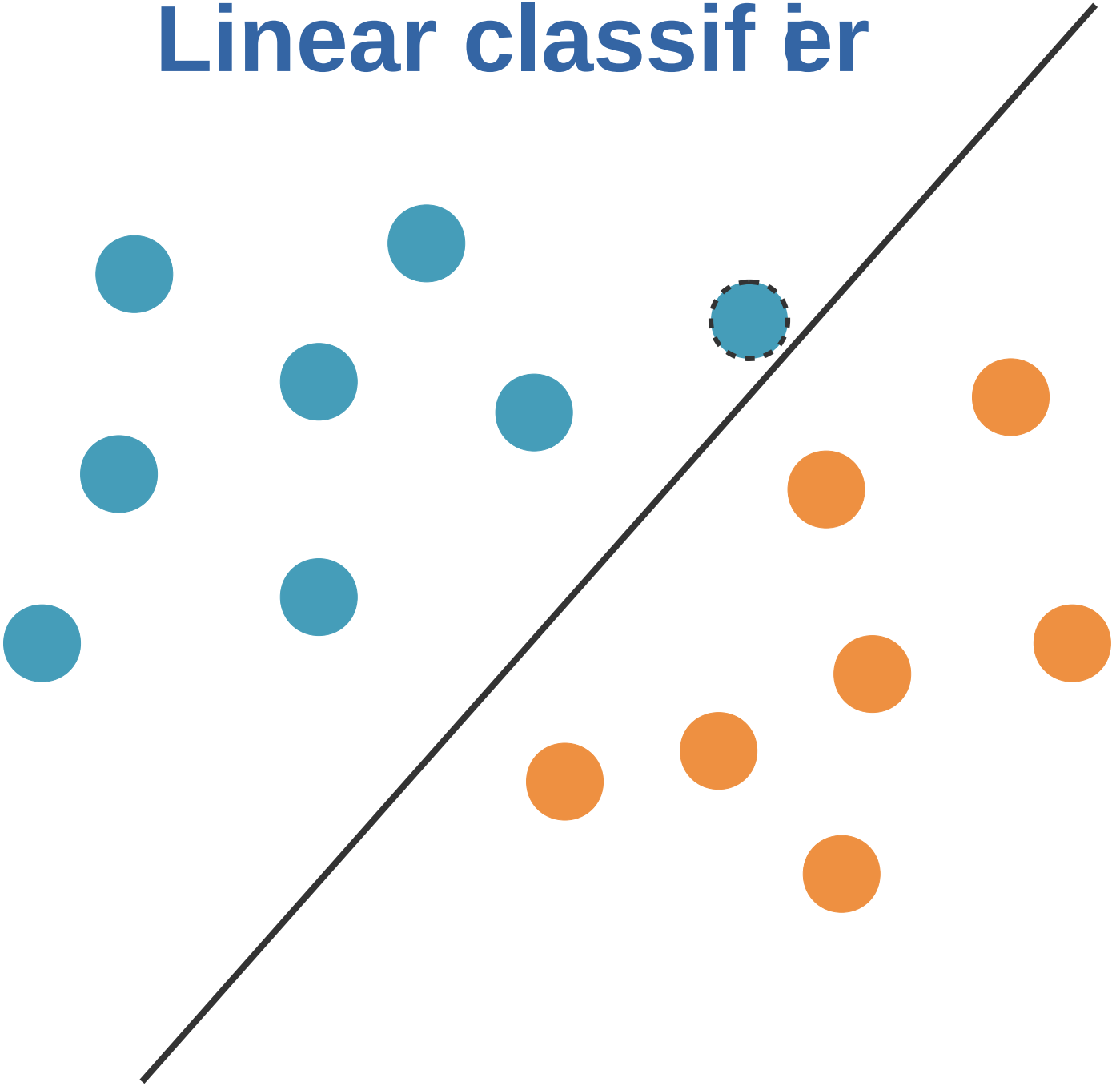
Linear classifier



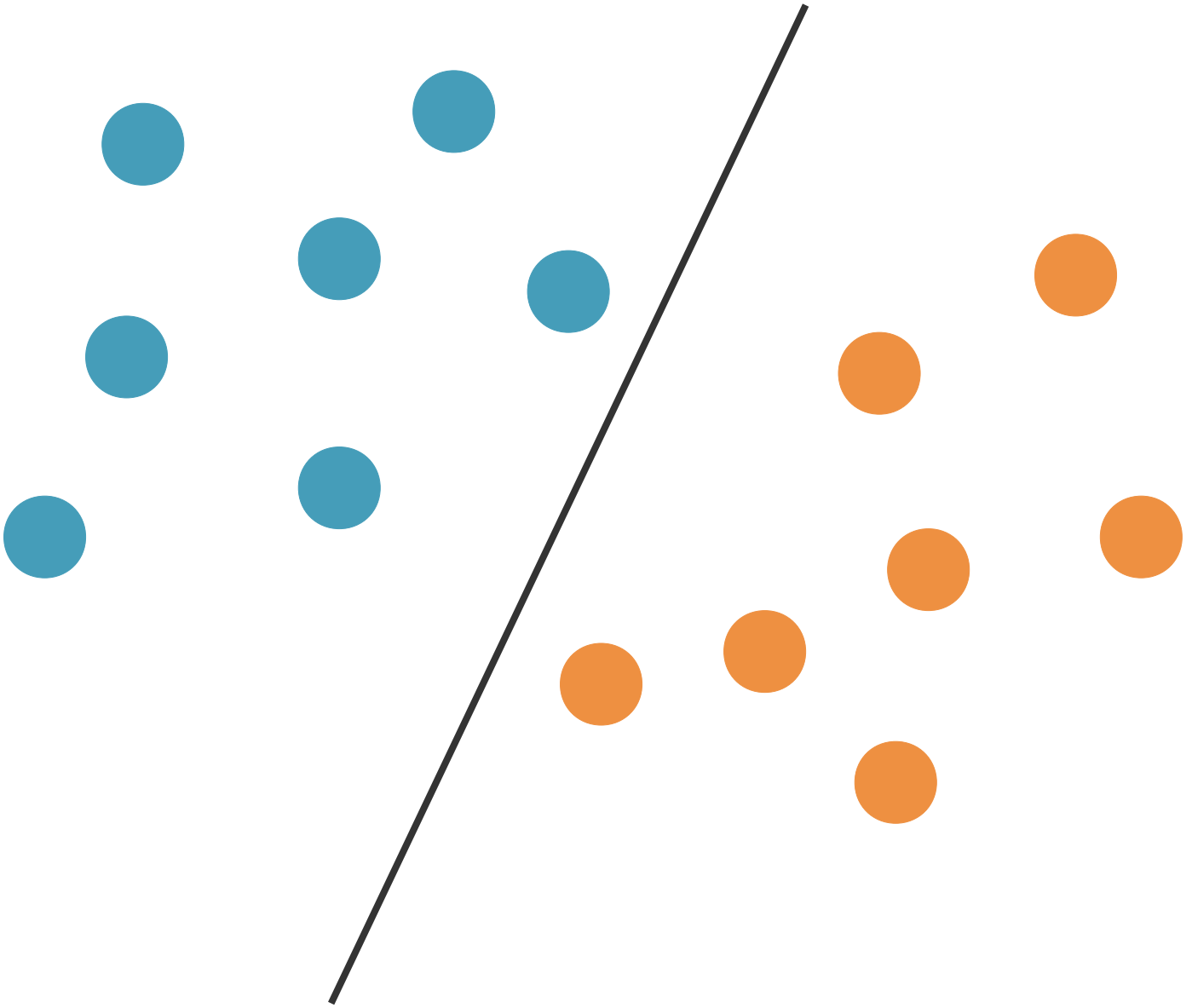
Linear classifier



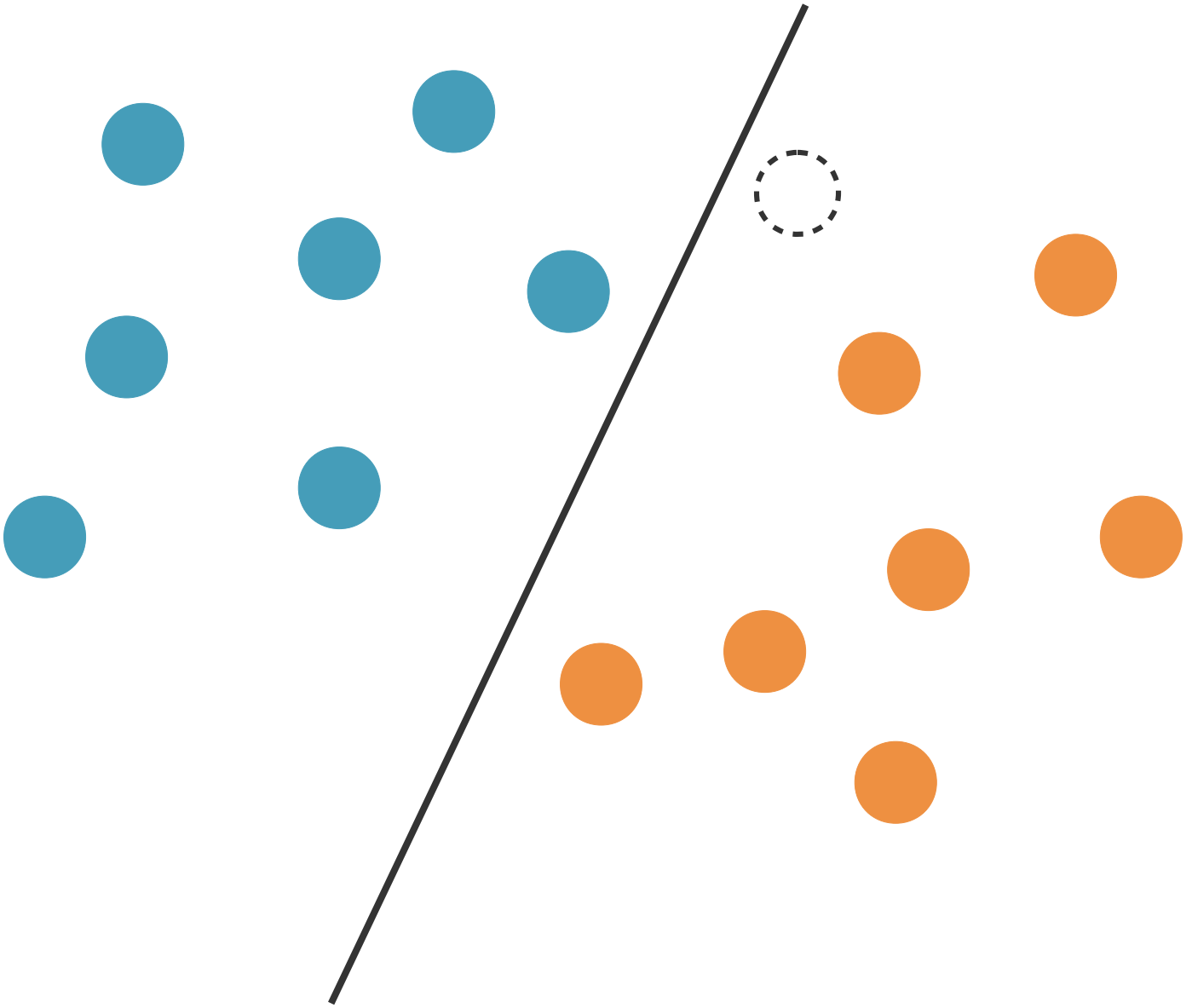
