

Advanced Learning Models

Jakob Verbeek

jakob.verbeek@inria.fr

December 10, 2015

<http://lear.inrialpes.fr/people/mairal/teaching/2015-2016/MSIAM/>

Course content

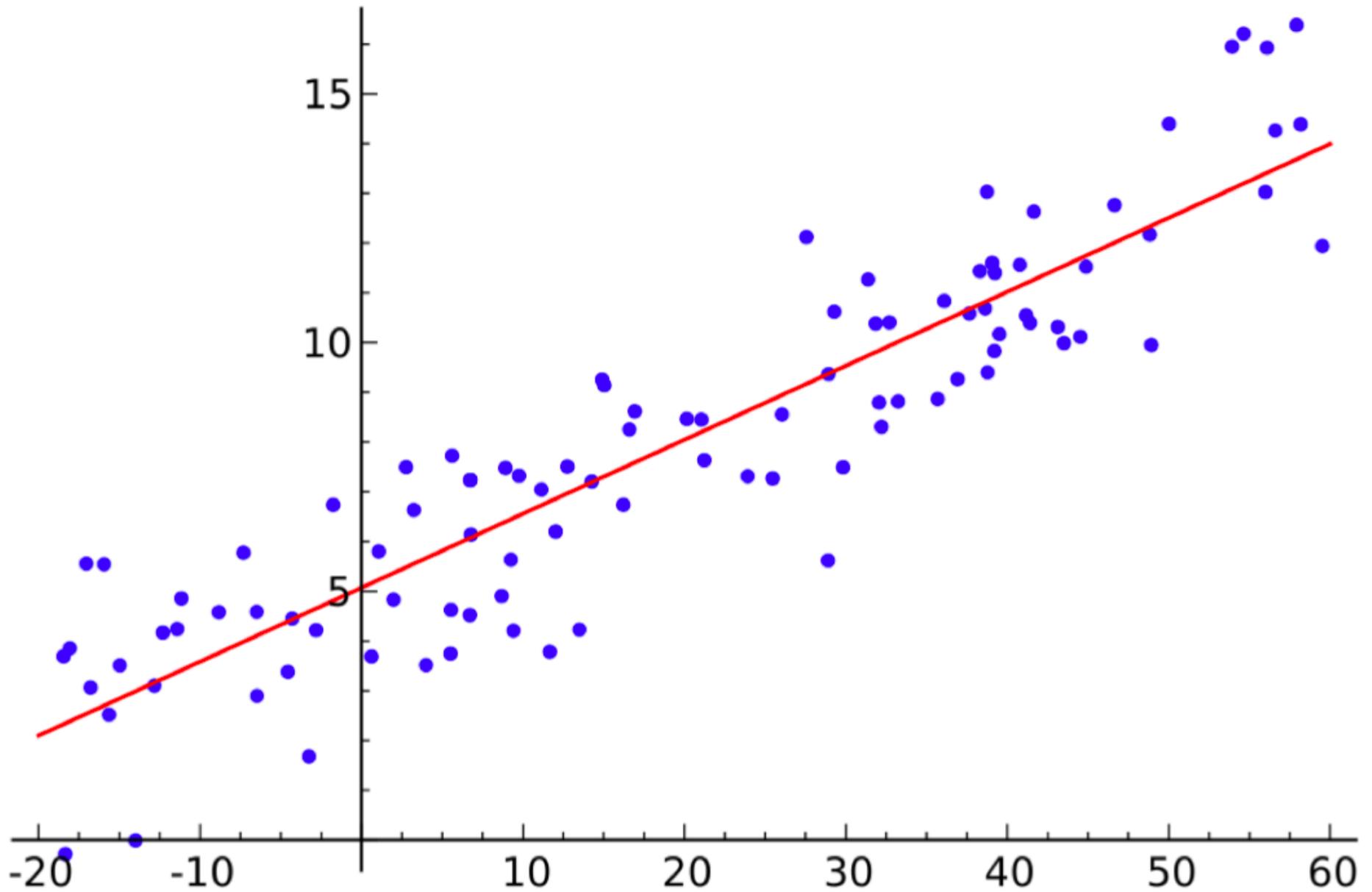
- Lecture 1
 - ▶ Introduction
 - ▶ Linear classification
 - ▶ Non-linear classification with kernels
 - ▶ Kernel-trick more generally
 - ▶ Bias-variance decomposition

- Lectures 2,3,4 (Julien Mairal)
 - ▶ Theory on kernels

- Lectures 5,6 (Jakob Verbeek)
 - ▶ Fisher kernel
 - ▶ Convolutional and recurrent neural networks

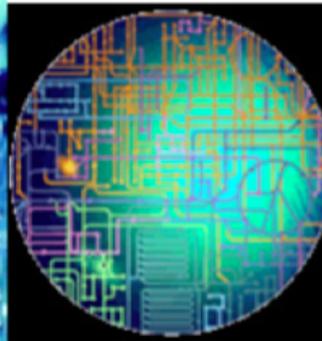
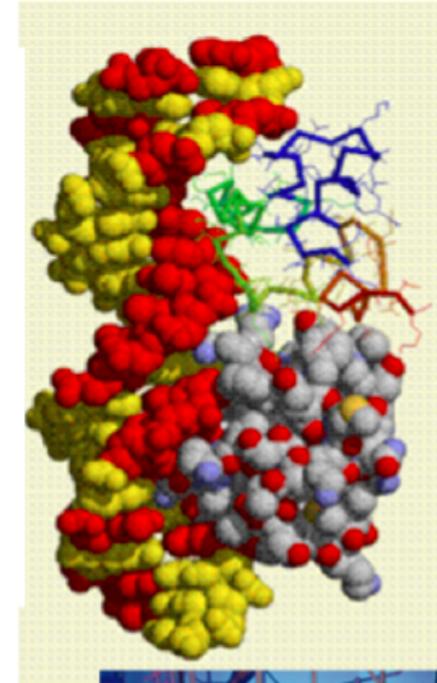
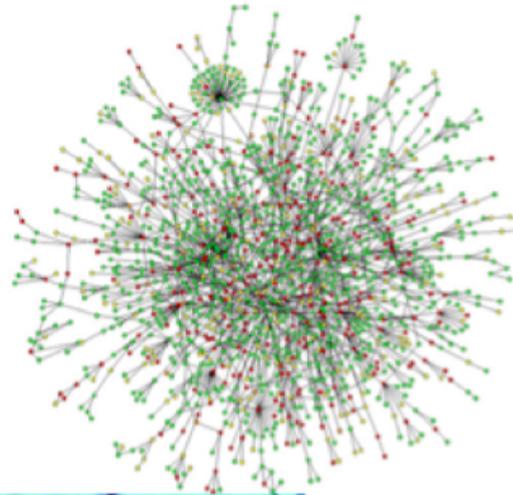
Course content

- From classic linear learning problems



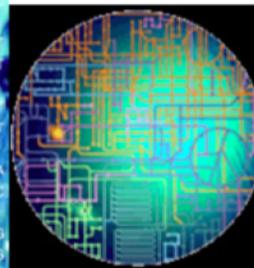
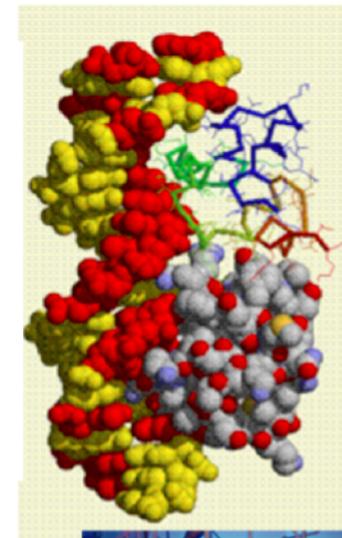
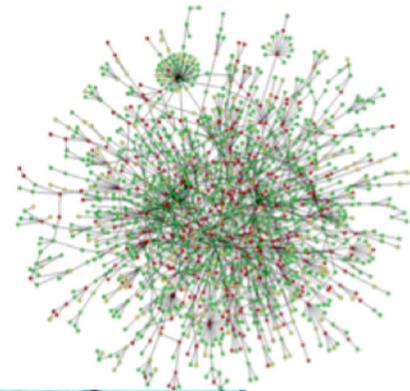
Course content

- To current practical learning problems



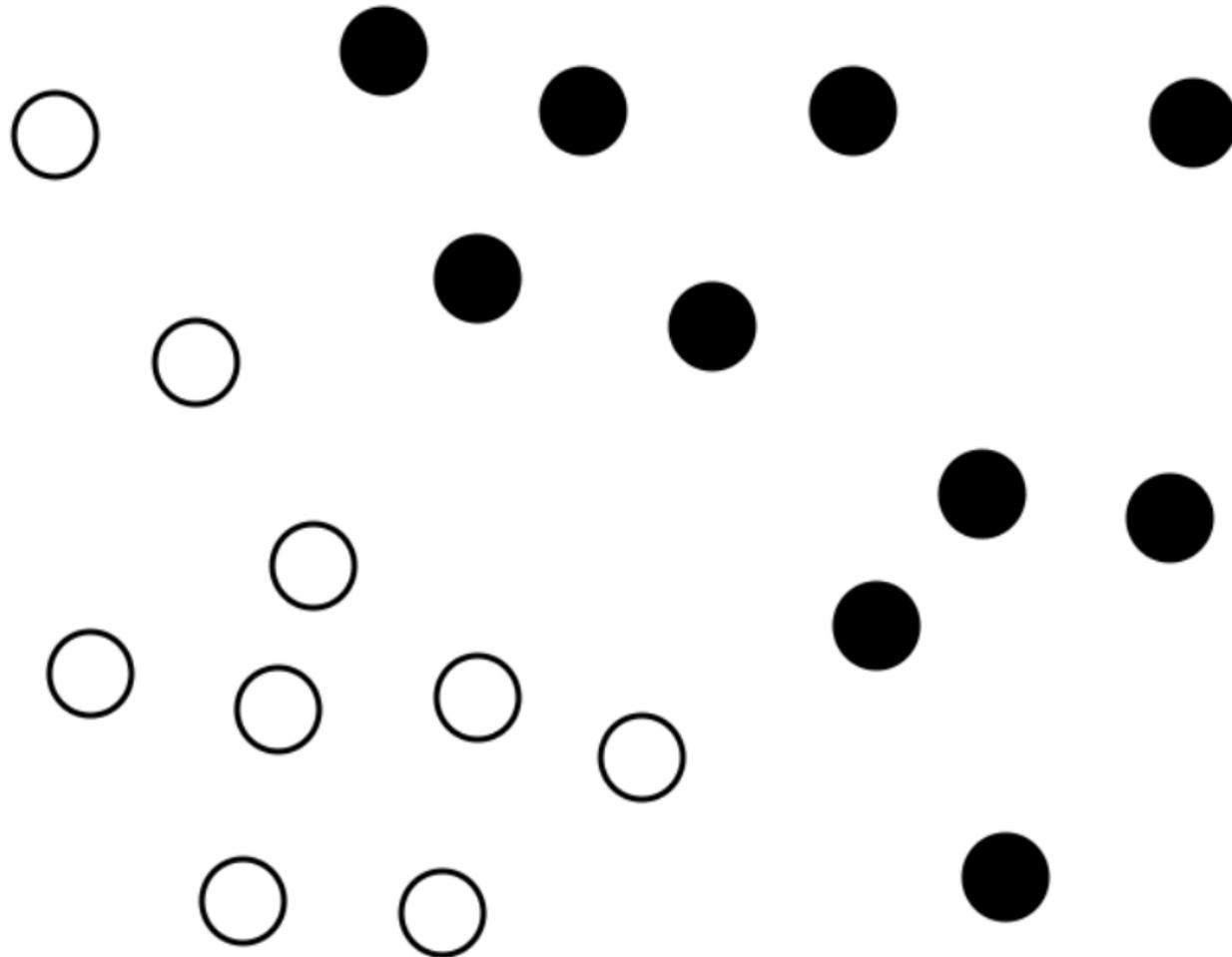
Course content

- Extend well understood linear statistical learning techniques to real-world complicated, structured and high-dimensional data (images, text, time series, graphs, distributions, permutations, ...)
- Kernels: basic theory and kernel design
- Neural networks: learning convolutional and recurrent architectures