Linear classifier



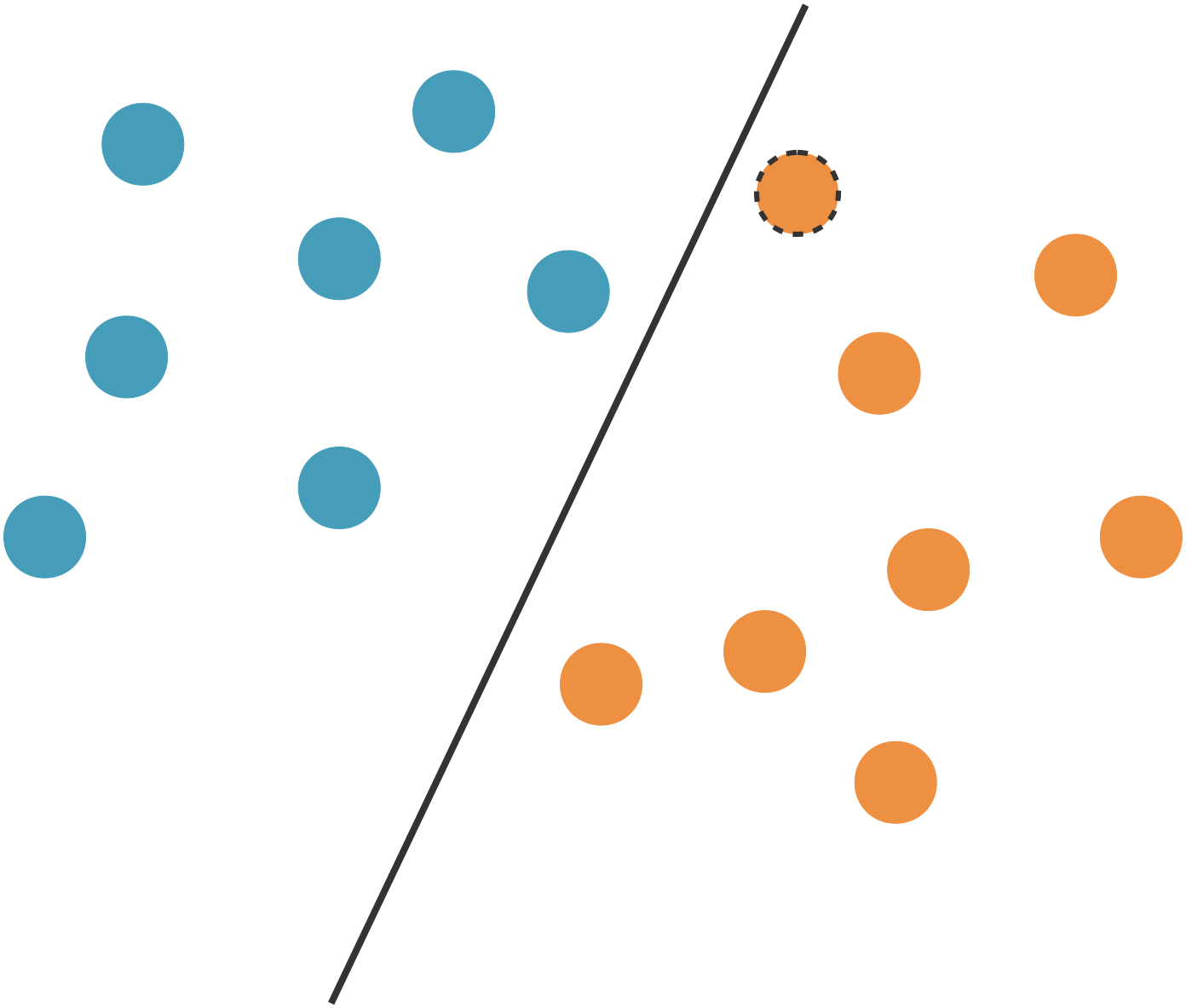
Linear classifier



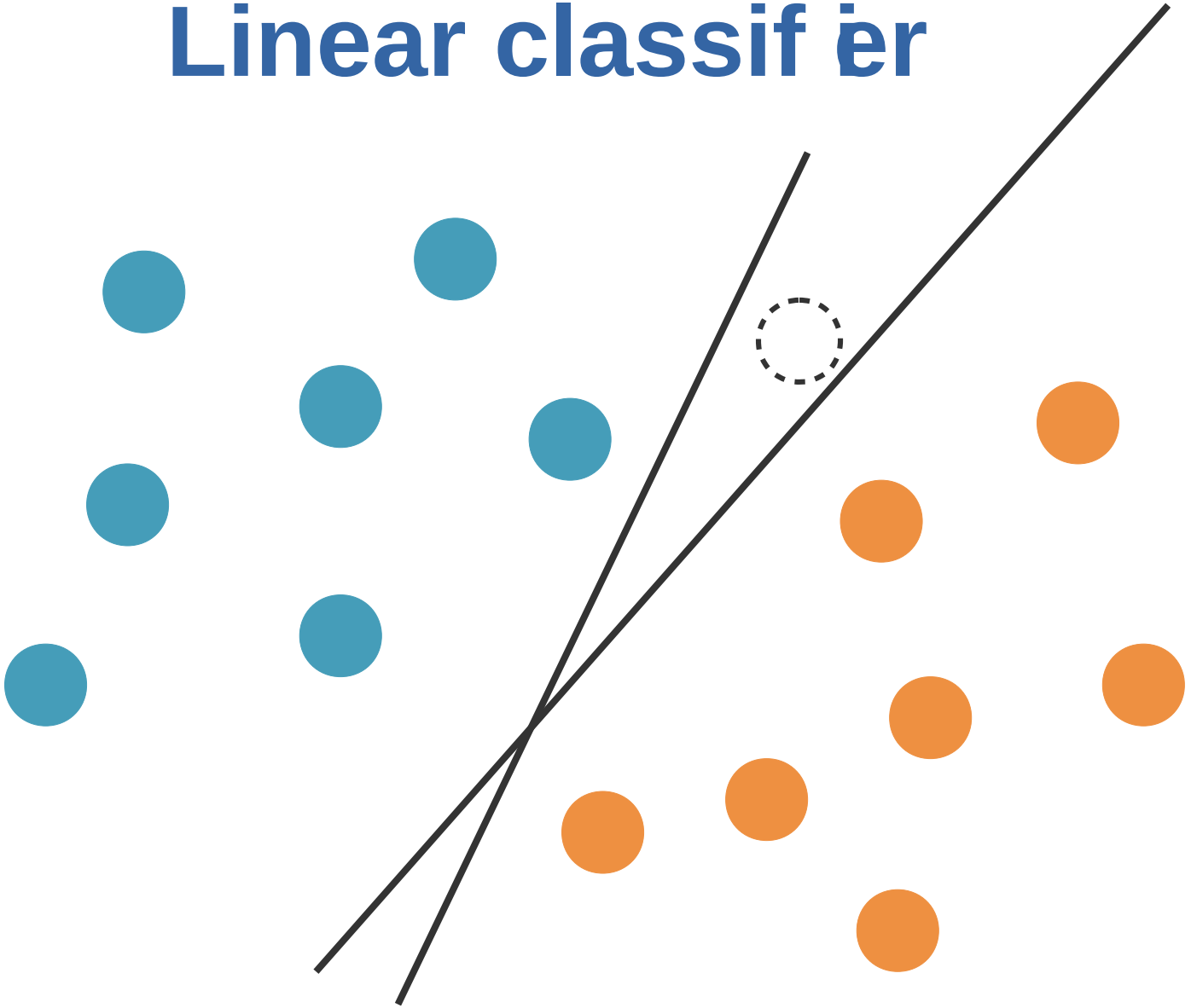
Linear classifier



Linear classifier

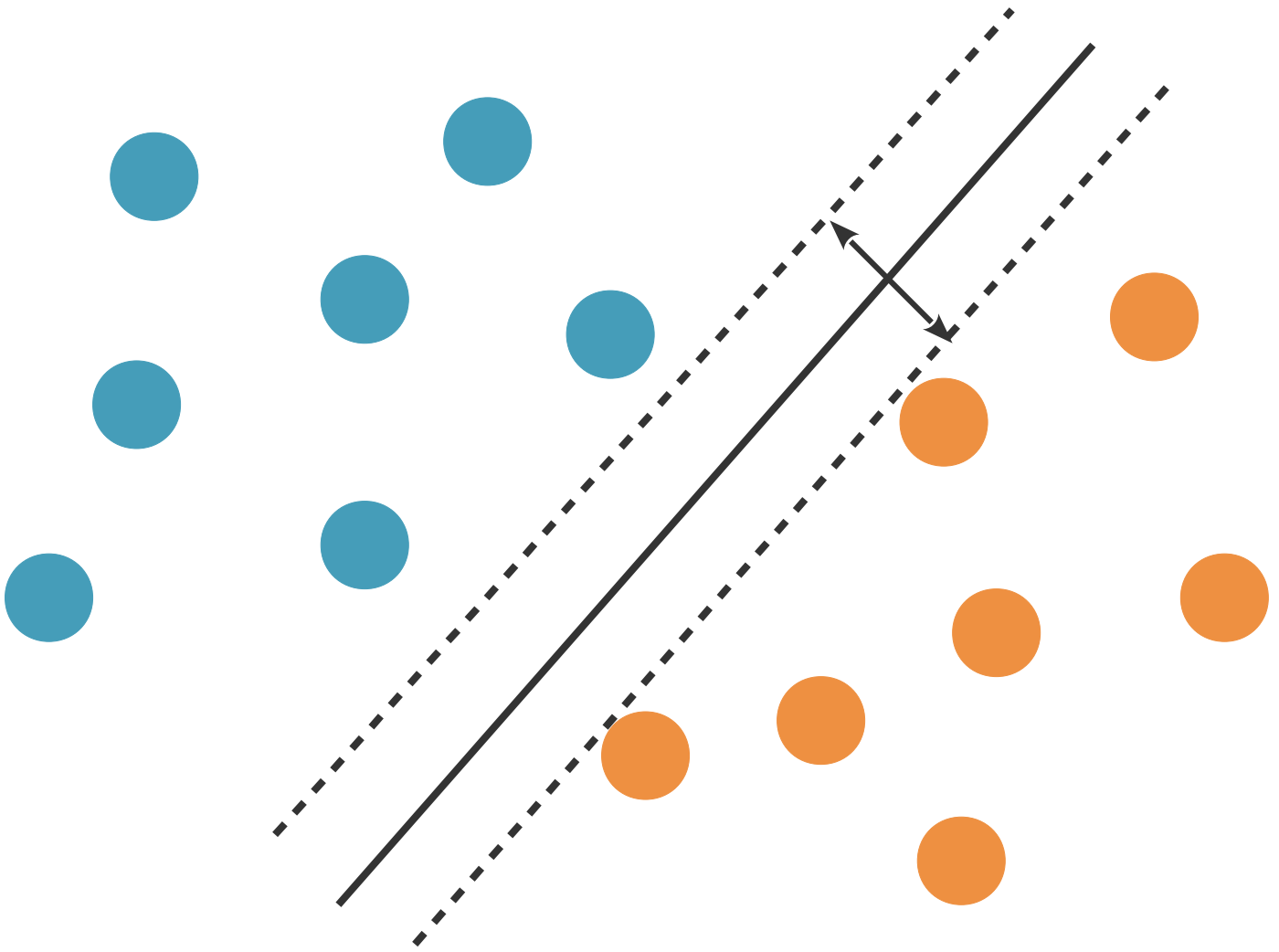


Linear classifier



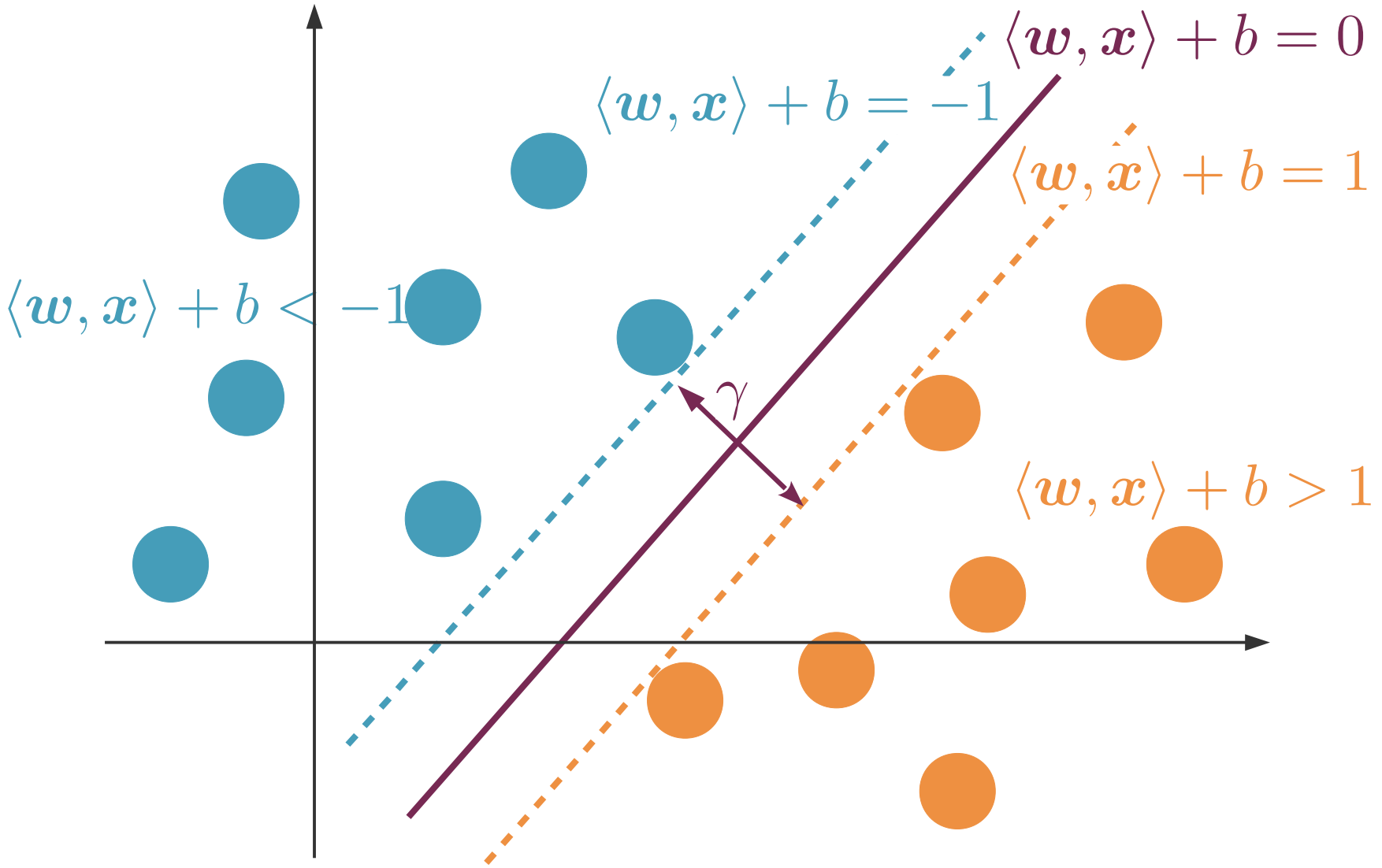
Which one is better?