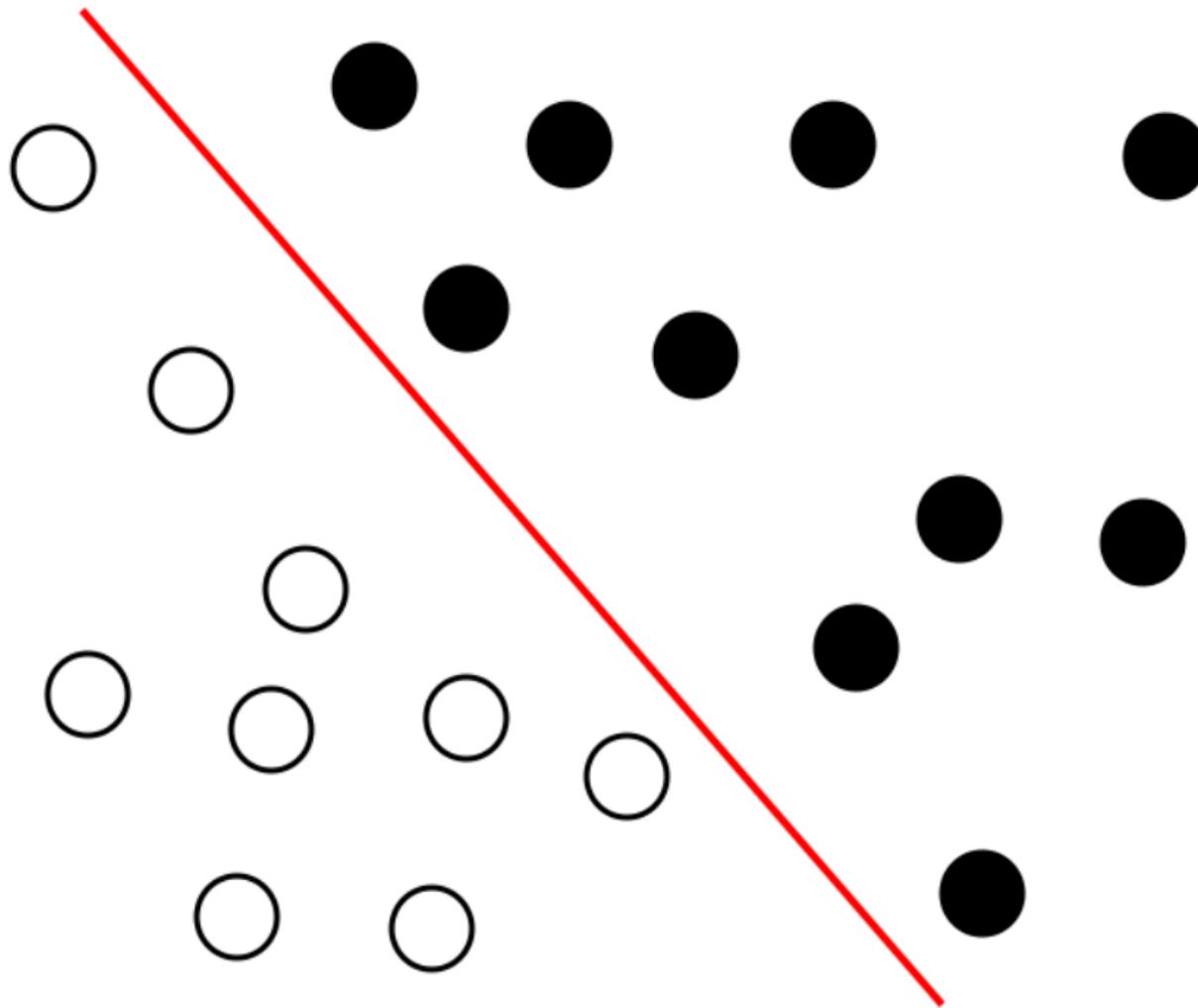
Learning predictive models from data

- Given training data labeled for two or more classes



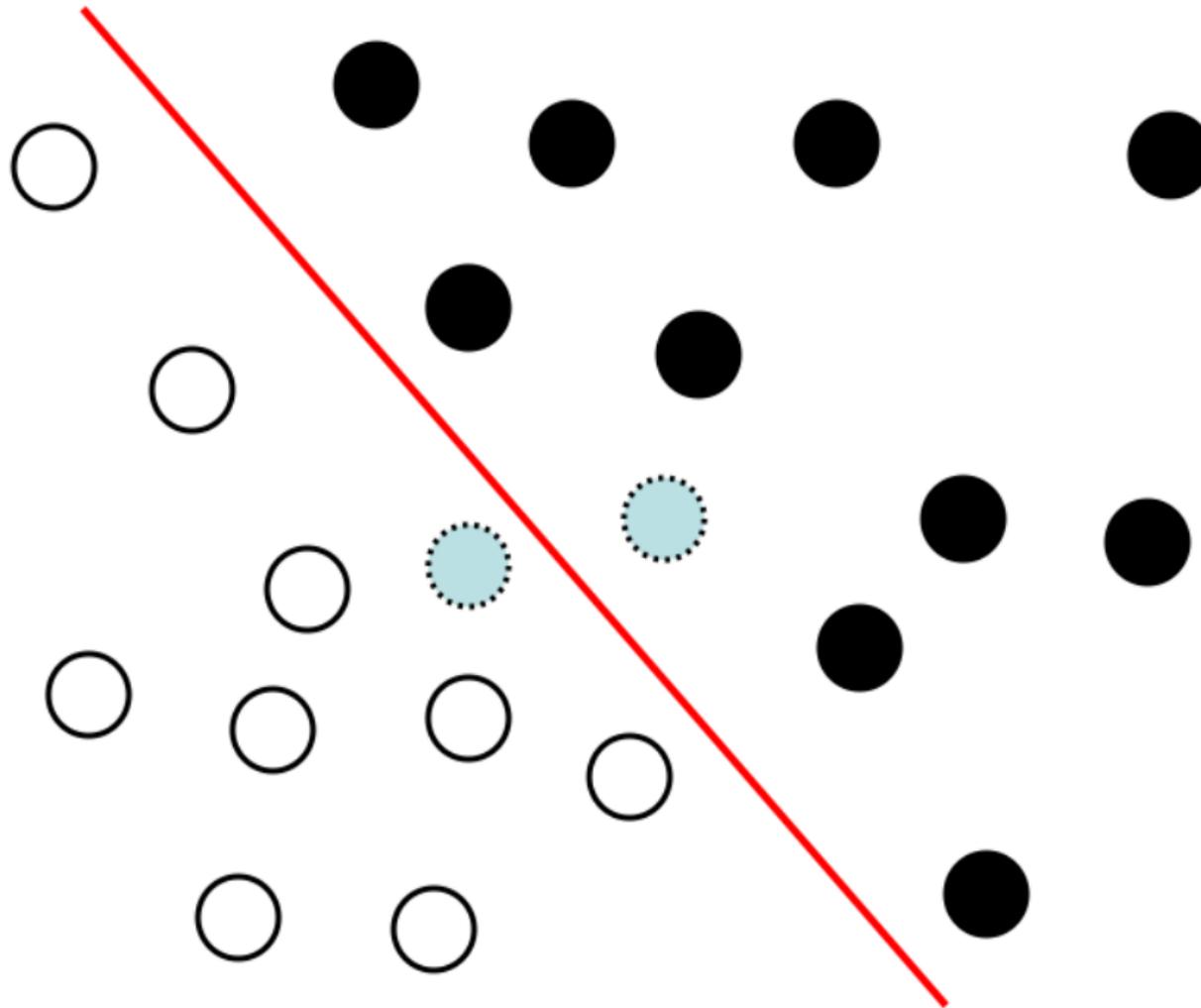
Learning predictive models from data

- Given training data labeled for two or more classes
- Determine a decision surface that separates those classes



Learning predictive models from data

- Given training data labeled for two or more classes
- Determine a decision surface that separates those classes
- Use that surface to predict the class membership of new data



Recommender systems

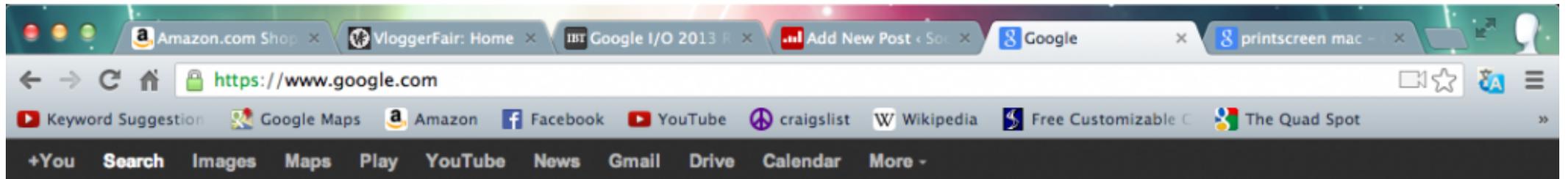
- Given a dataset of users and the movies they liked
- Predict which other movies a given user would also like

The image shows a screenshot of the Netflix website interface. At the top, there is a red navigation bar with a 'Browse' menu icon on the left, the 'NETFLIX' logo in the center, and a search bar on the right. Below the navigation bar, the main content is organized into several sections:

- Continue Watching:** This section features four large video thumbnails. Each thumbnail has a red play button icon in the center. The titles are: 'Mad Men: Season 3: Ep. 2 - Love Among ...', 'Pitch Black', 'Breaking Bad: Season 2: Ep. 3 - Bit by a ...', and 'Sleeper Cell: Season 1: Ep. 9'. Each thumbnail also includes a small information icon (i) in the bottom right corner.
- Top Picks for You:** This section displays a grid of movie and TV show covers. From left to right, the visible titles are: 'Being John Malkovich', 'Avatar: The Last Airbender', 'Damages', 'Drop Dead Diva', 'Jericho', 'Waking the Dead', and 'Heroes'.
- Instant Queue:** This section shows a row of content thumbnails. From left to right, the visible titles are: 'The Sandlot', 'Saturday Night Live: The Best of ...', 'Quantum Leap', 'The Mentalist', 'The Vampire Diaries', and 'Dora the Explorer'.

Recommender systems

- Given a dataset of queries and click-through data
- Predict which are the most relevant pages for a given query



SIGN IN

Google

Google Search

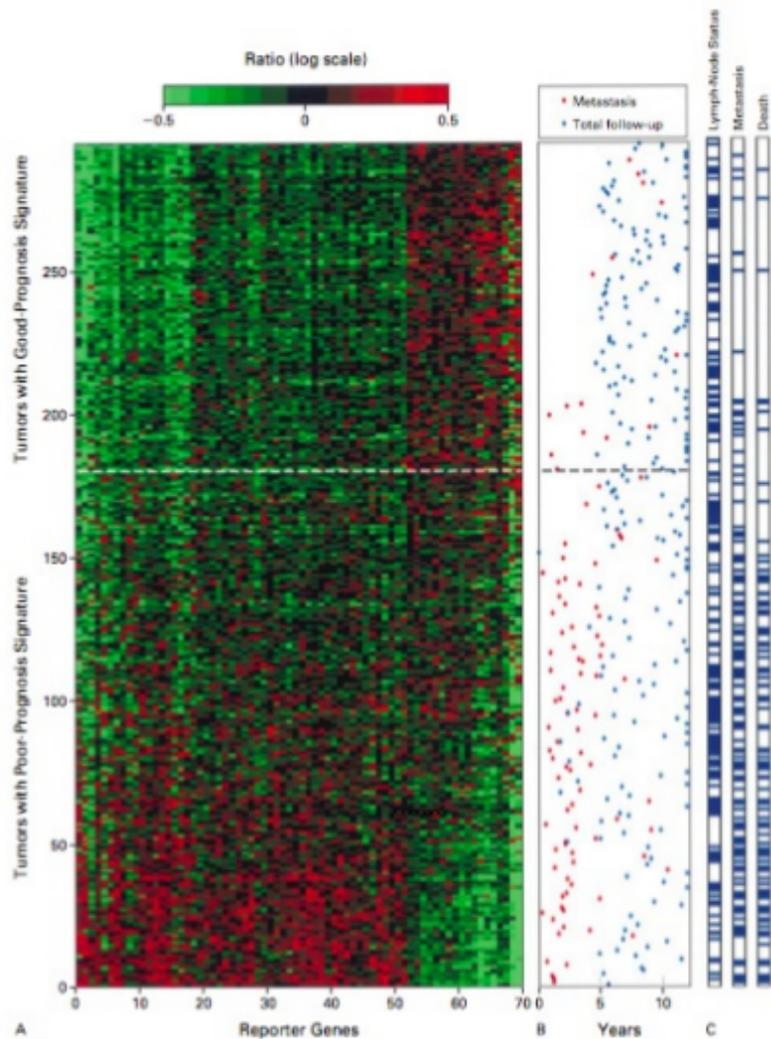
I'm Feeling Lucky

Natural Language Processing

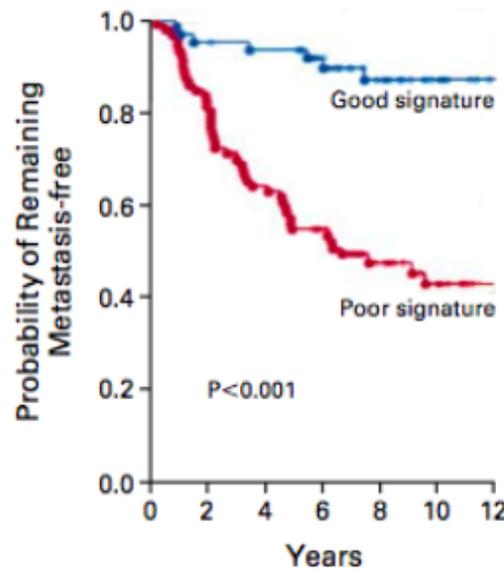
- Given a text, predict its topic
- Given an email, predict whether it is spam
- Given a text, predict its translation in another language
- Etc.

Tumor classification for prognosis

- Given the expression of genes in a new tumor, predict the development over the next 5 years

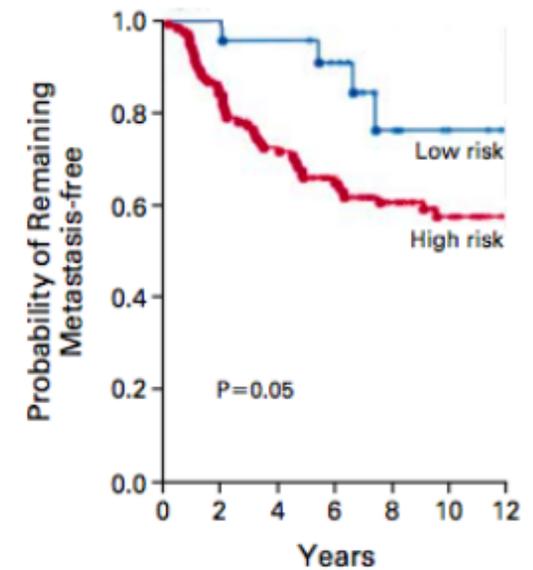


A Gene-Expression Profiling



	0	2	4	6	8	10	12
Good signature	60	57	54	45	31	22	12
Poor signature	91	72	55	41	26	17	9

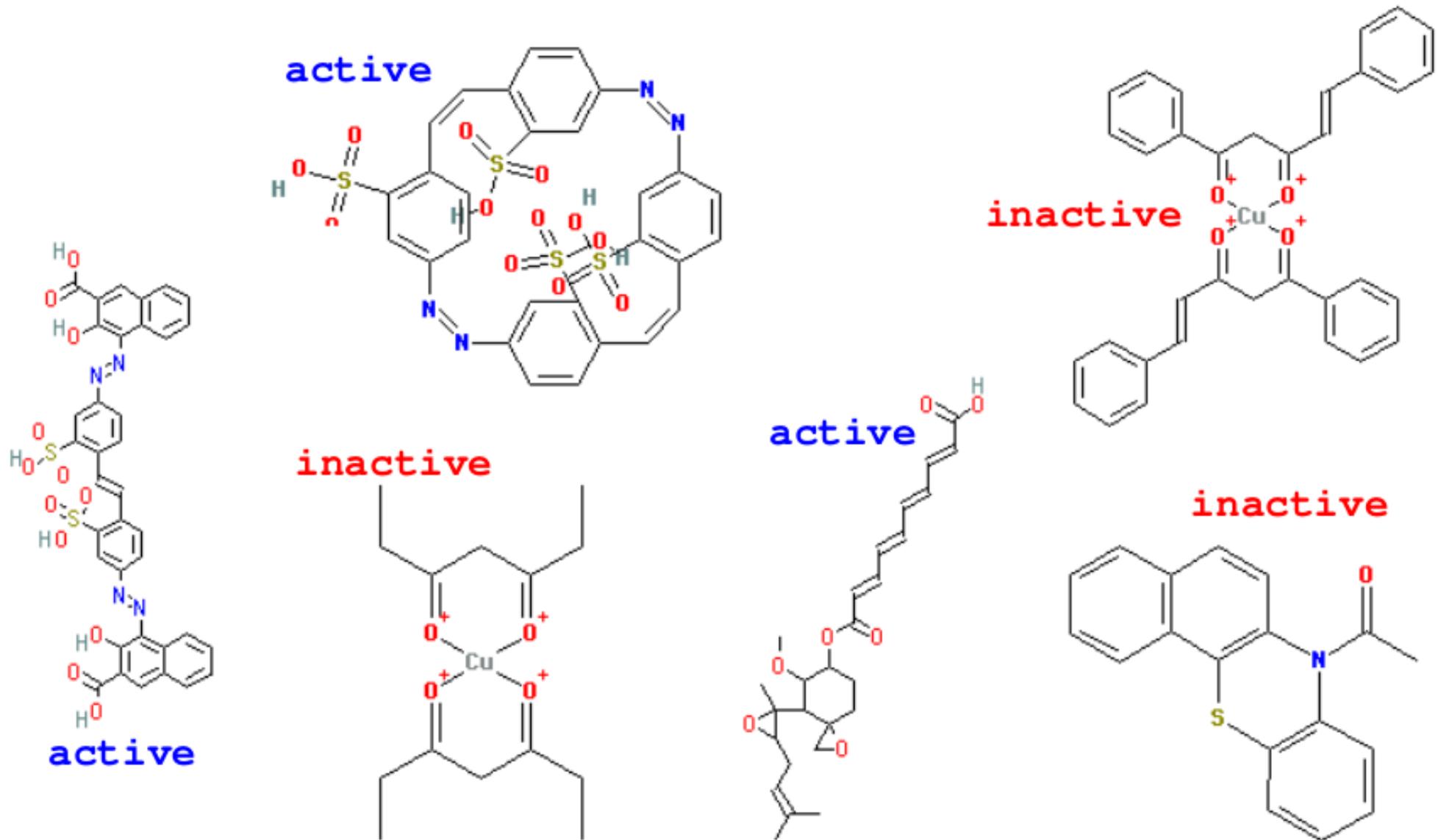
B St. Gallen Criteria



	0	2	4	6	8	10	12
Low risk	22	22	21	17	9	5	2
High risk	129	107	88	69	48	34	19

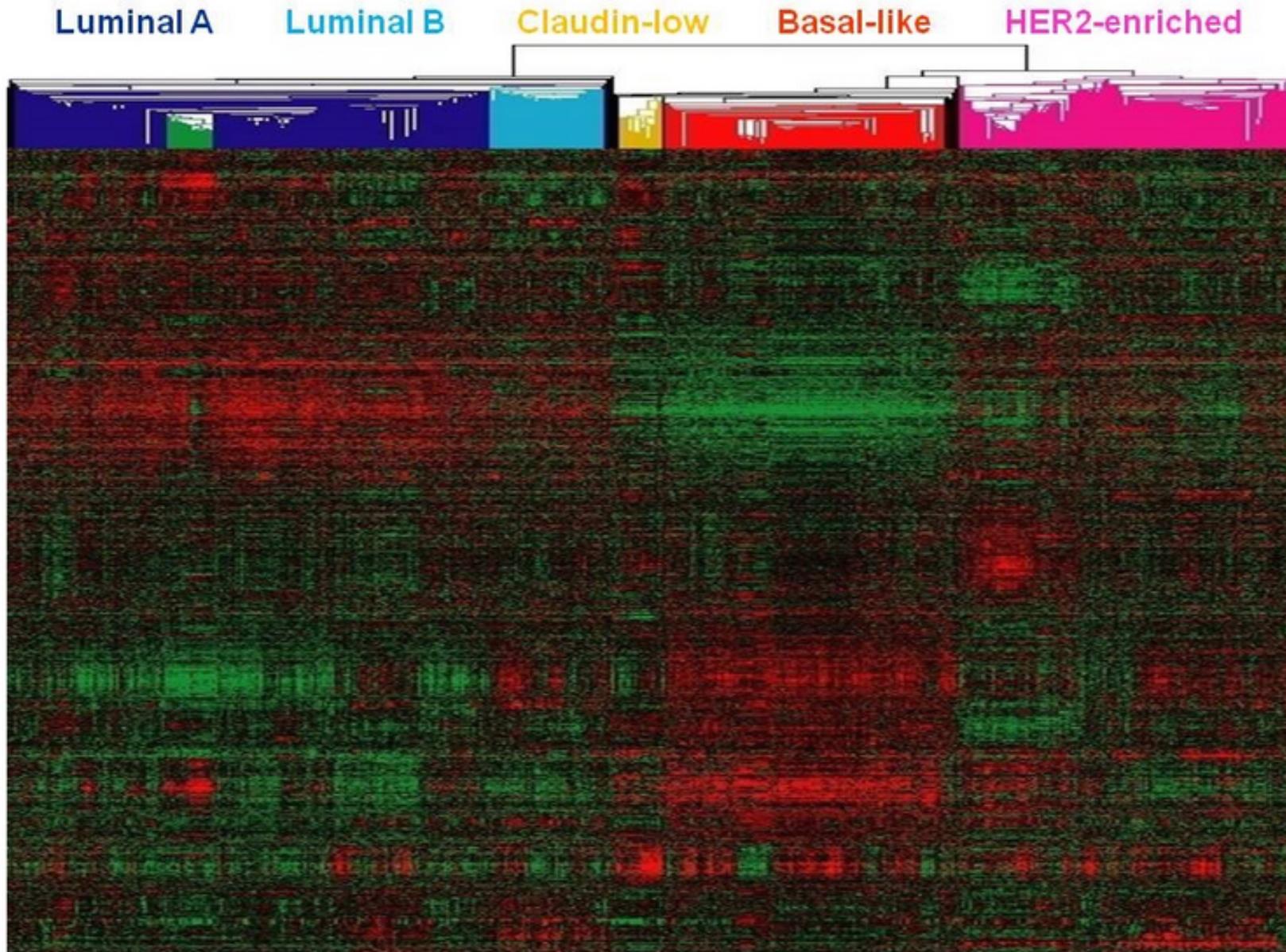
Molecule classification for drug design

- Given a candidate molecule, predict whether it is active against a certain condition



Gene expression clustering

- Are there groups of breast tumors with similar gene expression profile?



Audio understanding

- Given an audio stream, predict which song is played

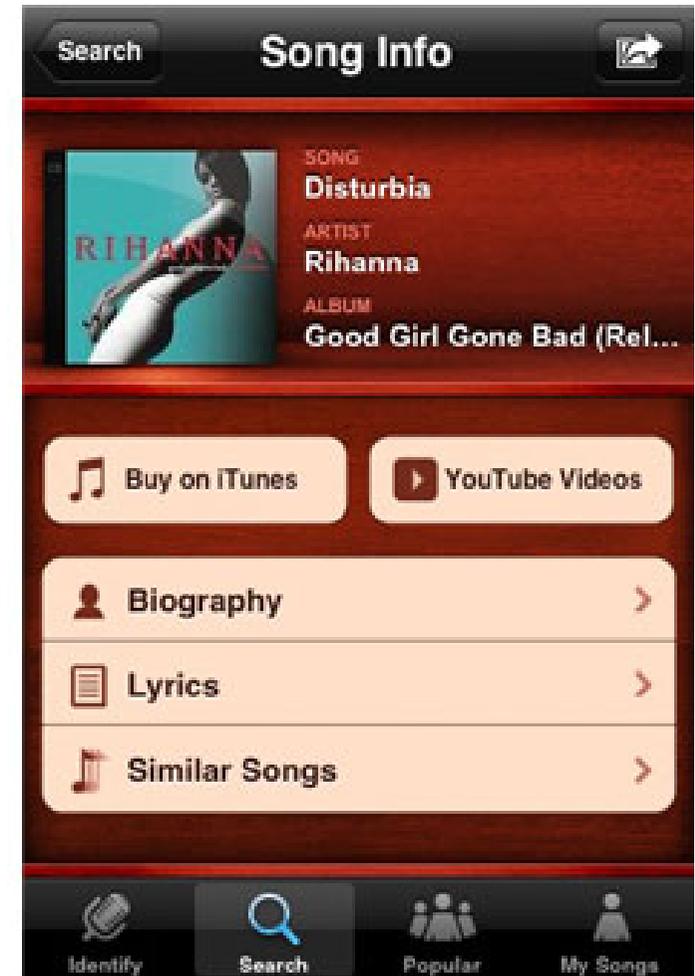
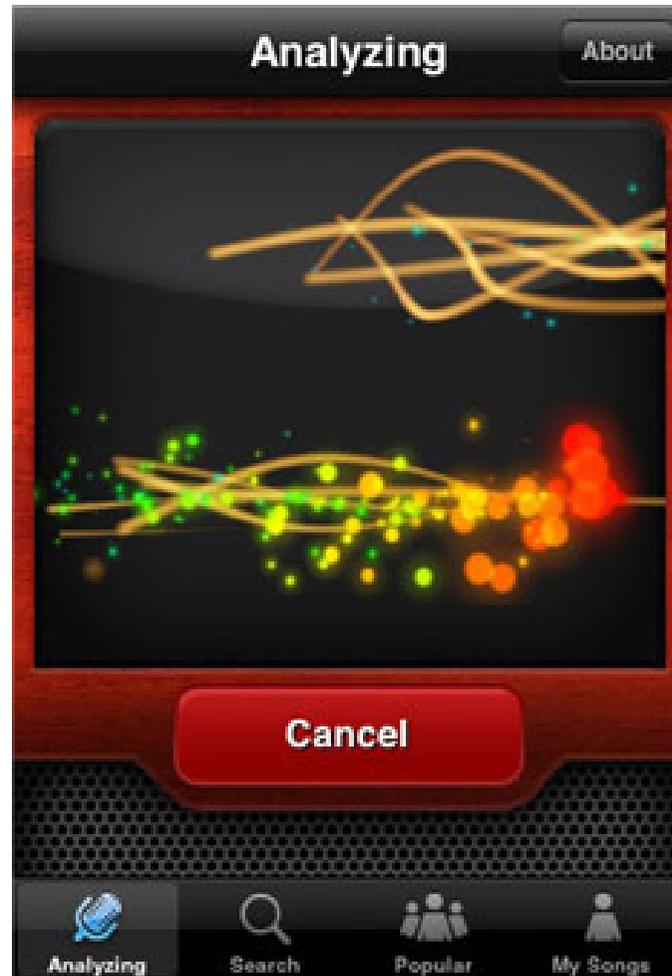


Image Inpainting

- Complete an image with missing parts
- predict each image patch, as a linear combination of dictionary elements



Since 1699, when French explorers landed at the great bend of the Mississippi River and celebrated the first Mardi Gras in North America, New Orleans has brewed a fascinating melange of cultures. It was French, then Spanish, then French again, then sold to the United States. Through all these years, and even into the 1900s, others arrived from everywhere: Acadians (Cajuns), Africans, indige-

Image Inpainting

- Complete an image with missing parts
- predict each image patch, as a linear combination of dictionary elements



Image Inpainting

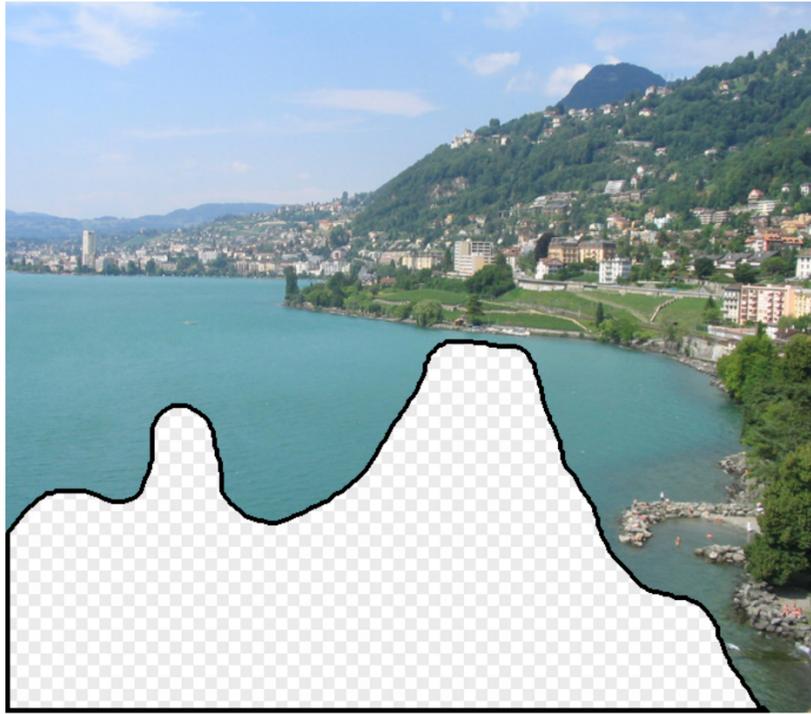


Image super resolution

- Given an image, predict a high-resolution version of it
- Predictions per-patch, ensure spatial consistency



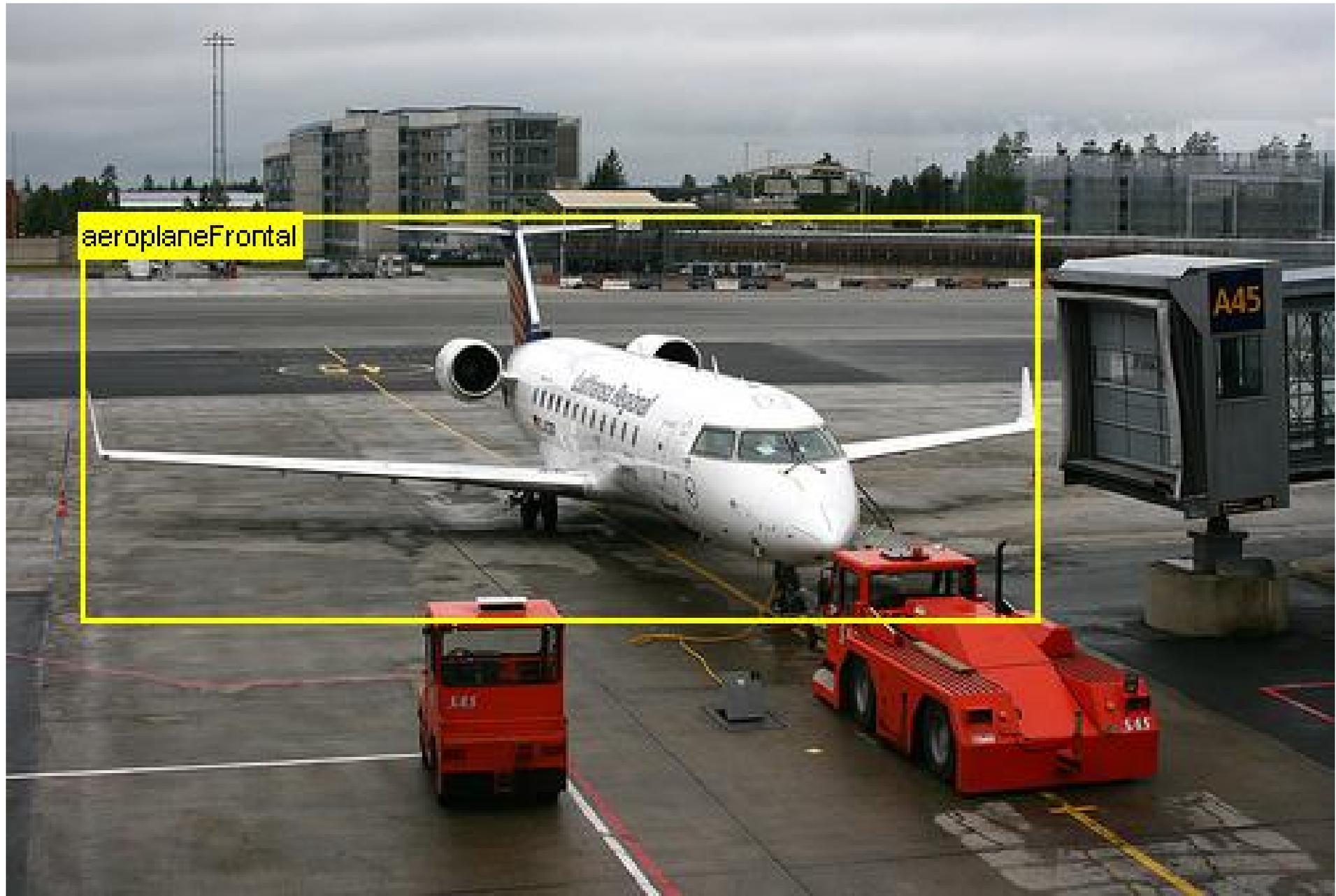
Classification examples in category-level recognition

- Given an image, predict if labels are relevant or not
- For example: Person = yes, TV = yes, car = no, ...



Classification examples in category-level recognition

- Category localization: predict bounding box coordinates for each object



Classification examples in category-level recognition

- Semantic segmentation: classify pixels to categories (multi-class)
- Impose spatial smoothness by Markov random field models.



Video understanding

- Given a video: predict the type of event that is shown: birthday party



Video understanding

- Given a video: predict spatio-temporal location of an action, eg drinking



Image captioning

- Given an image: predict a natural language description



a brown dog is running through the grass

Advanced learning models

- Each of these examples involves complex objects/large numbers of features for a restricted number of samples
- Intuitively, observing all these characteristics should allow us to predict or understand complex mechanisms
- But it also means that we should use very **rich model classes** that can capture a wealth of complex dependencies
- Introduces a **risk of overfitting**: modeling co-incidental structure in the data
- However, this wealth of features can cause trouble in statistical learning
- This course
 - ▶ Modeling complex data structures with **kernels** and **neural networks**
 - ▶ **Regularization** to avoid overfitting

Course content

- Introduction
- **Linear classification**
- Non-linear classification with kernels
- Kernel-trick more generally
- Bias-variance decomposition

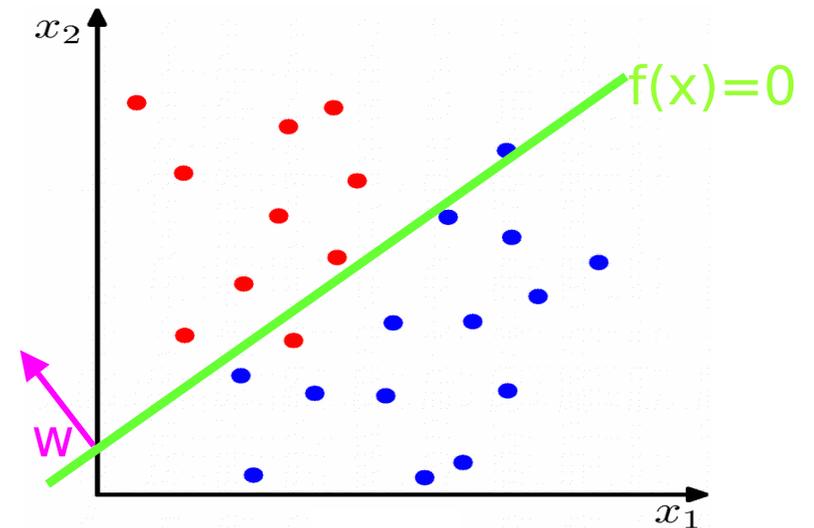
Binary linear classifier

- Decision function is linear in the features: $f(x) = w^T x + b$
- Classification based on the sign of $f(x)$

- Decision surface is $(d-1)$ dimensional hyper-plane orthogonal to w
- Offset from origin is determined by b