Margin of a linear classifier



Margin: Twice the distance from the separating hyperplane to the closest training point.

Largest margin hyperplane



Classification of the training points

- We want the training points to be on the correct side of the "road" defined by the separating hyperplane + margin
- **Correct classification of the training points:**
  - For positive examples:

$$y^i = 1 \text{ and } \langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b \geq 1$$

  - For negative examples:

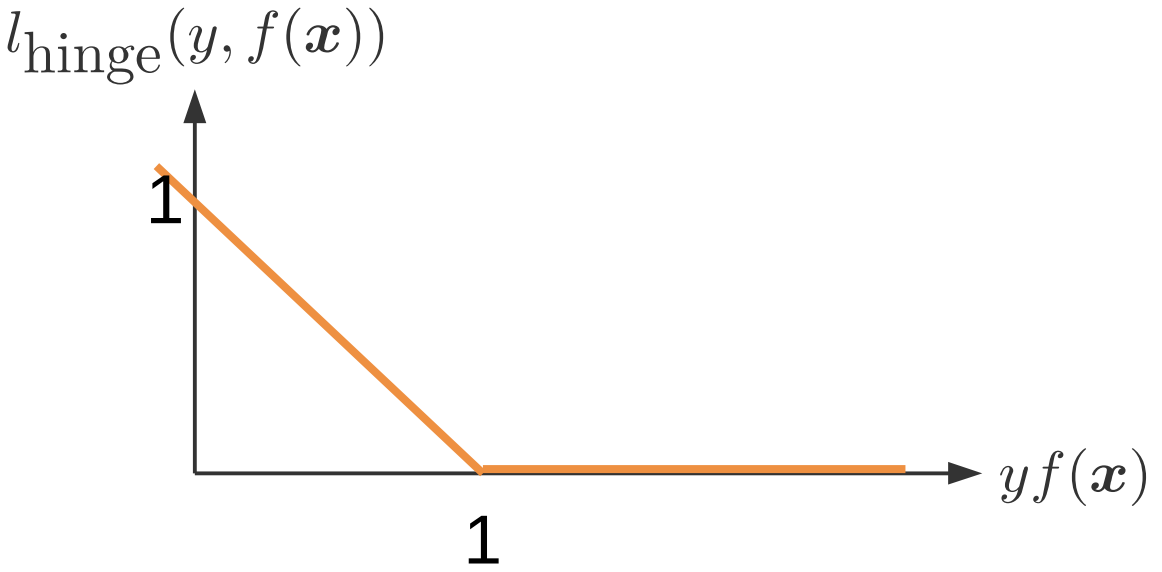
$$y^i = -1 \text{ and } \langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b \leq -1$$

  - Summarized as $y^i.(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) \geq 1$

Hinge loss

- We want for all i: $y^i f(\boldsymbol{x}^i) \geq 1$
- **Hinge loss** function:

$$
\begin{aligned}
l_{\text{hinge}}(u, y) &= \max(1 - yu, 0) \\
&= \begin{cases} 0 & \text{if } yu \geq 1 \\ 1 - yu & \text{otherwise} \end{cases}
\end{aligned}
$$



Classification

- Ridge regression:

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

- Uses the squared loss
- For classification, use a different loss
  - Hinge loss:

$$\max(0, 1 - y(\langle \boldsymbol{\beta}, \boldsymbol{x} \rangle + \beta_0))$$

Classification

- Ridge regression:

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

- Uses the squared loss
- For classification, use a different loss
  - Hinge loss:

$$\max\left(0, 1 - y(\langle \boldsymbol{\beta}, \boldsymbol{x}\rangle + \beta_0)\right)$$

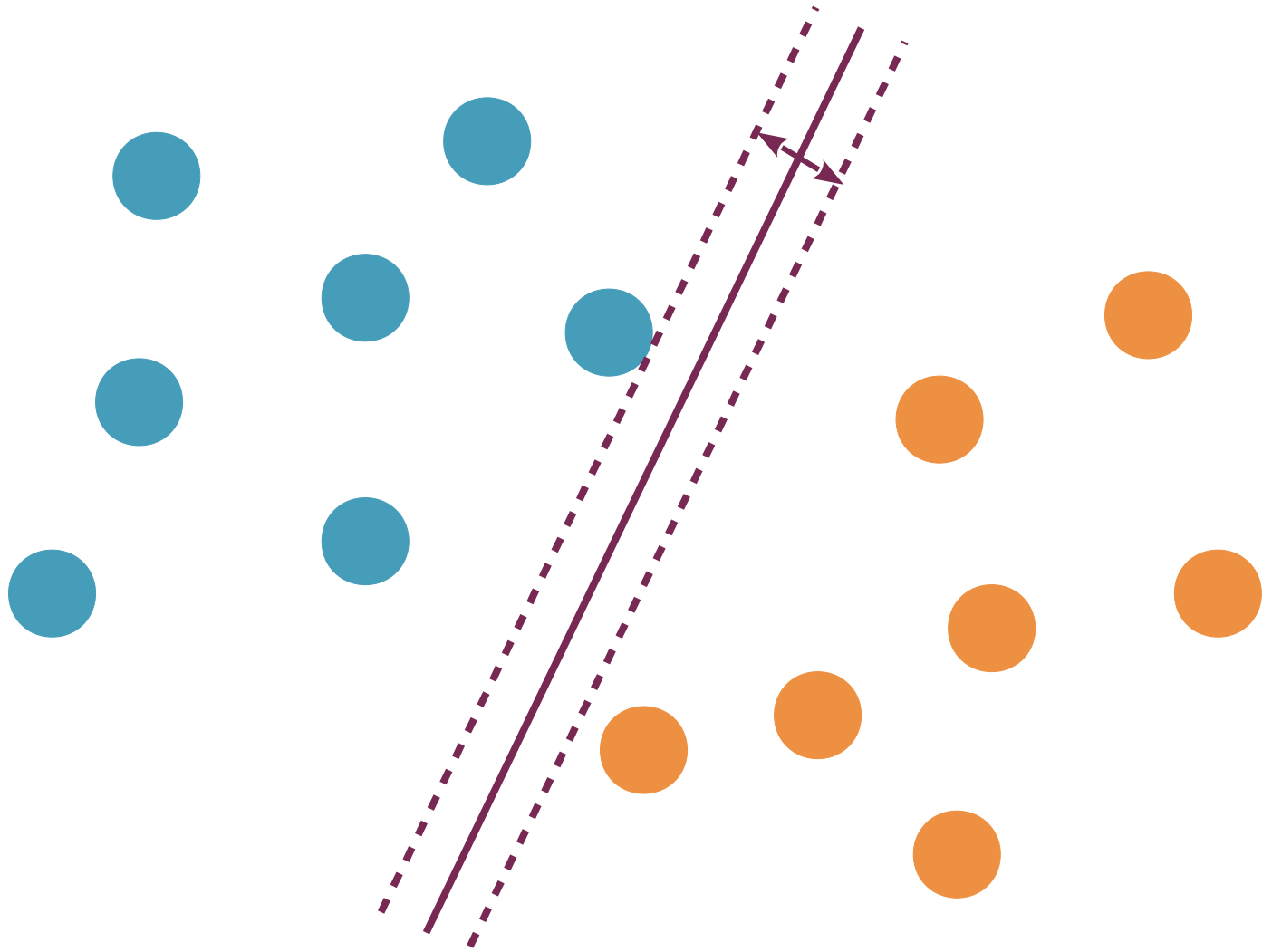
  - **Support Vector Machine:**

$$\arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \left(\frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{n} \max(0, 1 - y^i(\langle \boldsymbol{\beta}, \boldsymbol{x}^i\rangle + \beta_0)) \right)$$

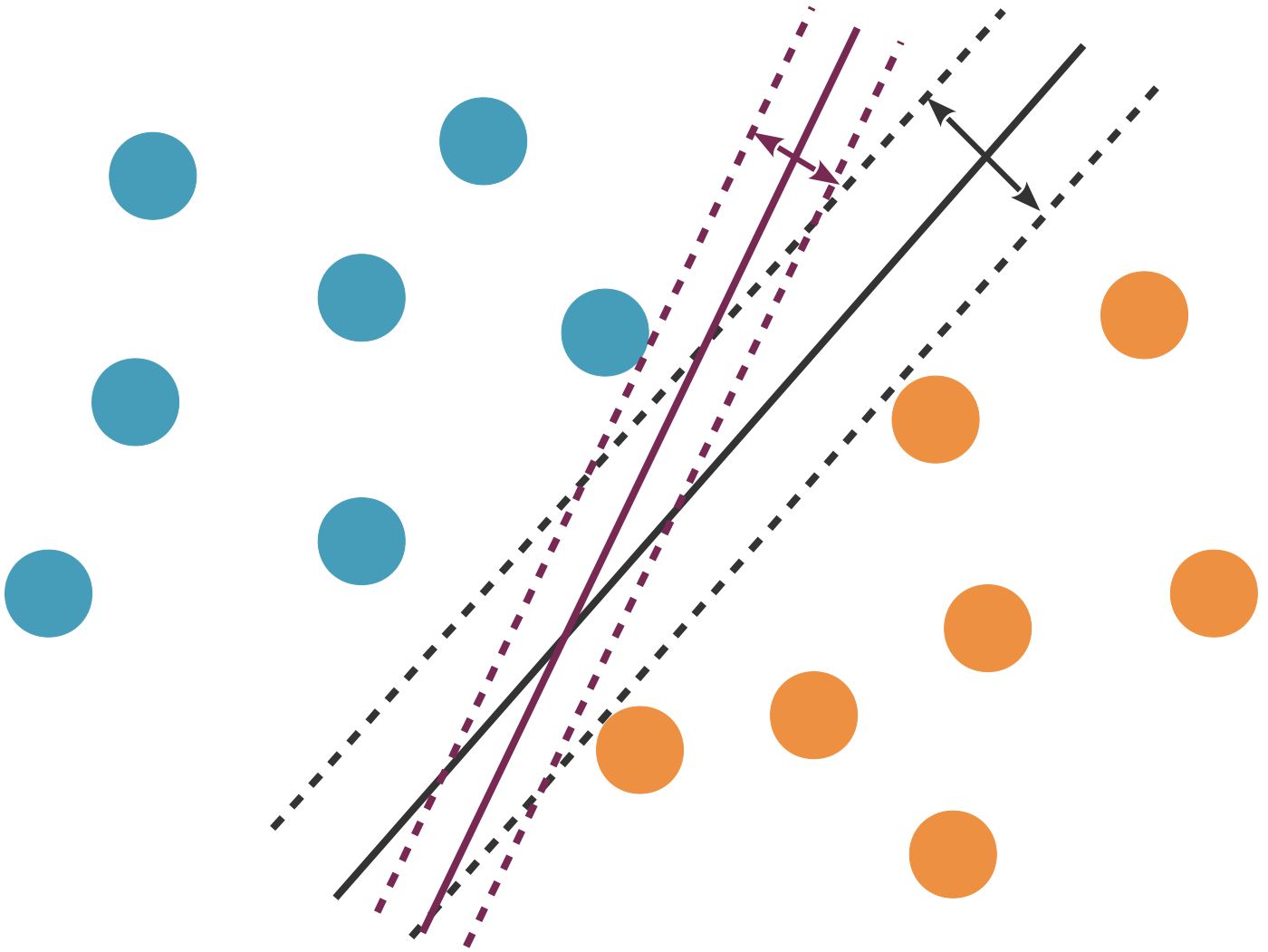
Large margin classifier

- Minimizing the l2 norm of the regression coefficient is equivalent to maximizing the margin

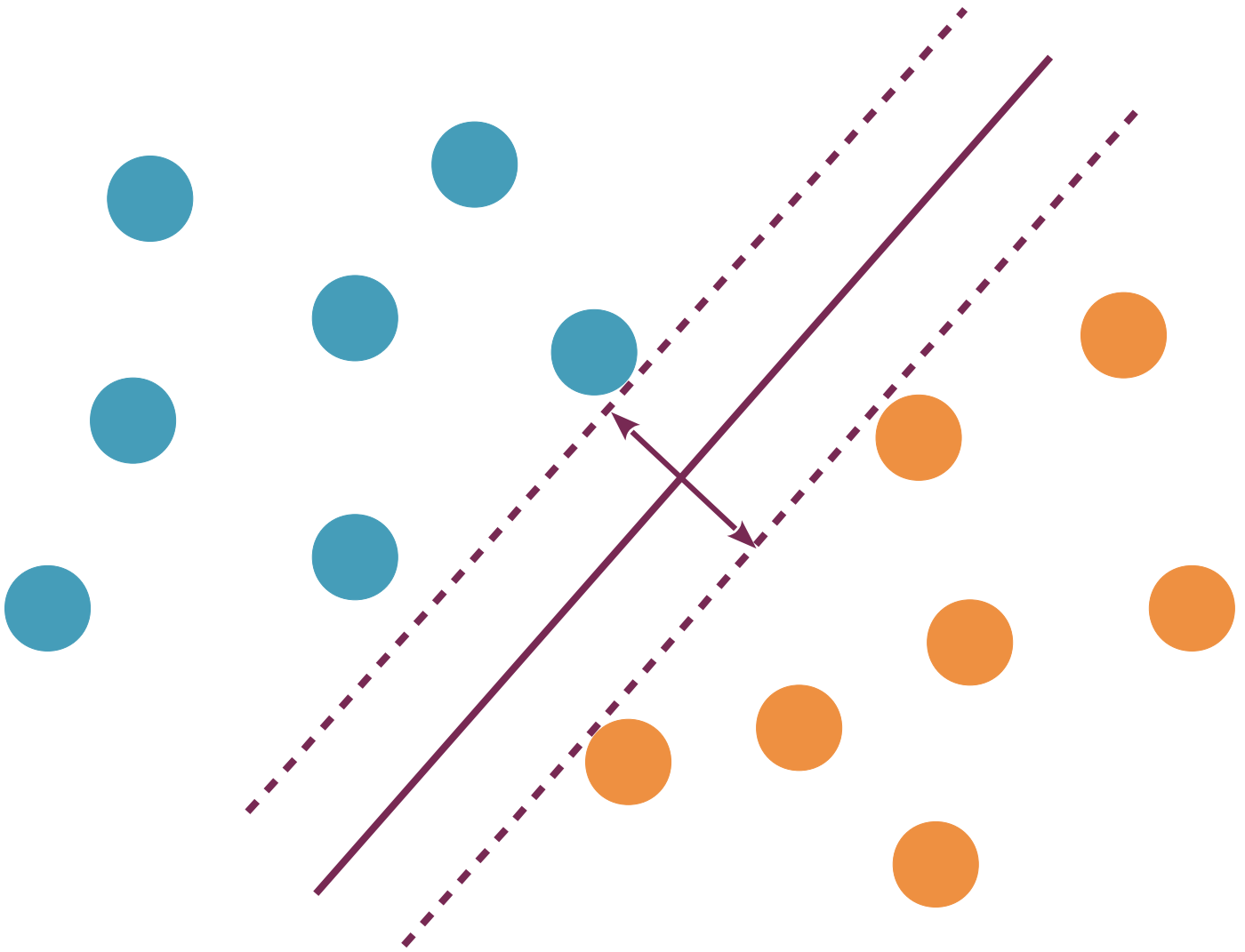
Margin of a linear classifier



Margin of a linear classifier



Largest margin classifier: Support vector machines



Dual formulation of the SVM

- Equivalently, maximize

$$q(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^i y^j \left\langle \boldsymbol{x}^i, \boldsymbol{x}^j \right\rangle$$

- under the constraints

$$\begin{cases} 0 \leq \alpha_i \leq C & \text{for } i = 1, \ldots, n \\ \sum_{i=1}^{n} \alpha_i y^i = 0 \end{cases}$$