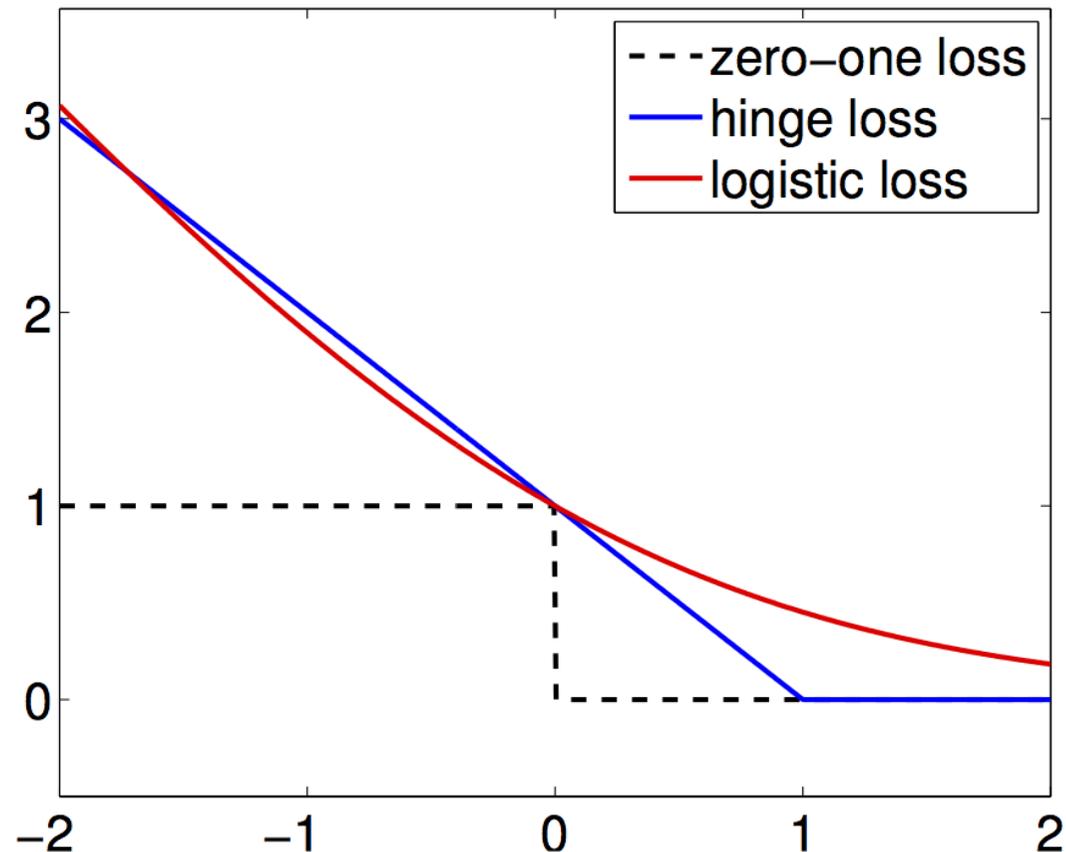
- We drop offset b , absorb it in x and w
 $x \leftarrow (x^T \ 1)^T$
 $w \leftarrow (w^T \ b)^T$

- We will now consider the two most commonly used linear classifiers
 - ▶ Logistic discriminant
 - ▶ Support vector machines



Common loss functions for classification

- Assign class label using $y = \text{sign}(f(x))$
 - ▶ Zero-One loss: $L(y_i, f(x_i)) = [y_i f(x_i) \leq 0]$
 - ▶ Hinge loss: $L(y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))$
 - ▶ Logistic loss: $L(y_i, f(x_i)) = \log_2(1 + e^{-y_i f(x_i)})$



Common loss functions for classification

- Assign class label using $y = \text{sign}(f(x))$
 - ▶ Zero-One loss: $L(y_i, f(x_i)) = [y_i f(x_i) \leq 0]$
 - ▶ Hinge loss: $L(y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))$
 - ▶ Logistic loss: $L(y_i, f(x_i)) = \log_2(1 + e^{-y_i f(x_i)})$
- The zero-one loss counts the number of misclassifications, which is the “ideal” empirical loss.
 - ▶ Discontinuity at zero makes optimization intractable.
- Hinge and logistic loss provide continuous and convex upperbounds
- Combined with convex penalties to prevent overfitting this leads to convex objective functions, for which global optima can be found.

Logistic discriminant classifier

- Map linear score function to class probabilities with sigmoid

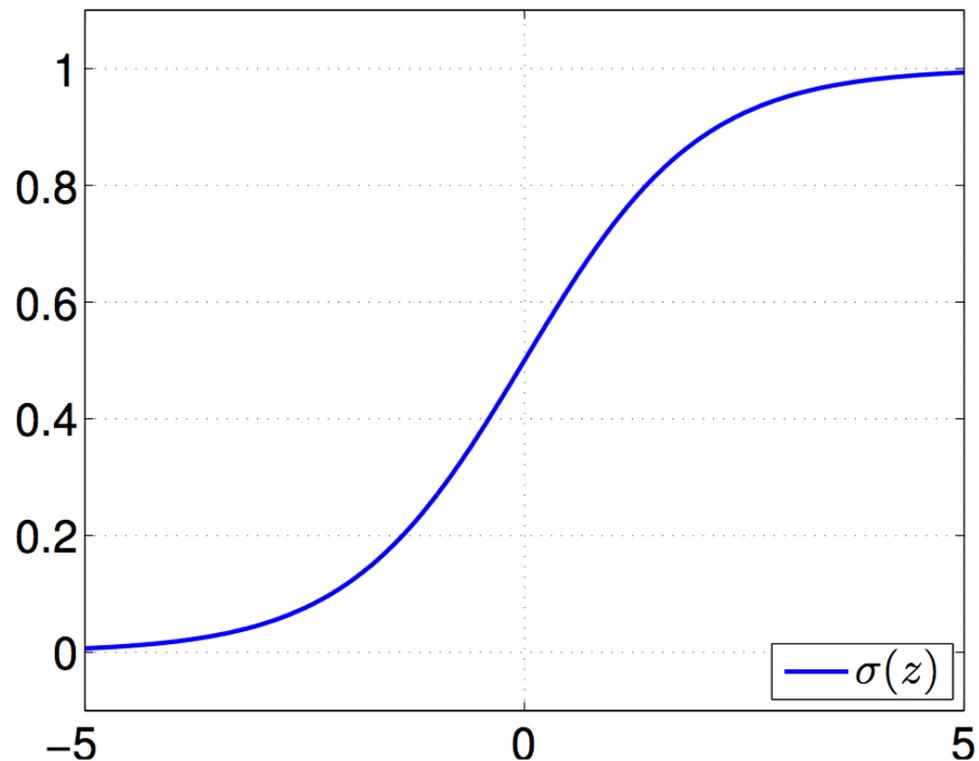
$$p(y=+1|x) = \sigma(w^T x)$$

- For binary classification problem, we have by definition

$$p(y=-1|x) = 1 - p(y=+1|x)$$

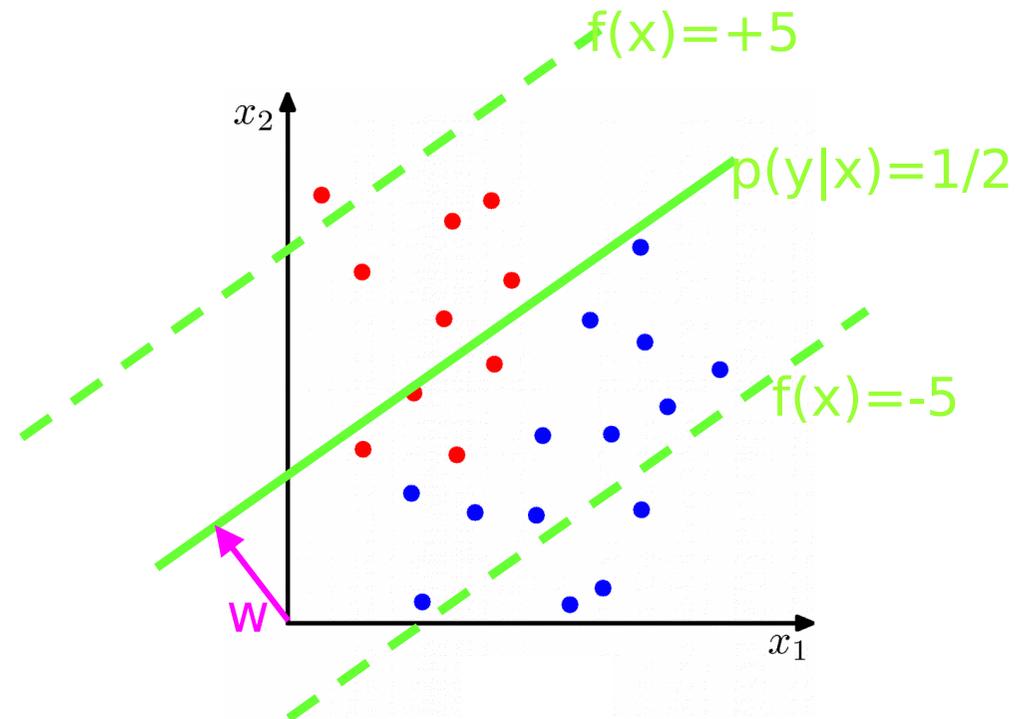
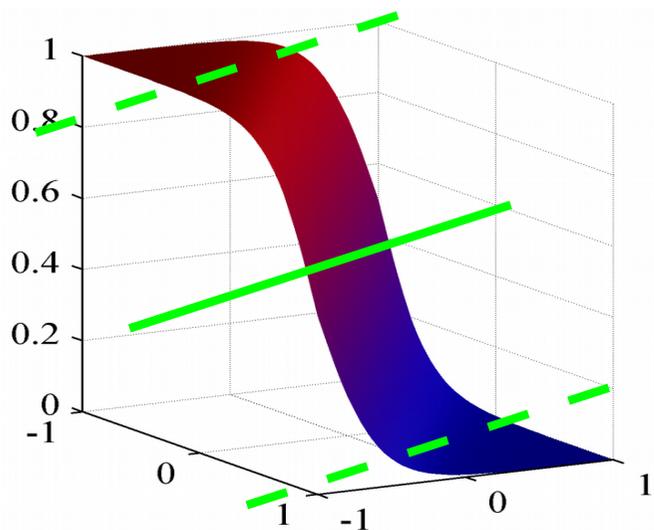
- ▶ Exercise: show that $p(y=-1|x) = \sigma(-w^T x)$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



Logistic discriminant classifier

- Map linear score function to class probabilities with sigmoid.
- The class boundary at $f(x)=0$, or equivalently $p(y|x)=1/2$.
- Soft transition between class assignment along decision boundary.



Logistic discriminant classifier

- Probability of class y given by sigmoid of score function times label

$$p(y|x) = \sigma(yw^T x)$$

- Log-likelihood of correct classification of i.i.d. data in training set

$$\begin{aligned}\log \prod_{i=1}^n p(y_i|x_i) &= \sum_{i=1}^n \log p(y_i|x_i) \\ &= \sum_{i=1}^n \log \sigma(y_i w^T x_i) \\ &= - \sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i)) \\ &= - \sum_{i=1}^n L_{\text{logistic}}(y_i, w^T x_i)\end{aligned}$$

- We have obtained the logistic loss as negative log-likelihood

Logistic discriminant estimation

- Estimate classifier from data by minimizing, e.g. L2, penalized loss:
 - ▶ Penalty reduces risk of overfitting

$$\begin{aligned} & \min_w \sum_{i=1}^n L(y_i, w^T x_i) + \lambda \frac{1}{2} w^T w \\ & = \min_w \sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i)) + \lambda \frac{1}{2} w^T w \end{aligned}$$

- Exercise 1: derive the gradient of the loss

$$\frac{\partial L(y_i, w^T x_i)}{\partial w} = -y_i(1 - p(y_i|x_i))x_i$$

- Exercise 2: Show that this is a convex optimization problem

Logistic discriminant estimation

- Estimate classifier from data by minimizing, e.g. L2, penalized loss:

$$\begin{aligned} & \min_w \sum_{i=1}^n L(y_i, w^T x_i) + \lambda \frac{1}{2} w^T w \\ & = \min_w \sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i)) + \lambda \frac{1}{2} w^T w \end{aligned}$$

- Exercise: Show that this is a convex optimization problem

- ▶ Calculate gradient of loss w.r.t. w

$$\frac{\partial L(y, w^T x)}{\partial w} = -yx \frac{1}{1 + \exp(y w^T x)}$$

- ▶ Calculate Hessian of Loss w.r.t. w

$$H(L) = yx \left(\frac{1}{1 + \exp(y w^T x)} \right)^2 \exp(y w^T x) yx^T$$

$$= \sigma(y w^T x) \sigma(-y w^T x) x x^T$$

Logistic discriminant estimation

- Consider arbitrary w with non-zero norm

$$\begin{aligned}w^T H(L) w &= w^T \left(\sigma(yw^T x) \sigma(-yw^T x) x x^T \right) w \\ &= \sigma(yw^T x) \sigma(-yw^T x) (w^T x)^2 \geq 0\end{aligned}$$

- Hessian is semi-positive definite, thus L is convex in w .
- Squared L2 norm also convex in w .

Logistic discriminant estimation

- Solve objective function using first or second order methods

$$\min_w \sum_{i=1}^n \log(1 + \exp(-y_i^T w)) + \lambda \frac{1}{2} w^T w$$

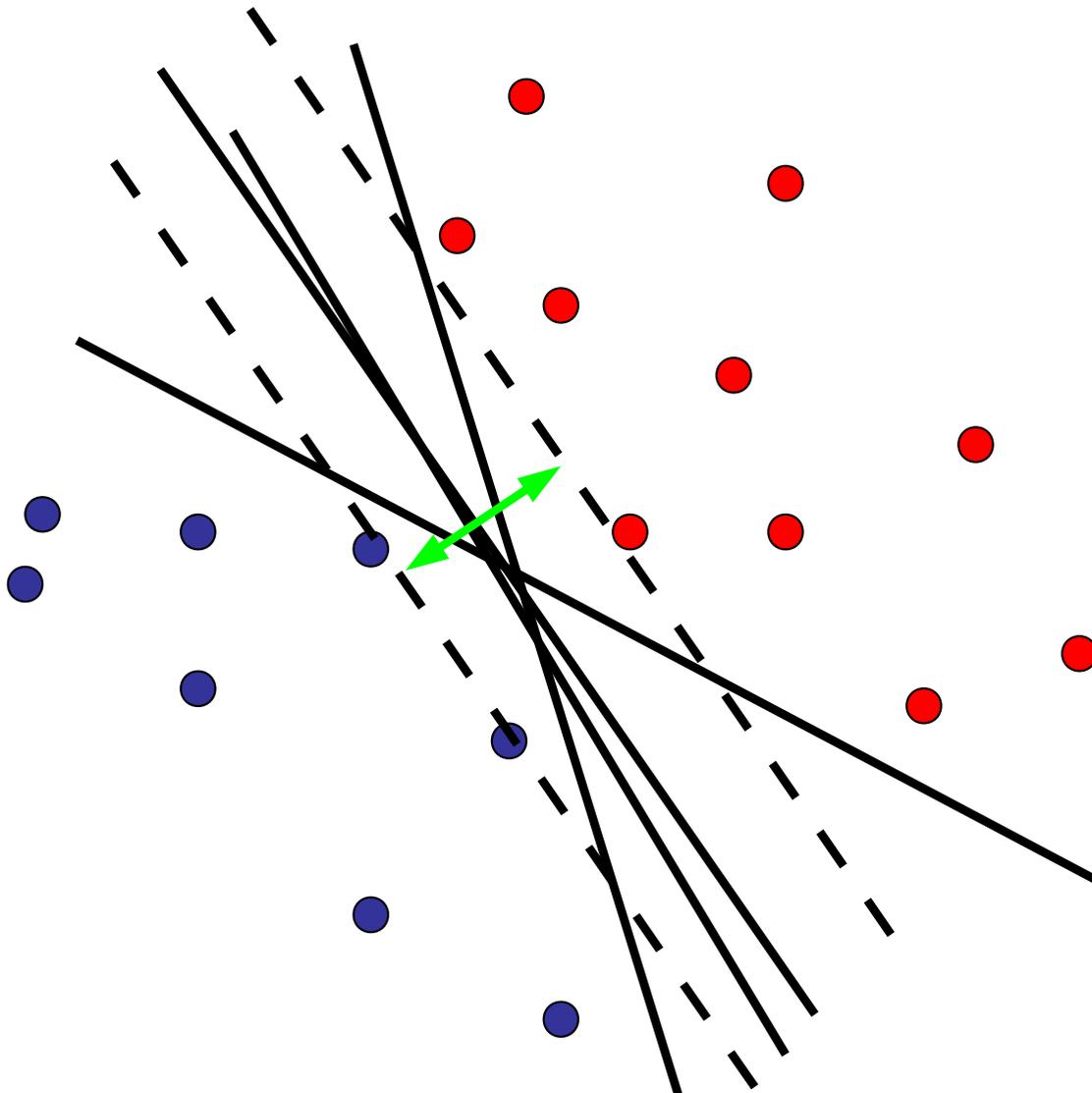
- ▶ E.g. using gradient descent, conjugate gradient descent,...
- ▶ Stochastic gradient descent for large-scale problems

- Recall the gradient $\frac{\partial L(y_i, w^T x_i)}{\partial w} = -y_i(1 - p(y_i|x_i))x_i$

- Consider gradient descent, starting from $w=0$
 - ▶ Each step we add to w a linear combination of the data points
 - ▶ Magnitude of weight given by probability of misclassification
 - ▶ Sign of weight given by the label
- The optimal w is a linear combination of the data samples
 - ▶ L2 regularization term does not change this property