From optimization theory

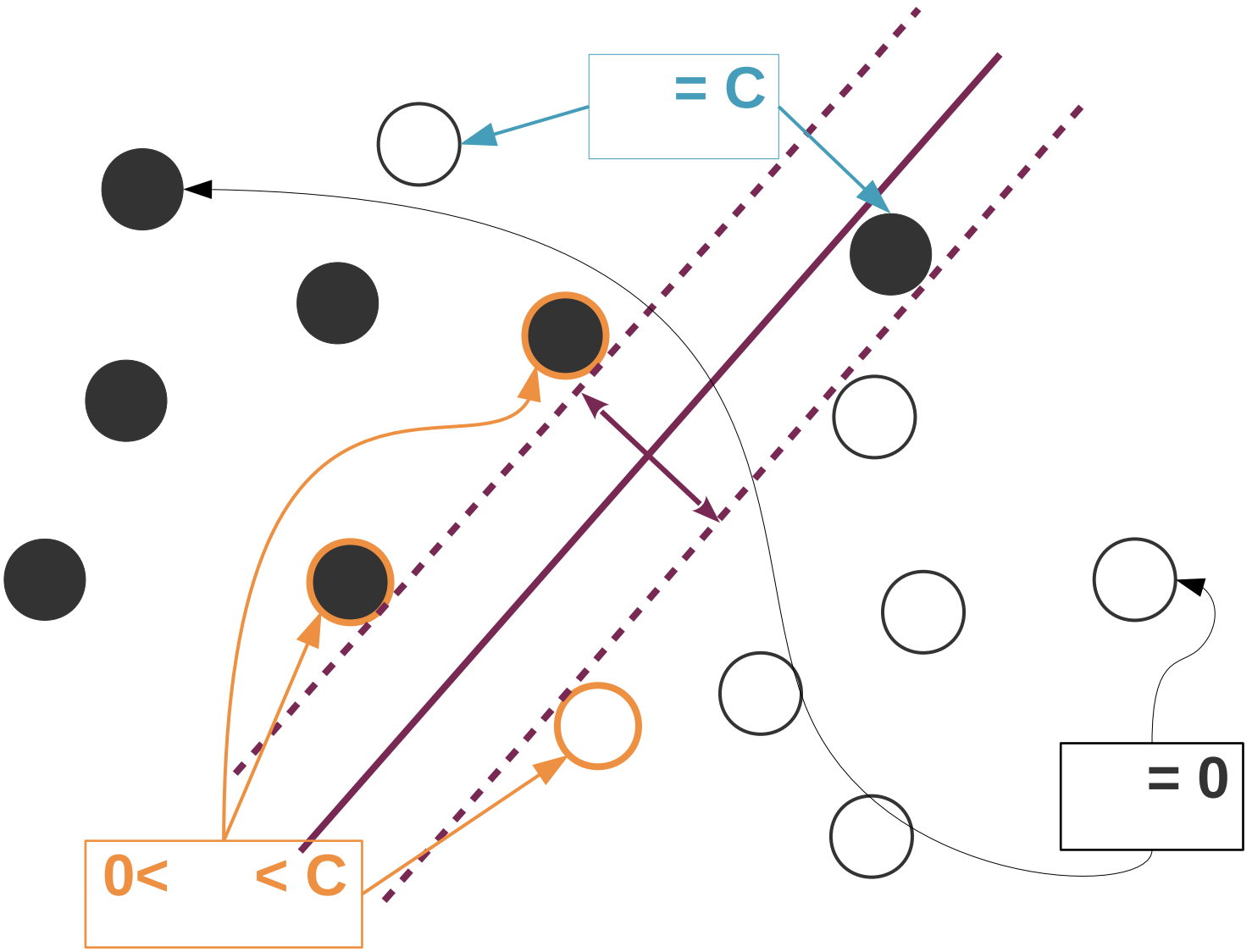
- Slater's condition (strong duality) primal and dual problems have the **same** optimum.

- Karush-Kuhn-Tucker Conditions give us a relation between dual and primal solution

- Geometric interpretation

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) > 1 && \text{“easy”} \\ \alpha_i = C &\Rightarrow y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) < 1 && \text{“hard”} \\ 0 < \alpha_i < C &\Rightarrow y^i(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b) = 1 && \text{“somewhat hard”} \end{aligned}$$

Support vectors of the soft-margin SVM



Primal vs. dual

- **What is the dimension of the primal problem?**

$$\arg\min_{\boldsymbol{w},b}\left(\sum_{i=1}^{n} l_{\text{hinge}}(\langle \boldsymbol{w}, x^i\rangle + b, y^i) + \lambda||\boldsymbol{w}||^2\right)$$

- **What is the dimension of the dual problem?**

$$\arg\max_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y^i y^j \langle \boldsymbol{x}^i, \boldsymbol{x}^j\rangle$$

$$0 \le \alpha_i \le C \text{ and } \sum_{i=1}^{n}\alpha_i y^i = 0$$

Primal vs. dual

- Primal: (**w**, b) has **dimension (p+1).**

$$\arg\min_{\boldsymbol{w},b}\left(\sum_{i=1}^{n} l_{\text{hinge}}(\langle \boldsymbol{w}, x^i\rangle + b, y^i) + \lambda||\boldsymbol{w}||^2\right)$$

Favored if the data is **low-dimensional.**

- Dual: has **dimension n.**

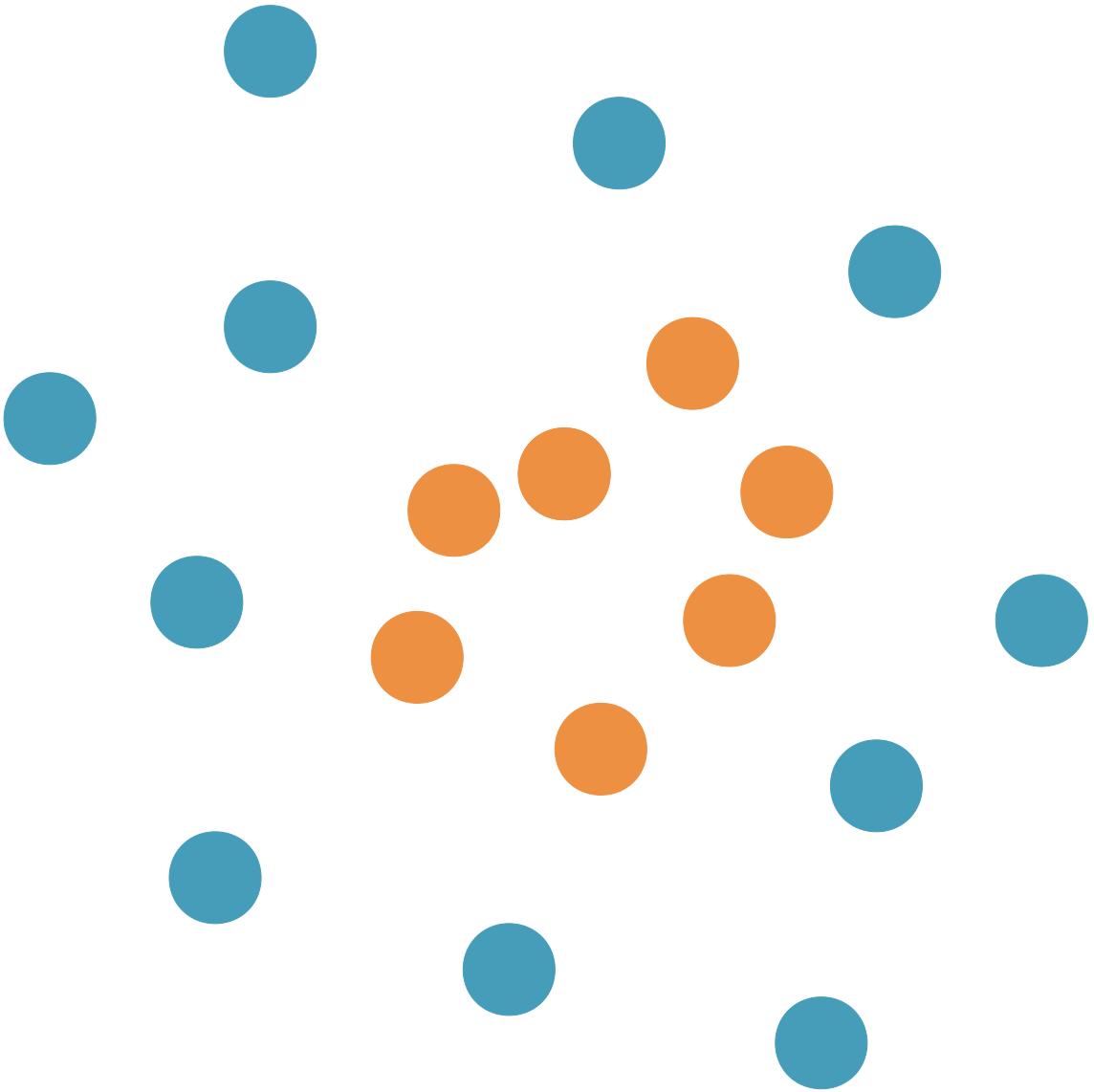
$$\arg\max_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y^i y^j \langle \boldsymbol{x}^i, \boldsymbol{x}^j\rangle$$

$$0 \le \alpha_i \le C \text{ and } \sum_{i=1}^{n}\alpha_i y^i = 0$$

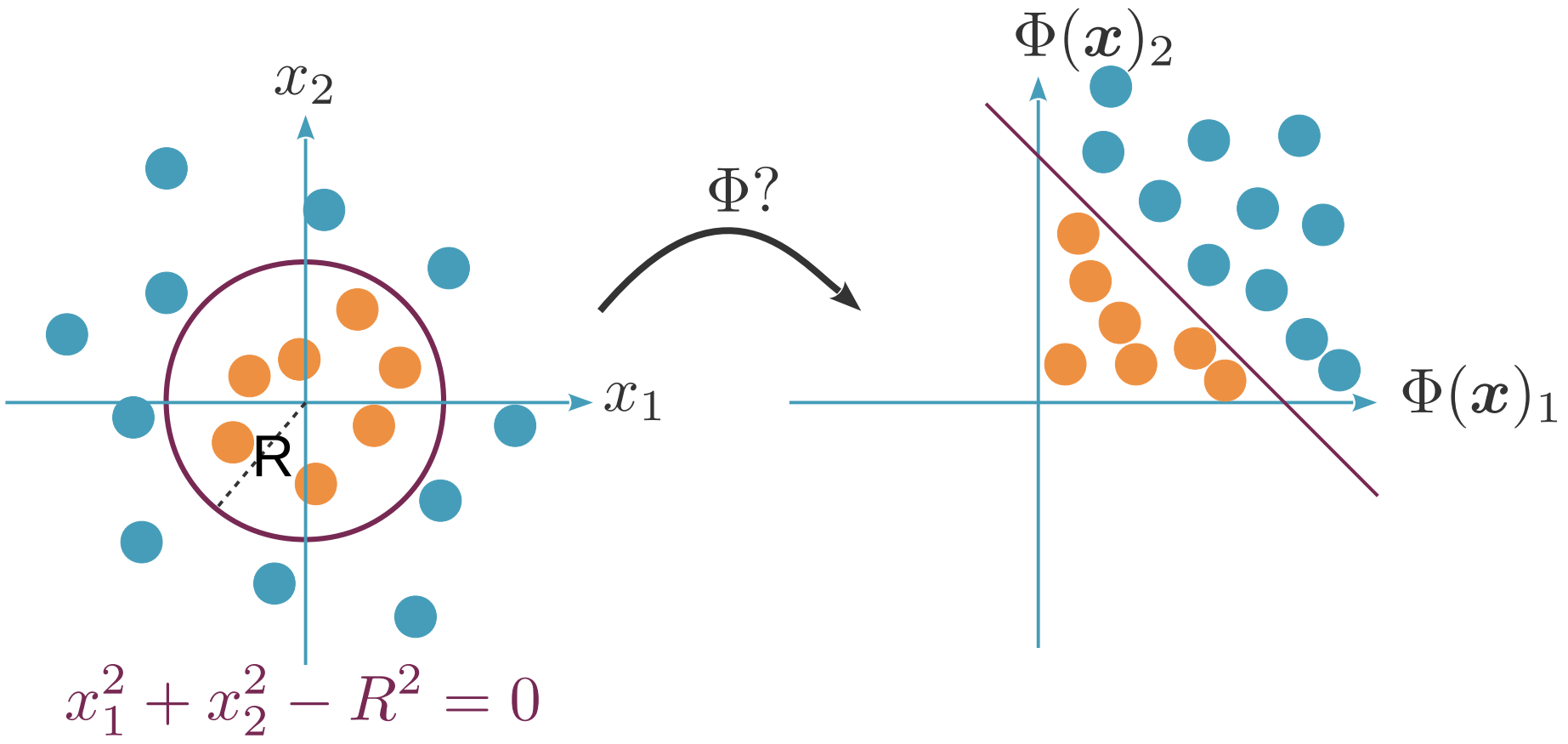
Favored is there is **little data** available.

Kernel methods

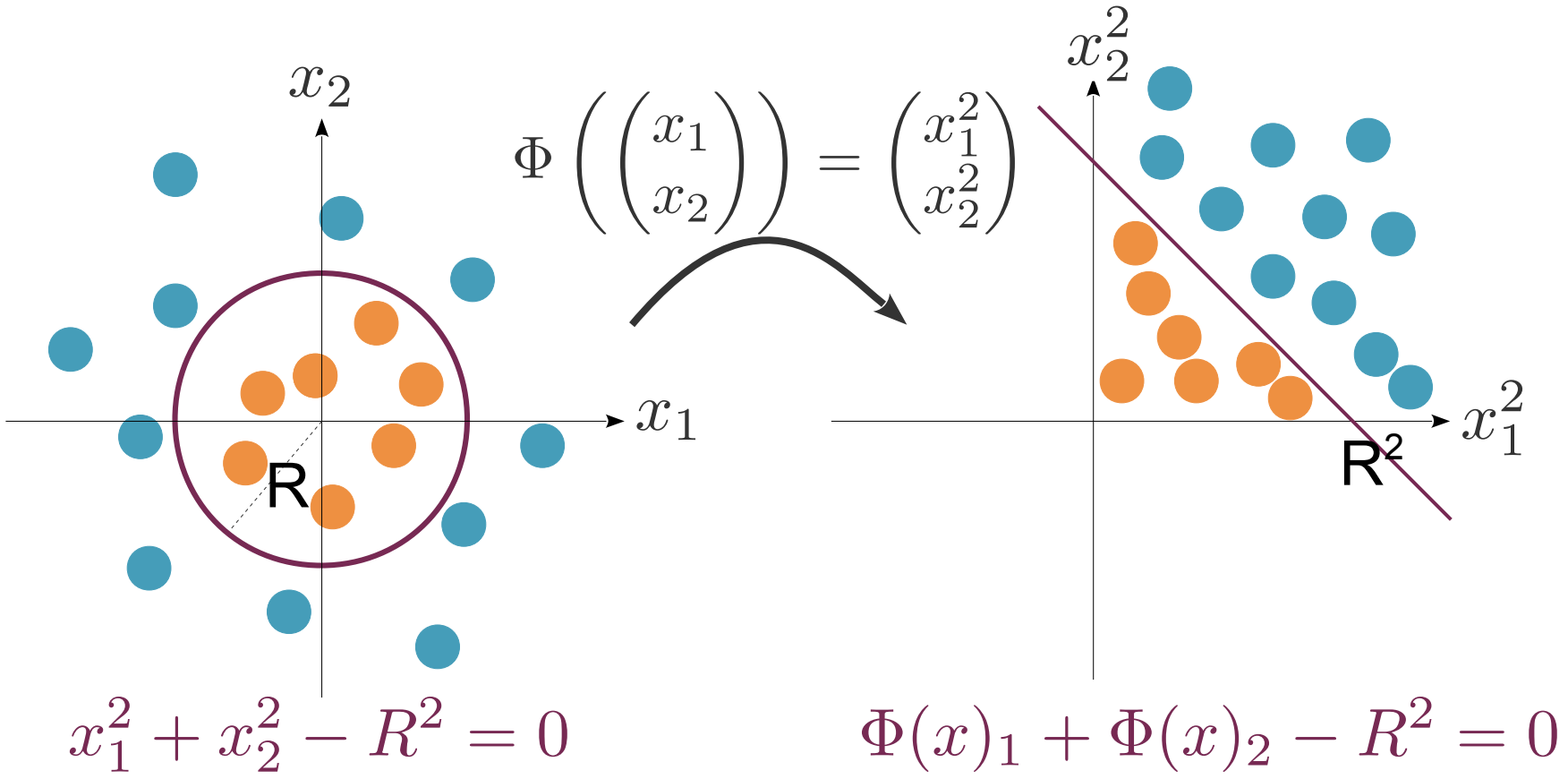
Non-linear SVMs



Non-linear mapping to a feature space



Non-linear mapping to a feature space



SVM in the feature space

- **Train:**

$$\arg\max_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^i y^j \langle \Phi(\boldsymbol{x}^i), \Phi(\boldsymbol{x}^j) \rangle_{\mathcal{H}}$$

under the constraints

$$0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{n} \alpha_i y^i = 0$$

- **Predict** with the decision function

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i^* y^i \langle \Phi(\boldsymbol{x}^i), \Phi(\boldsymbol{x}) \rangle_{\mathcal{H}} + b^*$$