Support Vector Machines

- Find linear function to separate positive and negative examples
- Which function best separates the samples ?
 - ▶ Function inducing the largest **margin**



$$y_i = +1 : w^T x_i + b > 0$$
$$y_i = -1 : w^T x_i + b < 0$$

Support vector machines

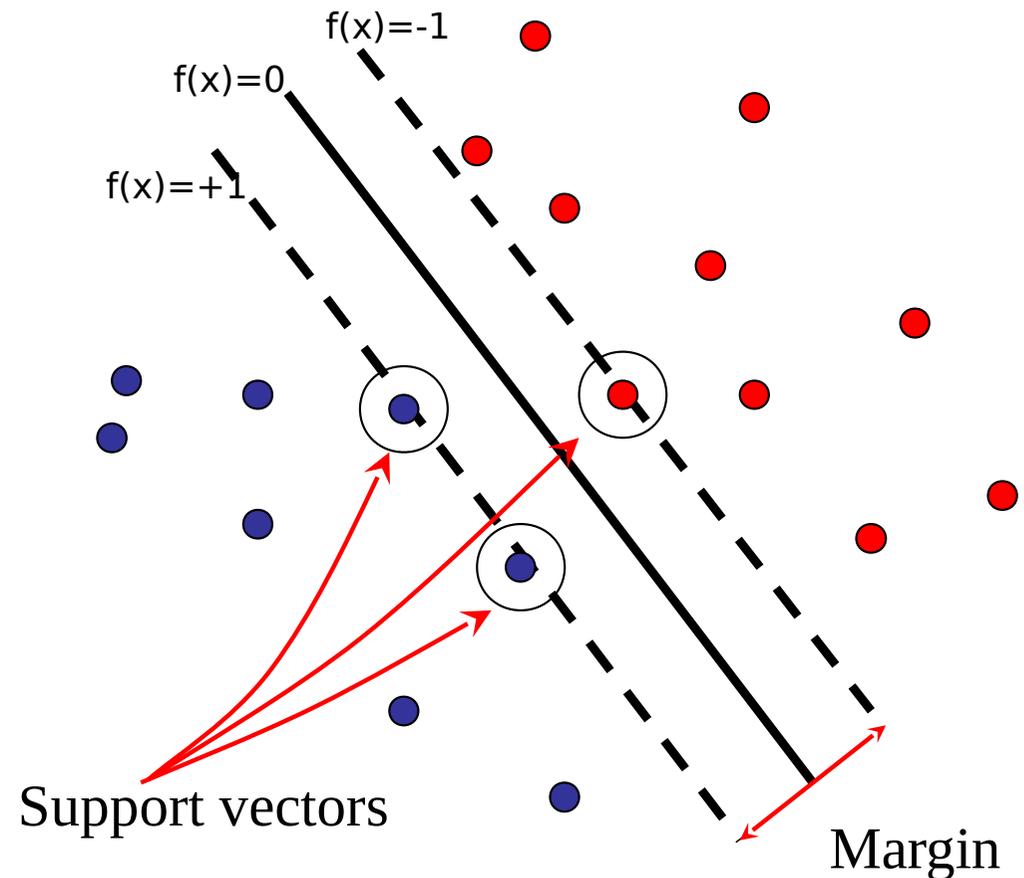
- Without loss of generality, define function value at margin as +/- 1
- Now constrain w to that all points fall on correct side of the margin:

$$y_i(w^T x_i + b) \geq 1$$

- By construction we have that the “support vectors”, the ones that define the margin, have function values

$$w^T x_i + b = y_i$$

- Express the size of the margin in terms of w .



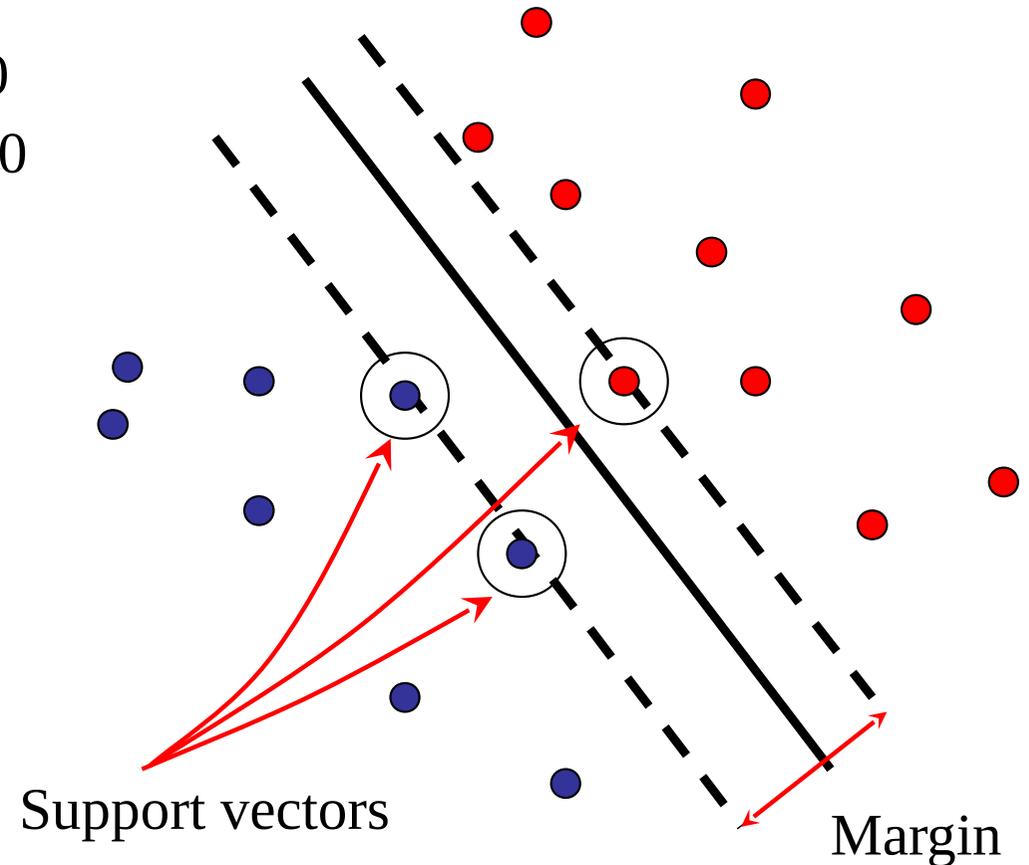
Support vector machines

- Let's consider a support vector x from the positive class $f(x) = w^T x + b = 1$
- Let z be its projection on the decision plane
 - ▶ Since w is normal vector to the decision plane, we have $z = x - \alpha w$
 - ▶ and since z is on the decision plane $f(z) = w^T (x - \alpha w) + b = 0$

- Solve for alpha
$$w^T (x - \alpha w) + b = 0$$
$$w^T x + b - \alpha w^T w = 0$$
$$\alpha w^T w = 1$$
$$\alpha = \frac{1}{\|w\|_2^2}$$

- Margin is twice distance from x to z

$$\|x - z\|_2 = \|x - (x - \alpha w)\|_2$$
$$\|\alpha w\|_2 = \alpha \|w\|_2$$
$$\frac{\|w\|_2}{\|w\|_2^2} = \frac{1}{\|w\|_2}$$

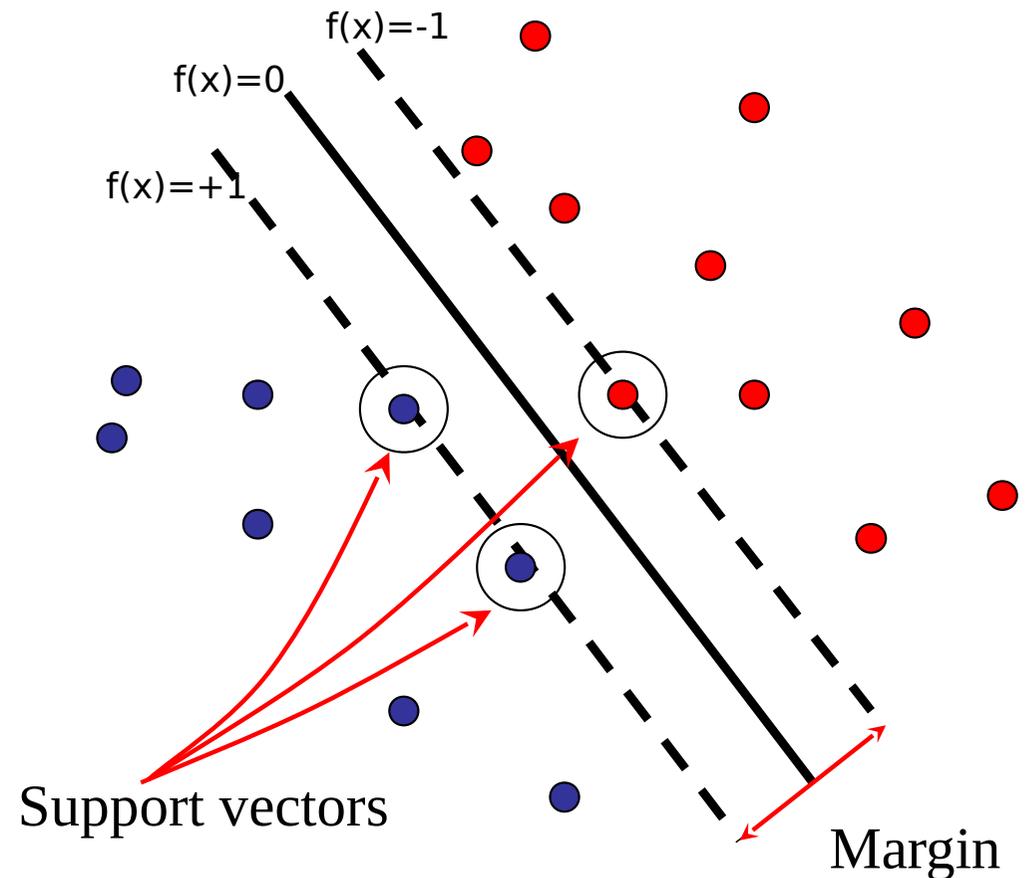


Support vector machines

- To find the maximum-margin separating hyperplane, we
 - ▶ Maximize the margin, while ensuring correct classification
 - ▶ Minimize the norm of w , s.t. $\forall_i: y_i(w^T x_i + b) \geq 1$
- Solve using quadratic program with linear inequality constraints over $p+1$ variables

$$\operatorname{argmin}_{w,b} \frac{1}{2} w^T w$$

subject to $y_i(w^T x_i + b) \geq 1$

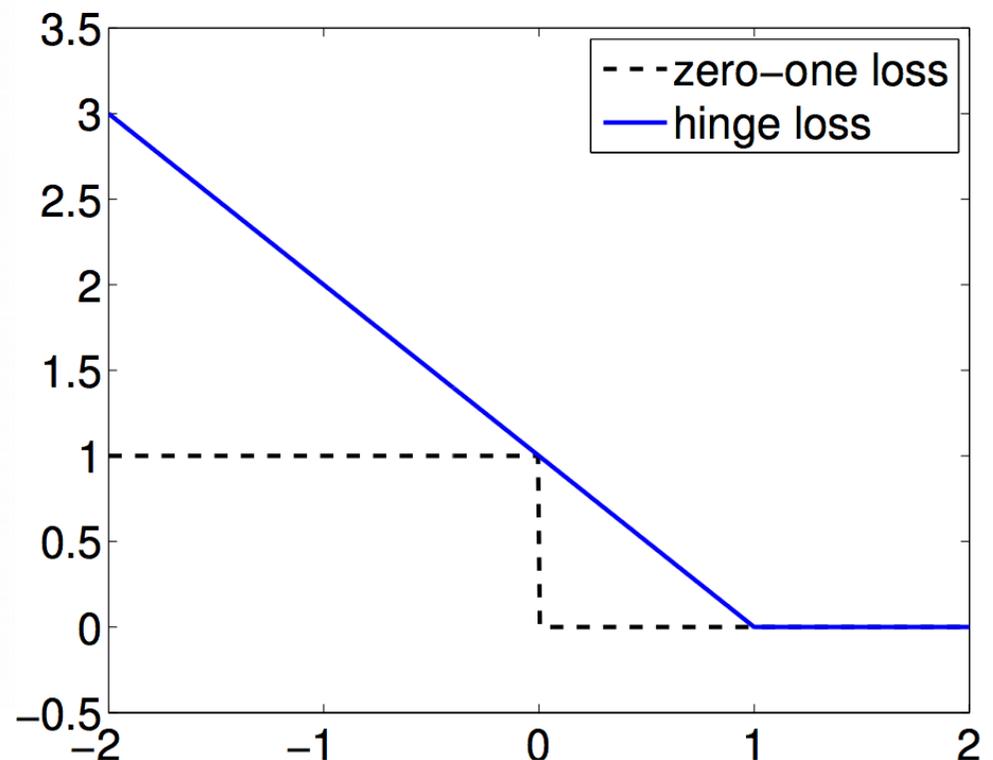
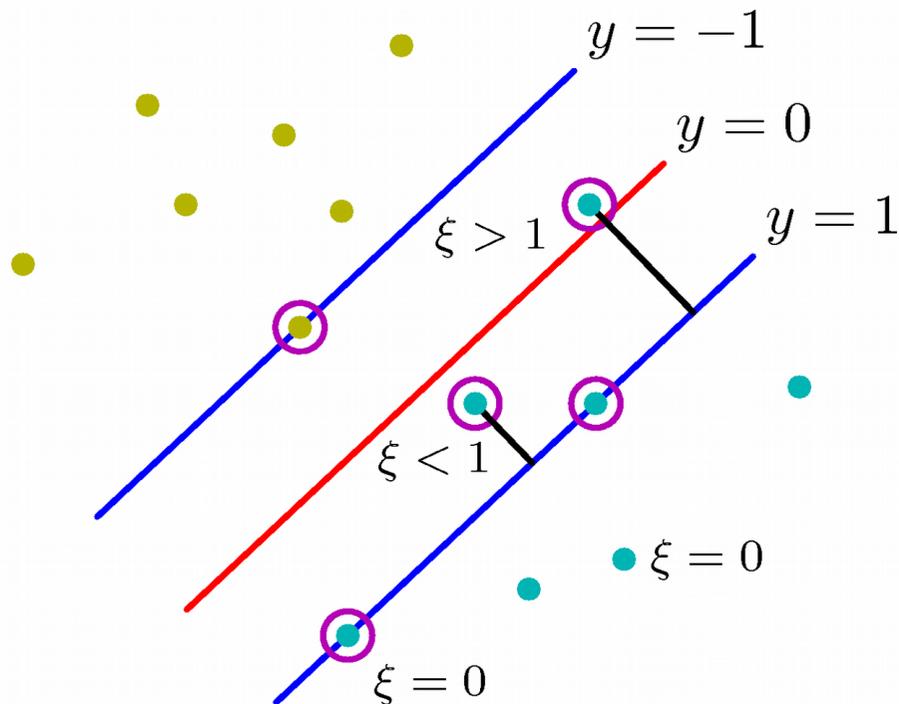


Support vector machines: inseparable classes

- For non-separable classes we incorporate hinge-loss

$$L(y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))$$

- Recall: convex and piecewise linear upper bound on zero/one loss.
 - ▶ Zero if point on the correct side of the margin
 - ▶ Otherwise given by absolute difference from score at margin



Support vector machines: inseperable classes

- Minimize penalized loss function

$$\min_{w,b} \lambda \frac{1}{2} w^T w + \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

- ▶ Quadratic function, plus **piecewise linear functions**.