Kernels

For a given mapping

$$\Phi : \mathcal{X} \mapsto \mathcal{H}$$

from the space of objects X to some Hilbert space H, the **kernel** between two objects x and x' is the inner product of their images in the feature spaces.

$$\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}, K(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$$

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$$

e.g.

$$K(\boldsymbol{x}, \boldsymbol{x}') = \left\langle \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix}, \begin{pmatrix} {x'_1}^2 \\ {x'_2}^2 \end{pmatrix} \right\rangle = x_1^2 {x'_1}^2 + x_2^2 {x'_2}^2$$

SVM with a kernel

- **Train:**

$$\arg\max_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^i y^j K(\boldsymbol{x}^i, \boldsymbol{x}^j)$$

under the constraints

$$0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{n} \alpha_i y^i = 0$$

- **Predict** with the decision function

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i^* y^i K(\boldsymbol{x}^i, \boldsymbol{x}) + b^*$$

Kernel trick

- Many linear algorithms (in particular, linear SVMs) can be performed in the feature space H **without explicitly computing the images (x)**, but instead by computing kernels **K(x, x'):**
  - SVMs, but also
  - Ridge regression (but not the Lasso)
  - Dimensionality reduction: PCA
  - Clustering: k-means
- It is sometimes easy to compute kernels which correspond to large-dimensional feature spaces: **K(x, x') is often much simpler to compute than (x).**

Exemple 1: Polynomial kernels

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2 \qquad \Phi(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix} \in \mathbb{R}^3$$

$$\begin{aligned} K(\boldsymbol{x}, \boldsymbol{x}') &= x_1^2 {x'_1}^2 + 2x_1x_2x'_1x'_2 + x_2^2 {x'_2}^2 \\ &= \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^2 \end{aligned}$$

More generally, for $\mathcal{X} = \mathbb{R}^p$

$$K(x, x') = (\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1)^d$$

is an inner product in a feature space of $\binom{d}{p+d}$ monomials of degree up to d.

K is much easier to compute than .

Which functions are kernels?

- A function K(**x**, **x'**) defined on a set X is a **kernel** iff it exists a Hilbert space H and a mapping : X H such that, for any **x**, **x'** in X:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$$

- A function K(**x**, **x'**) defined on a set X is **positive definite** iff it is **symmetric** and satisfies:

$$\forall N \in \mathbb{N}, \forall (\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N) \in \mathcal{X}^N \text{ and } (a_1, a_2, \ldots, a_N) \in \mathbb{R}^N$$
$$\sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j K(\boldsymbol{x}^i, \boldsymbol{x}^j) \geq 0.$$

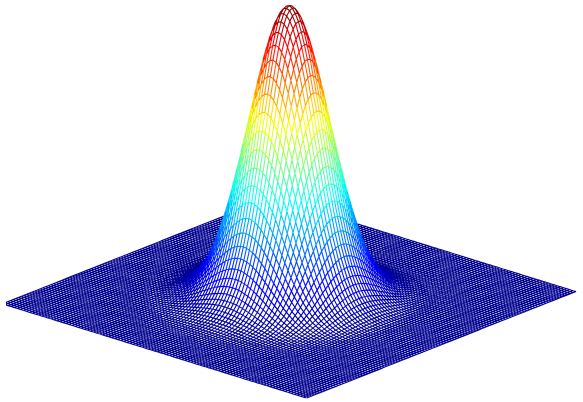
- Theorem [Aronszajn, 1950]: **K is a kernel iff it is positive definite.**
- Matrix K(**x**,x**'**) for any **x**, **x'** in X is called Gram matrix

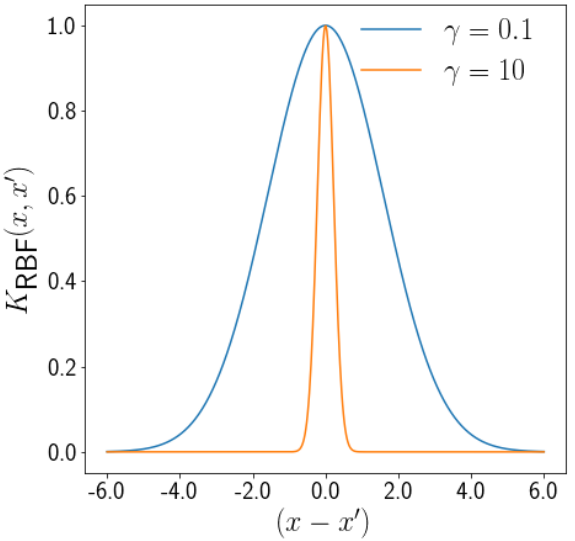
Exemple 1: Gaussian RBF kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\sigma^2}\right)$$

- Corresponds to a feature space of **inf nite dimension** (containing all monomials)

$$\begin{aligned} K(x, x') &= \exp\left(\tfrac{-1}{2\sigma^2}||x||^2\right) \exp\left(\tfrac{1}{\sigma^2}\langle x, x'\rangle\right) \exp\left(\tfrac{-1}{2\sigma^2}||x'||^2\right) \\ &= f(x) \sum_{r=0}^{+\infty} \frac{\langle x, x'\rangle^r}{\sigma^{2r} r!} f(x') \end{aligned}$$





Exemple 2: String kernels

- Exemple of application: protein classification

  Goal: predict which proteins are secreted or not, based on their sequence.

- Secreted proteins:

  MASKATLLLAFTLLFATCIARHQQRQQQQNQCQLQNIEA...
  MARSSLFTFLCLAVFINGCLSQIEQQSPWEFQGSEVW...
  MALHTVLIMLSLLPMLEAQNPEHANITIGEPITNETLGWL...
  ...

- Non-secreted proteins:

  MAPPSVFAEVPQAQPVLVFKLIADFREDPDPRKVNLGVG...
  MAHTLGLTQPNSTEPHKISFTAKEIDVIEWKGDILVVG...
  MSISESYAKEIKTAFRQFTDFPIEGEQFEDFLPIIGNP..
  ...

Substring-based representations

- Represent strings based on the presence/absence of substrings of fixed length.

$$\Phi(\boldsymbol{x}) = \{\Phi_u(\boldsymbol{x})\}_{u \in \mathcal{A}^k}$$

  - Number of occurrences of u in **x**: **spectrum kernel** [Leslie et al., 2002].

Spectrum kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{u \in \mathcal{A}^k} \Phi_u(\boldsymbol{x}) \Phi_u(\boldsymbol{x}')$$

- **Implementation:**
  - Formally, a sum over $|A^k|$terms
  - At most **|x|** - k + 1 non-zero terms $\Phi(\boldsymbol{x})$
  - Hence: Computation in O(**|x|**+**|x'|**)
- **Fast prediction** for a new sequence **x:**

$$\begin{aligned} f(\boldsymbol{x}) &= \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b \\ &= \sum_{u \in \mathcal{A}^k} w_u \Phi_u(\boldsymbol{x}) + b \\ &= \sum_{j=1}^{|\boldsymbol{x}|-k+1} w_{x_j x_{j+1} \ldots x_{j+k-1}} + b \end{aligned}$$

Spectrum kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{u \in \mathcal{A}^k} \Phi_u(\boldsymbol{x}) \Phi_u(\boldsymbol{x}')$$

- **Implementation:**
  - Formally, a sum over $|A^k|$ terms
  - At most $|\mathbf{x}|$ - k + 1 non-zero terms $\Phi(\boldsymbol{x})$
  - Hence: Computation in $O(|\mathbf{x}|+|\mathbf{x}'|)$

for proteins: alphabet of 22 amino acids
considering 5-mers: 21^5 > 4 million

- **Fast prediction** for a new sequence **x:**

$$\begin{aligned} f(\boldsymbol{x}) &= \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b \\ &= \sum_{u \in \mathcal{A}^k} w_u \Phi_u(\boldsymbol{x}) + b \\ &= \sum_{j=1}^{|\boldsymbol{x}|-k+1} w_{x_j x_{j+1} \ldots x_{j+k-1}} + b \end{aligned}$$