- Can again be transformed to a quadratic program

- ▶ Define “slack variables” that measure the loss for each data point
- ▶ Should be non-negative, and at least as large as the loss

$$\min_{w,b,\{\xi_i\}} \lambda \frac{1}{2} w^T w + \sum_i \xi_i$$

$$\text{subject to } \forall_i: \xi_i \geq 0 \text{ and } \xi_i \geq 1 - y_i(w^T x_i + b)$$

Support vector machines: solution

- Minimize penalized loss function

$$\min_{w, b, \{\xi_i\}} \lambda \frac{1}{2} w^T w + \sum_i \xi_i$$

subject to $\forall_i: \xi_i \geq 0$ and $\xi_i \geq 1 - y_i(w^T x_i + b)$

- Solution for w will be a linear combination of the input data
 - ▶ Split w into a part inside and outside the span of the data

$$w = w_p + w_o \quad \forall_i: w_o^T x_i = 0 \quad w_p = \sum_i \alpha_i x_i$$

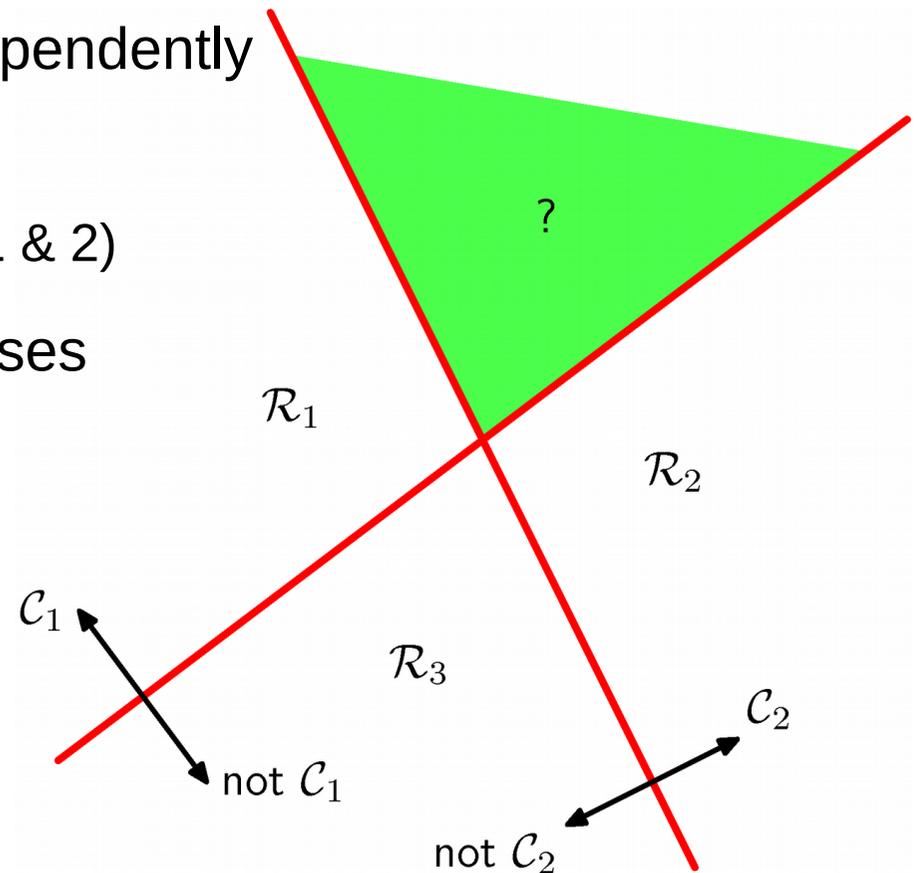
- ▶ Only norm of w depends on part of w outside the data span
- ▶ Note that

$$w^T w = w_p^T w_p + w_o^T w_o \geq w_p^T w_p$$

- ▶ Therefore optimal w is a linear combination of the data
- This is a special case of the more general “representer theorem”

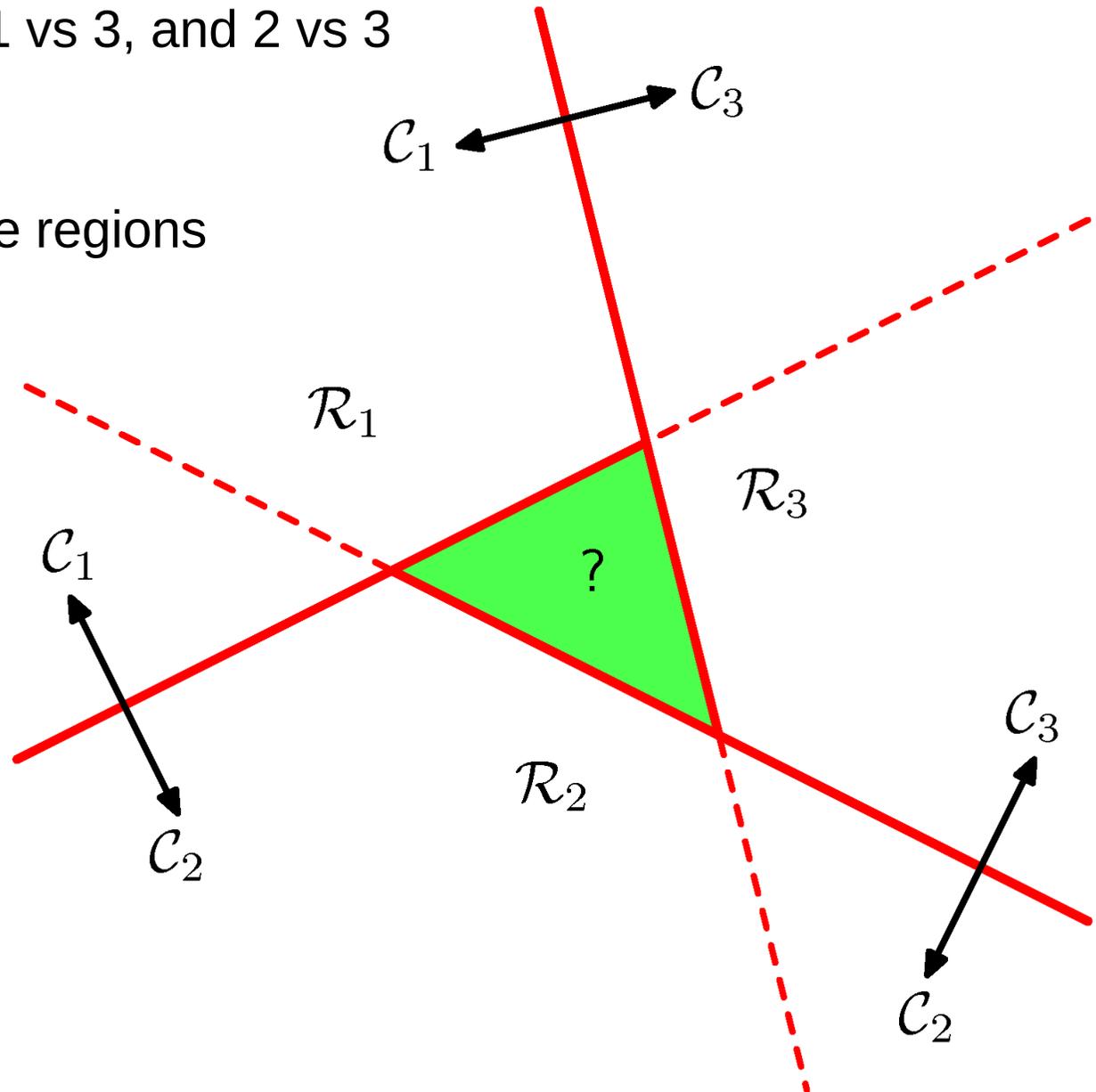
Dealing with more than two classes

- So far, we have only considered the, useful, case for two classes
 - ▶ E.g., is this email spam or not ?
- Many practical problems have more classes
 - ▶ E.g., which fruit is placed on the supermarket weight scale: apple, orange, or banana ?
- First idea: construction from multiple binary classifiers
 - ▶ Learn binary “base” classifiers independently
- One vs rest approach:
 - ▶ Train: 1 vs (2 & 3), 2 vs (1 & 3), 3 vs (1 & 2)
- Issue: regions claimed by several classes



Dealing with more than two classes

- One vs one approach:
 - ▶ Train: 1 vs 2, and 1 vs 3, and 2 vs 3
- Issue: conflicts in some regions



Dealing with more than two classes

- Instead: define a separate linear score function for each class

$$f_k(x) = w_k^T x$$

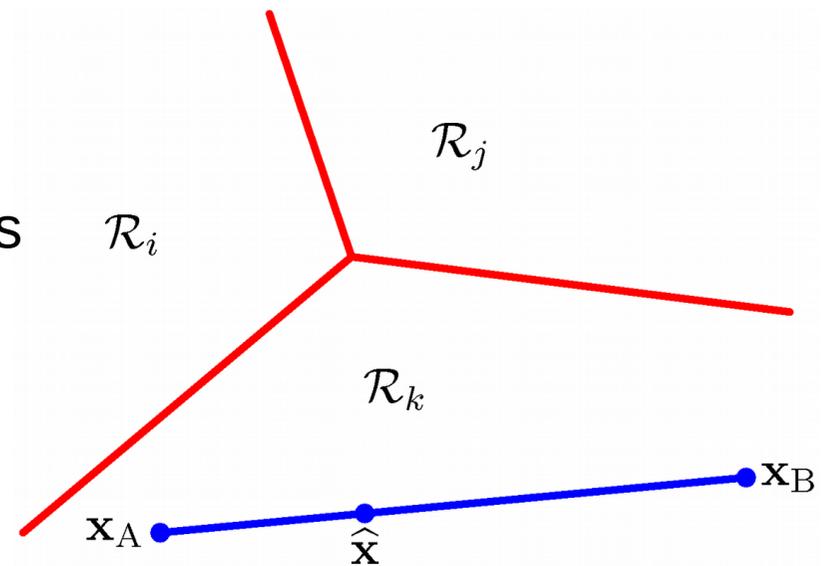
- Assign sample to the class of the function with maximum value

$$y = \arg \max_k f_k(x)$$

- Exercise 1: give the expression for points where two classes have equal score

- Exercise 2: show that the set of points assigned to a class is convex

- ▶ If two points are assigned to a class, then all points on connecting line are also assigned to that class.



Multi-class logistic discriminant classifier

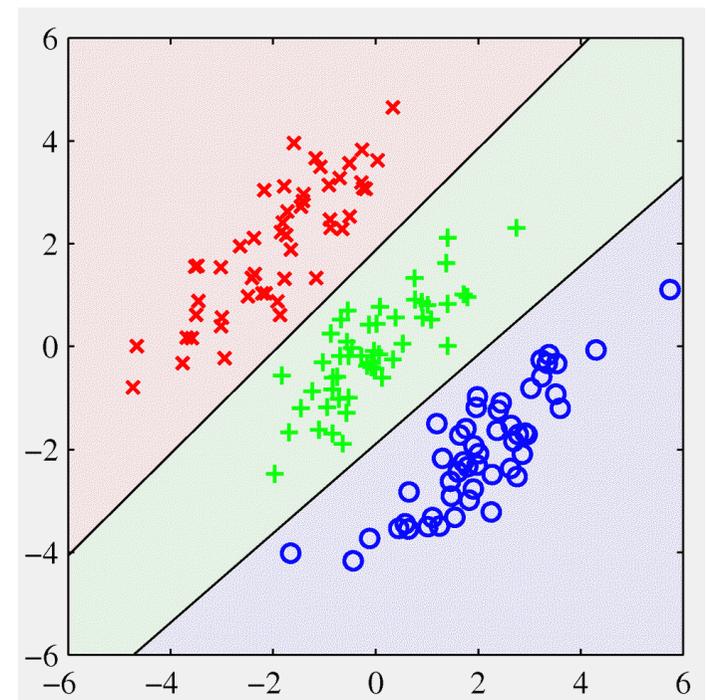
- Map score functions to class probabilities with “soft-max”

$$f_k(x) = w_k^T x \qquad p(y=c|x) = \frac{\exp(f_c(x))}{\sum_{k=1}^K \exp(f_k(x))}$$

- ▶ The class probability estimates are non-negative, and sum to one.
- Relative probability of classes changes exponentially with the difference in the linear score functions

$$\frac{p(y=c|x)}{p(y=k|x)} = \frac{\exp(f_c(x))}{\exp(f_k(x))} = \exp(f_c(x) - f_k(x))$$

- For any given pair of classes, they are equally likely on a hyperplane in the feature space



Multi-class logistic discriminant: estimation

- Consider the likelihood of correct classification of i.i.d. data in training set

$$\begin{aligned}\log \prod_{i=1}^n p(y_i|x_i) &= \sum_{i=1}^n \log p(y_i|x_i) \\ &= \sum_{i=1}^n \left(f_{y_i}(x_i) - \log \sum_{k=1}^K \exp(f_k(x_i)) \right)\end{aligned}$$

- As before, we define loss function as negative log-likelihood

$$L(y, \{f_k(x)\}) = -f_y(x) + \log \sum_{k=1}^K \exp(f_k(x))$$

- Estimate model by means of penalized empirical risk

$$\min_w \sum_{i=1}^n L(y_i, \{f_k(x_i)\}) + \lambda \frac{1}{2} \sum_{k=1}^K w_k^T w_k$$

Multi-class logistic discriminant: estimation

- Derivative of loss function has an intuitive interpretation
 - ▶ Focus on points with poor classification, w is linear combination of x 's

$$L = \sum_{i=1}^n L(y_i, \{f_k(x_i)\})$$

$$\frac{\partial L}{\partial w_k} = \sum_{i=1}^n ([y_i=k] - p(y_i=k|x_i)) x_i$$

- Gradient is zero when $\sum_{i=1}^n [y_i=k] x_i = \sum_{i=1}^n p(y_i=k|x_i) x_i$

- ▶ If x also contains the constant 1 as last element then empirical count of each class matches expected count.

$$\sum_{i=1}^n [y_i=k] = \sum_{i=1}^n p(y_i=k|x_i)$$

- ▶ Therefore, for each class 1st order moment matches for empirical distribution and the model's class conditional distribution.

$$\frac{\sum_{i=1}^n [y_i=k] x_i}{\sum_{i=1}^n [y_i=k]} = \frac{\sum_{i=1}^n p(y_i=k|x_i) x_i}{\sum_{i=1}^n p(y_i=k|x_i)}$$

Summary of linear classifiers

- Two most widely used binary linear classifiers:
 - ▶ Logistic discriminant, also considered the extension to >2 classes.
 - ▶ Support vector machines, similar multi-class extensions exist.
- Both minimize convex upper bounds on the 0/1 loss
- In both cases the optimal weight vector w is a linear combination of the data points

$$w = \sum_{i=1}^n \alpha_i x_i$$

- Therefore, **we only need the inner-products between data points to use linear classifiers.** This also holds for the optimization of w .

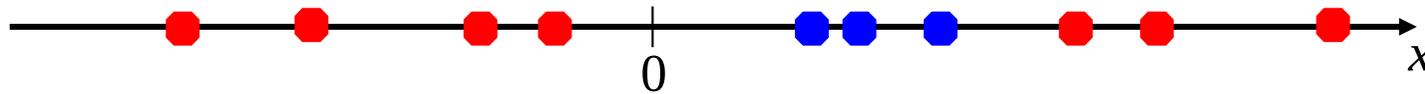
$$\begin{aligned} f(x) &= w^T x + b \\ &= \sum_{i=1}^n \alpha_i (x_i^T x) + b \end{aligned}$$

Course content

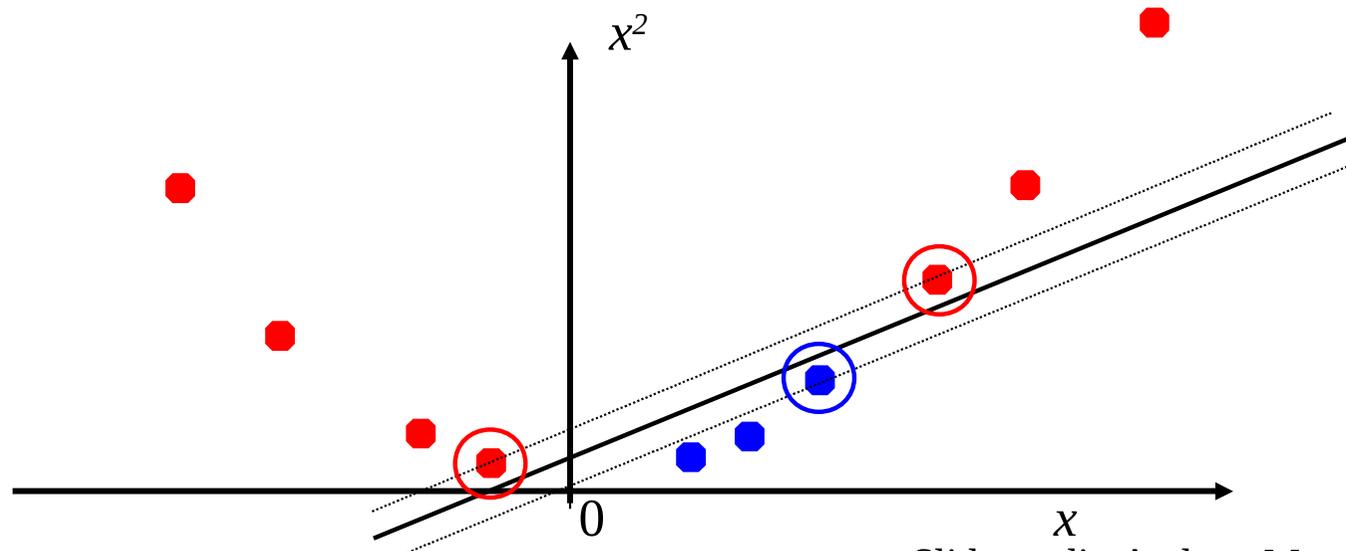
- Introduction
- Linear classification
- **Non-linear classification with kernels**
- Kernel-trick more generally
- Bias-variance decomposition

Nonlinear Classification

- So far we just considered linear classifiers.
- Obviously limits the problems that can be addressed.
- What to do if the data is not linearly separable?

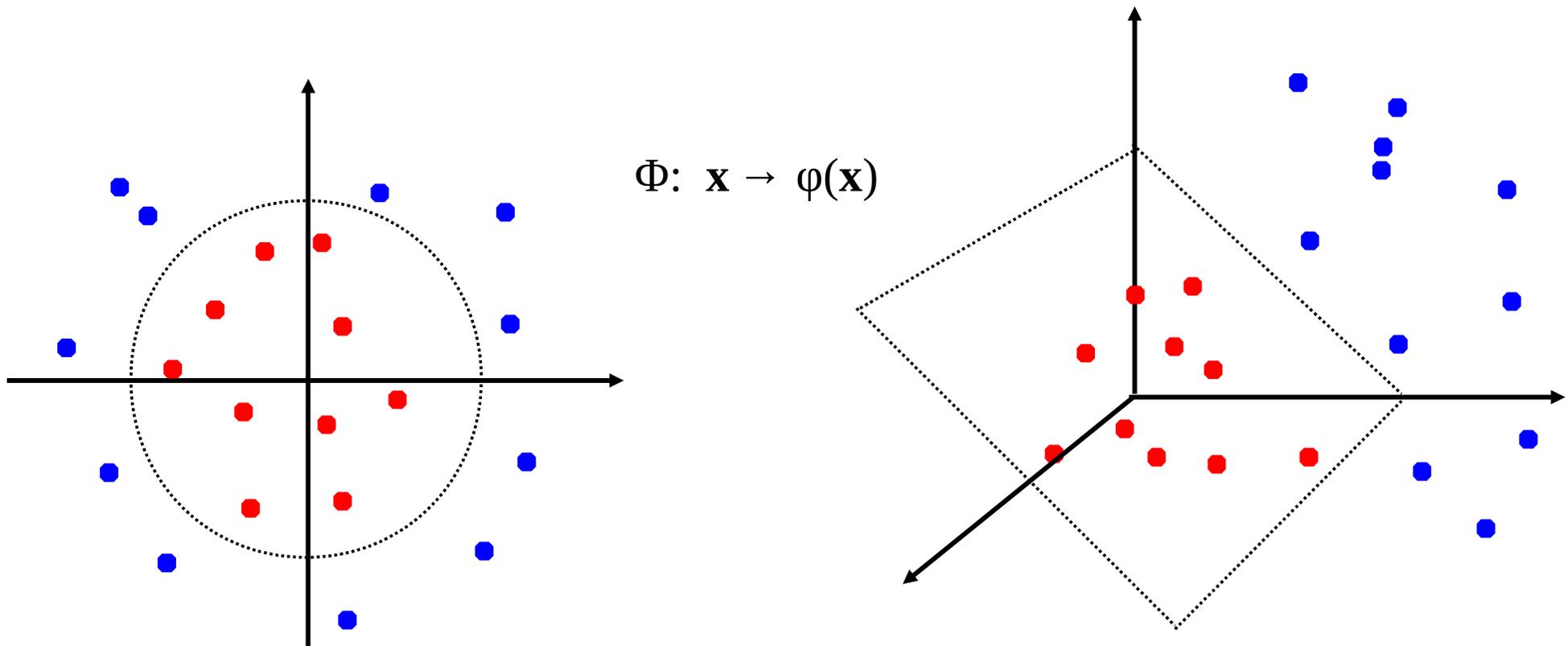


- Similar to what we considered last week for regression with higher-order polynomials, we can do linear classification on non-linear features. For example augment map the data to \mathbb{R}^2 by adding x^2 .



Non-linear feature mappings for classification

- Map the original input space to some higher-dimensional feature space where the training set is separable
- Data occupies a (non-linear) subspace of dimension equal to the original space.
- Which features could separate this 2dimensional data linearly ?



Non-linear feature mappings for classification

- Remember that for classification we only need dot-products.
- Let's calculate the dot-product explicitly for our example.
 - ▶ New dot-product easily computed from the original one.

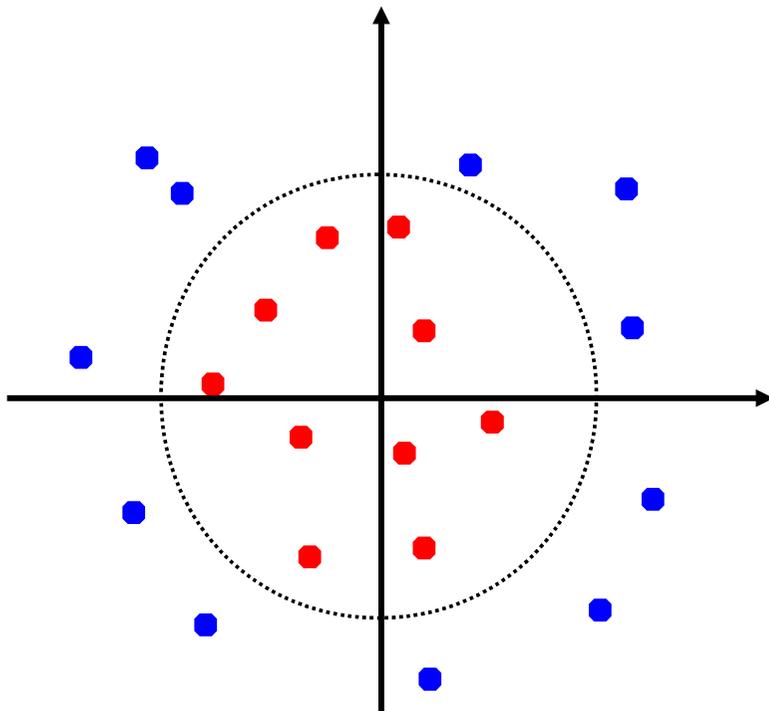
$$\varphi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

$$k(\mathbf{x}, \mathbf{z}) = \varphi(\mathbf{x})^T \varphi(\mathbf{z}) = ?$$

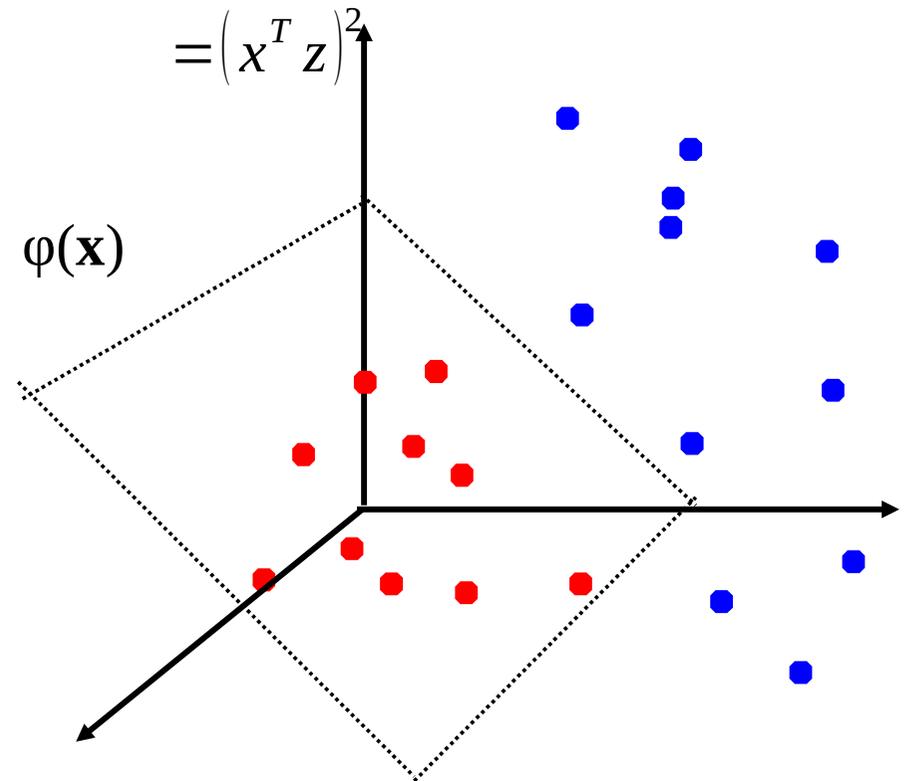
$$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 x_2 z_1 z_2$$

$$= (x_1 z_1 + x_2 z_2)^2$$

$$= (\mathbf{x}^T \mathbf{z})^2$$

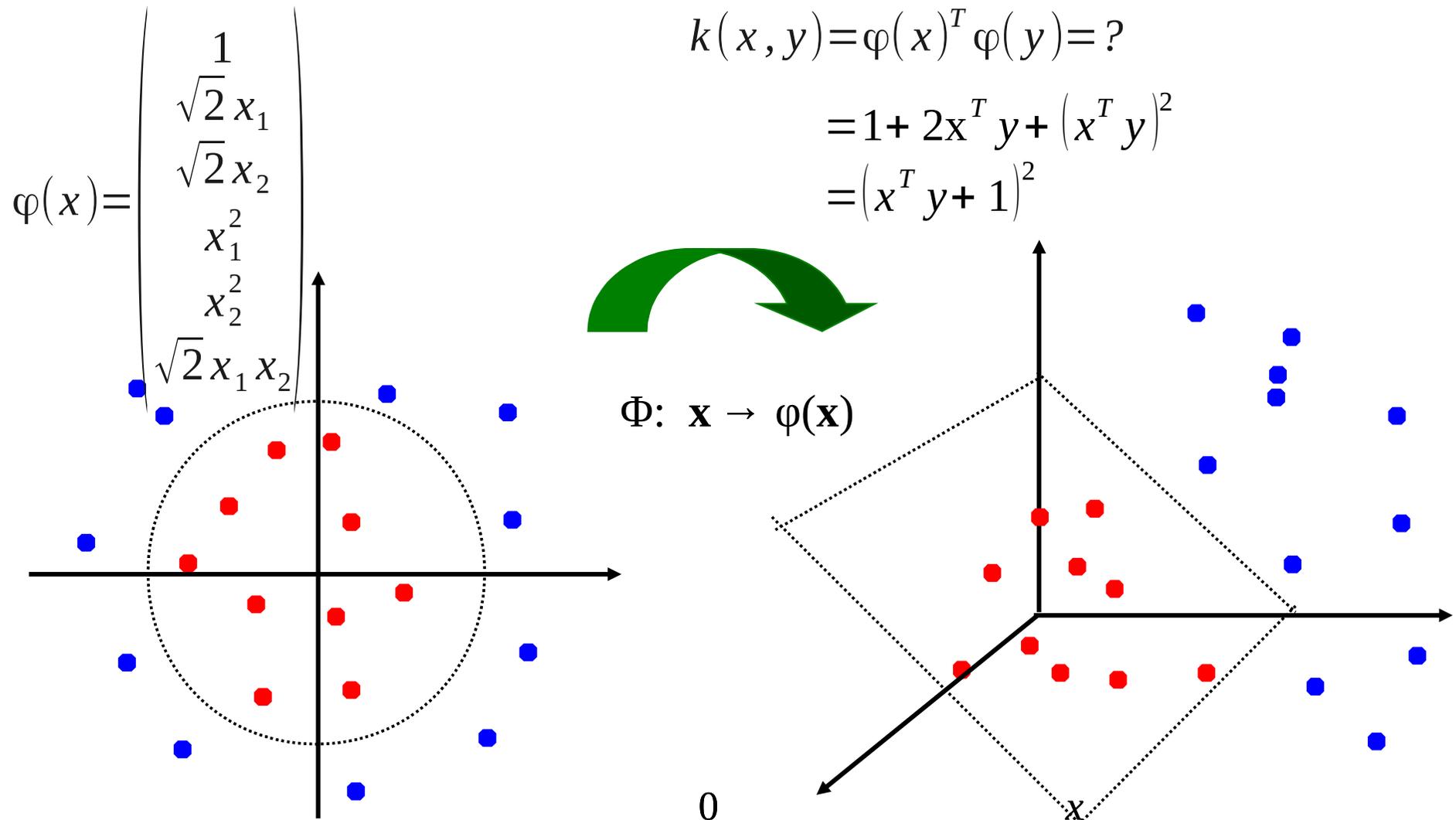


$\Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x})$



Non-linear feature mappings for classification

- Suppose we also want to keep the original features to still be able to implement linear functions
 - ▶ Again efficient computation in 6d, roughly at cost of 2d dot-product



Non-linear feature mappings for classification

- What happens if we do the same for higher dimensional data

- ▶ Which feature vector $\varphi(x)$ corresponds to it ?

$$k(x, y) = (x^T y + 1)^2 = 1 + 2x^T y + (x^T y)^2$$

- ▶ First term, encodes an additional 1 in each feature vector
- ▶ Second term, encodes scaling of the original features by sqrt(2)

- ▶ Let's consider the third term $(x^T y)^2 = (x_1 y_1 + \dots + x_D y_D)^2$

$$= \sum_{d=1}^D (x_d y_d)^2 + 2 \sum_{d=1}^{D-1} \sum_{i=d+1}^D (x_d y_d)(x_i y_i)$$

$$= \sum_{d=1}^D x_d^2 y_d^2 + 2 \sum_{d=1}^{D-1} \sum_{i=d+1}^D (x_d x_i)(y_d y_i)$$

- ▶ In total we have $1 + 2D + D(D-1)/2$ features !
- ▶ But computed as efficiently as dot-product in original space

$$\varphi(x) = \left(\underbrace{1, \sqrt{2} x_1, \sqrt{2} x_2, \dots, \sqrt{2} x_D}_{\text{Original features}}, \underbrace{x_1^2, x_2^2, \dots, x_D^2}_{\text{Squares}}, \underbrace{\sqrt{2} x_1 x_2, \dots, \sqrt{2} x_1 x_D, \dots, \sqrt{2} x_{D-1} x_D}_{\text{Products of two distinct elements}} \right)^T$$

Original features

Squares

Products of two distinct elements

Nonlinear classification with kernels

- The kernel trick: instead of explicitly computing the feature transformation $\varphi(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

- This allows us to obtain nonlinear classification in the original space:

$$\begin{aligned} f(x) &= b + w^T \varphi(x) \\ &= b + \sum_i \alpha_i \varphi(x)^T \varphi(x_i) \\ &= b + \sum_i \alpha_i k(x, x_i) \\ &= b + \alpha^T k(x, \cdot) \end{aligned}$$

$$\begin{aligned} w^T w &= \sum_i \sum_j \alpha_i \alpha_j \varphi(x_i)^T \varphi(x_j) \\ &= \sum_i \sum_j \alpha_i \alpha_j k(x_i, x_j) \\ &= \alpha^T K \alpha \end{aligned}$$

Summary of classification

- Linear classifiers learned by minimizing convex cost functions
 - ▶ Logistic loss: smooth objective, minimized using gradient descent, etc.
 - ▶ Hinge loss: piecewise linear objective, quadratic programming
 - ▶ Both require only computing inner product between data points
- Non-linear classification can be done with linear classifiers over new features that are non-linear functions of the original features
 - ▶ Kernel functions efficiently compute inner products in (very) high-dimensional spaces, can even be infinite dimensional.
- Using kernel functions non-linear classification has drawbacks
 - ▶ Requires storing the data with non-zero weights, memory cost
 - ▶ Kernel evaluations for test point may be computationally expensive

Course content

- Introduction
- Linear classification
- Non-linear classification with kernels
- **Kernel-trick more generally**
- Bias-variance decomposition

Representation by pairwise comparisons

- We can think of a kernel function as a pairwise comparison function

$$K: X \times X \rightarrow R$$

- Represent a set of n data points by the $n \times n$ matrix $[K]_{ij} = K(x_i, x_j)$
- Always an $n \times n$ matrix, whatever the nature of the data
 - ▶ Same algorithms will work for any type of data: images, text...
- Modularity between the choice of K and the choice of algorithms.
- Poor scalability with respect to the data size (squared in n).
- We will restrict attention to a specific class of kernels.

Positive definite kernels

- Definition: A positive definite kernel on the set X is a function

$$K: X \times X \rightarrow R$$

which is symmetric:

$$\forall (x, x') \in X^2: K(x, x') = K(x', x)$$

and which satisfies

$$\begin{aligned} & \forall n \in N \\ & \forall (x_1, \dots, x_n) \in R^n \text{ and } (a_1, \dots, a_n) \in R^n \\ & \sum_{i=1}^n \sum_{j=1}^n a_i a_j K(x_i, x_j) \geq 0 \end{aligned}$$

- Equivalently, a kernel K is positive definite if and only if, for any n and any set of n points, the similarity matrix K is positive semidefinite:

$$a^T K a \geq 0$$

The simplest positive definite kernel

- Lemma: The kernel function defined by the inner product over vectors is a positive definite kernel.
 - ▶ This kernel is known as the “linear kernel”

$$K: X \times X \rightarrow R$$
$$\forall (x, x') \in X^2: K(x, x') = x^T x'$$

- Proof

- ▶ Symmetry: $K(x, x') = x^T x' = (x')^T x = K(x', x)$

- ▶ Positive definiteness:

$$\sum_{i=1}^n \sum_{j=1}^n a_i a_j K(x_i, x_j) = \sum_{i=1}^n \sum_{j=1}^n a_i a_j x_i^T x_j = \left\| \sum_{i=1}^n a_i x_i \right\|_2^2 \geq 0$$

More generally: for any embedding function

- Lemma: The kernel function defined by the inner product over data points embedded in a vector space by a function φ is a positive definite kernel.

$$K: X \times X \rightarrow R$$
$$\forall (x, x') \in X^2: K(x, x') = \langle \varphi(x), \varphi(x') \rangle_H$$

- Proof

- ▶ Symmetry: $K(x, x') = \langle \varphi(x), \varphi(x') \rangle_H = \langle \varphi(x'), \varphi(x) \rangle_H = K(x', x)$

- ▶ Positive definiteness:

$$\sum_{i=1}^n \sum_{j=1}^n a_i a_j K(x_i, x_j) = \sum_{i=1}^n \sum_{j=1}^n a_i a_j \langle \varphi(x_i), \varphi(x_j) \rangle_H = \left\| \sum_{i=1}^n a_i \varphi(x_i) \right\|_H^2 \geq 0$$

Conversely: Kernels as inner products

- Theorem (Aronszajn, 1950)

K is a positive definite kernel on the set X if and only if there exists a Hilbert space H and a mapping

$$\Phi: X \rightarrow H$$

such that for any x and x' in X

$$K(x, x') = \langle \varphi(x), \varphi(x') \rangle_H$$

- Establishes the correspondence between kernels and representations.

The kernel trick

- Choosing a p.d. kernel K on a set X amounts to embedding the data in a Hilbert space: there exists a Hilbert space H and a mapping $\Phi : X \rightarrow H$

such that for all x and x' in X

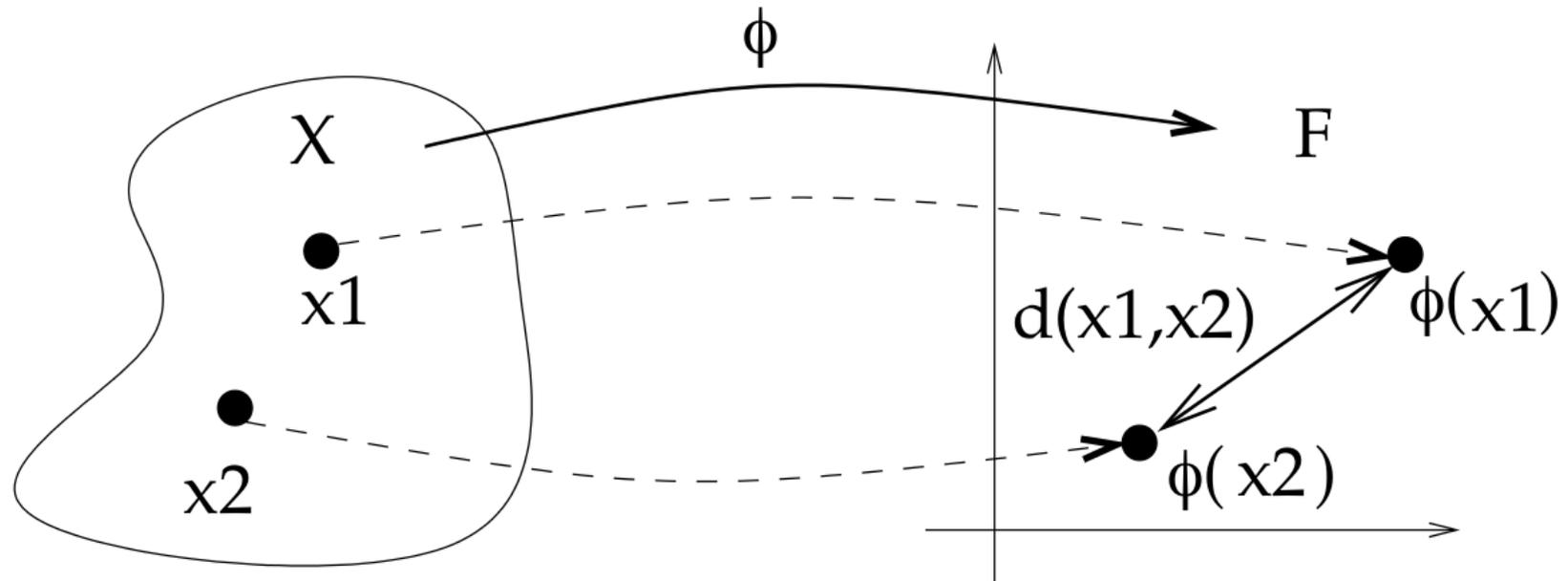
$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle_H.$$

- This mapping might not be explicitly given, nor convenient to work with in practice, e.g. for very large or even infinite dimensions.
- The “trick” is to work implicitly in the feature space H by means of kernel evaluations.

The kernel trick

- Any algorithm to process finite dimensional vectors that can be expressed only in terms of pairwise inner products can be applied to potentially infinite-dimensional vectors in the feature space of a p.d. kernel by replacing each inner product evaluation by a kernel evaluation.
- This statement is trivially true, since the kernel computes the inner product in the associated RKHS.
- The practical implications of this “trick” are important.
- Vectors in the feature space are only manipulated implicitly, through pairwise inner products, there is no need to explicitly represent any data in the feature space.

Example 1: computing distances in the feature space



$$\begin{aligned}d_k(x, x')^2 &= \|\varphi(x) - \varphi(x')\|_H^2 \\ &= \langle \varphi(x) - \varphi(x'), \varphi(x) - \varphi(x') \rangle_H \\ &= \langle \varphi(x), \varphi(x) \rangle_H + \langle \varphi(x'), \varphi(x') \rangle_H - 2 \langle \varphi(x), \varphi(x') \rangle_H \\ &= k(x, x) + k(x', x') - 2k(x, x')\end{aligned}$$

Distance for the Gaussian kernel

- The Gaussian kernel with bandwidth sigma is given by

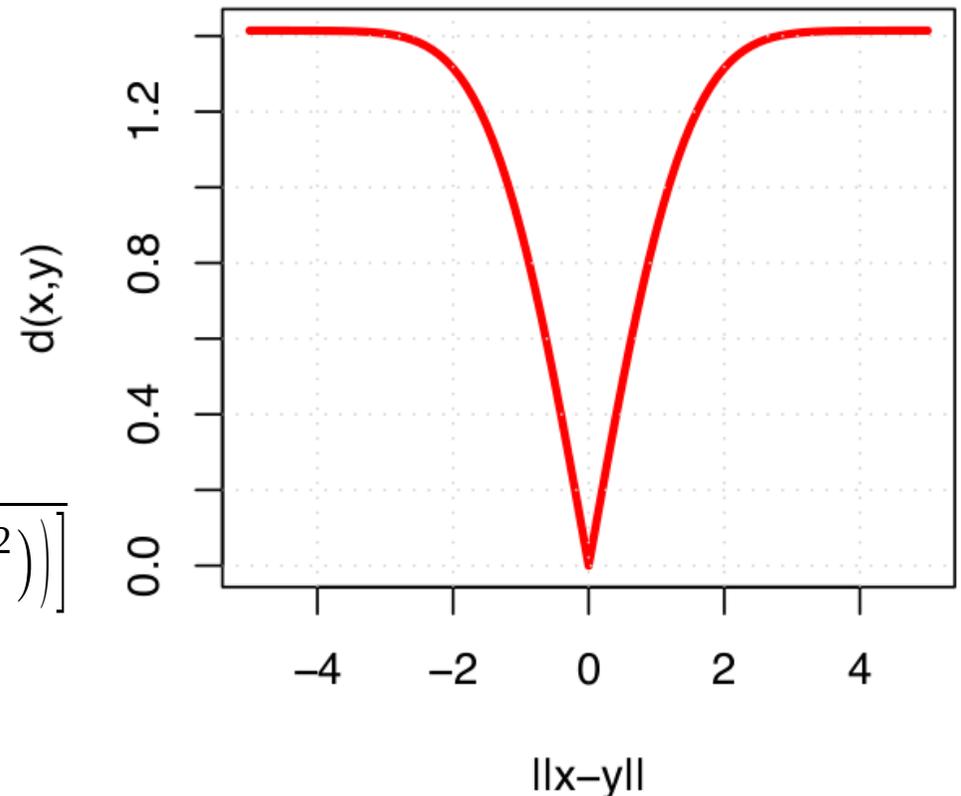
$$k(x, x') = \exp\left(-\|x - x'\|_2 / (2\sigma^2)\right)$$

- In the feature space, all points are embedded on the unit sphere since

$$k(x, x) = \|\varphi(x)\|_H^2 = 1$$

- The distance in the feature space between x and x' is given by

$$d_k(x, x') = \sqrt{2\left[1 - \exp\left(-\|x - x'\|_2 / (2\sigma^2)\right)\right]}$$

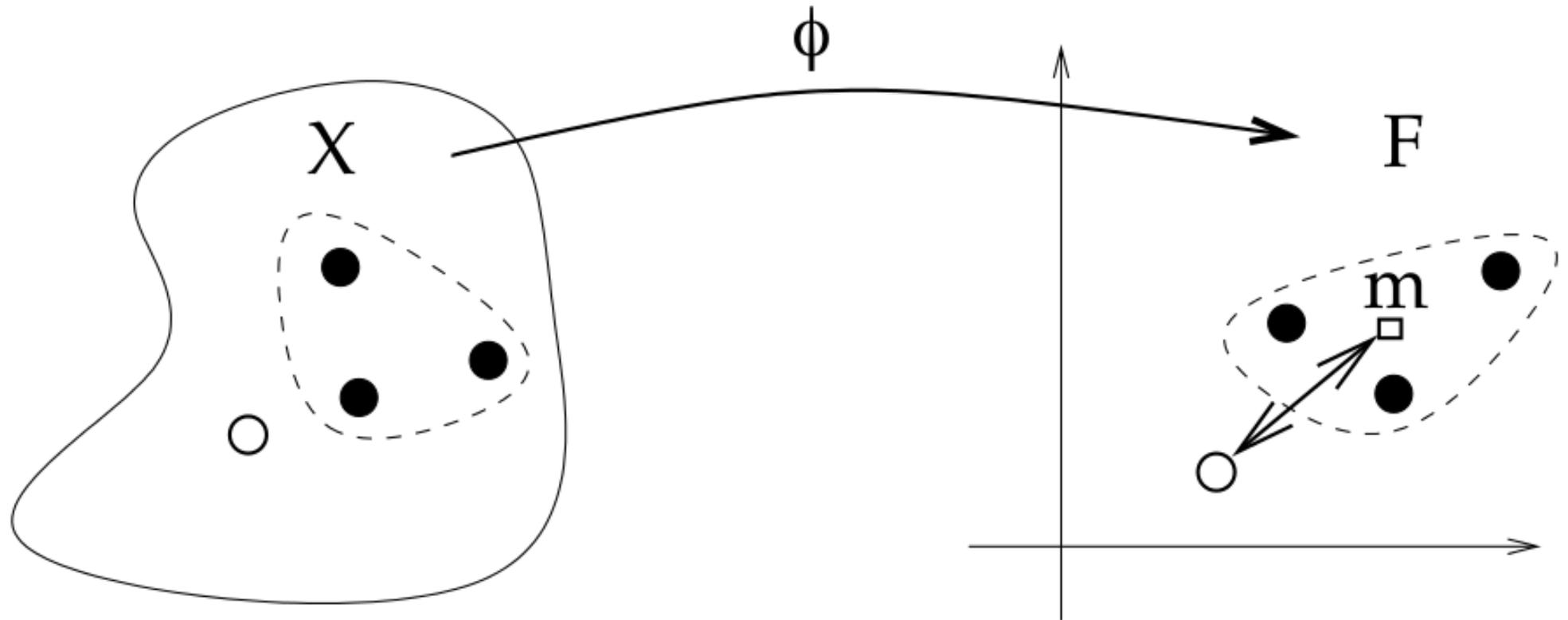


Example 2: distance between a point and a set

- Let S be a finite set of points in X : $S = (x_1, \dots, x_n)$
- How to define and compute the similarity between any point x in X and the set S ?
- The following is a simple approach:
 - ▶ Map all points to the feature space
 - ▶ Summarize S by the barycenter of the points $m = \frac{1}{n} \sum_{i=1}^n \varphi(x_i)$
 - ▶ Define the distance between x and S as

$$d_k(x, S) = \|\varphi(x) - m\|_H$$

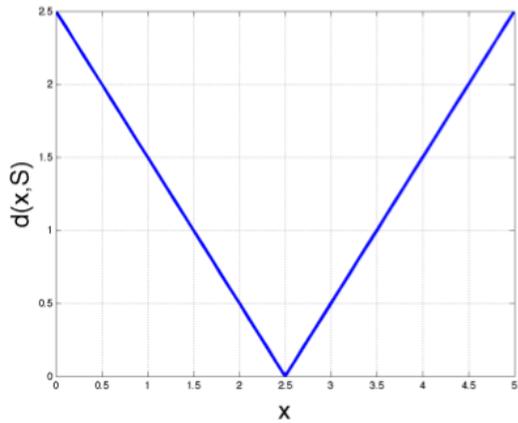
Example 2: distance between a point and a set



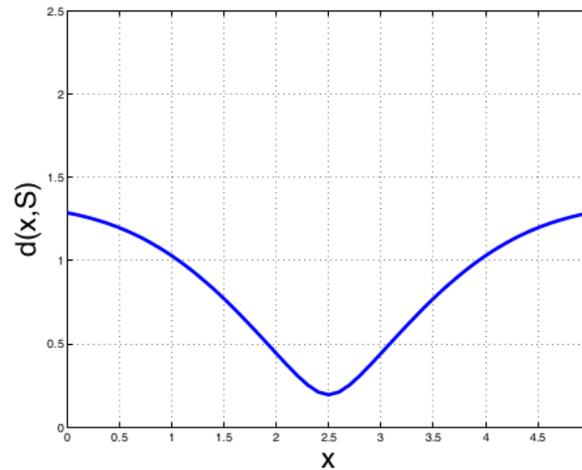
$$\begin{aligned}d_k(x, S) &= \|\varphi(x) - m\|_H \\ &= \left\| \varphi(x) - \frac{1}{n} \sum_{i=1}^n \varphi(x_i) \right\|_H \\ &= \sqrt{k(x, x) - \frac{2}{n} \sum_{i=1}^n k(x, x_i) + \frac{1}{n^2} \sum_{i, j=1}^n k(x_i, x_j)}\end{aligned}$$

Uni-dimensional illustration

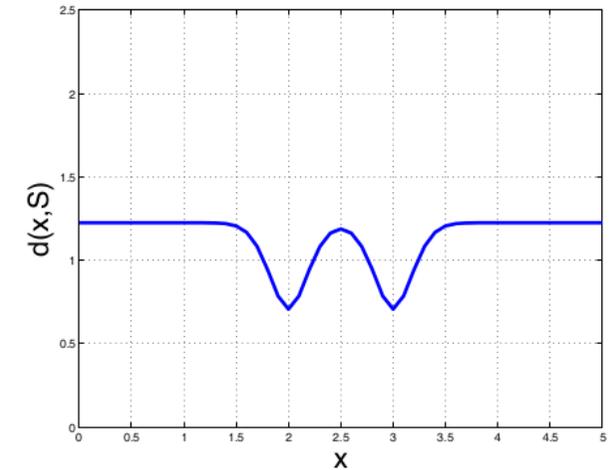
- Let $S = \{2,3\}$, plot $f(x) = d(x,S)$.



Linear kernel



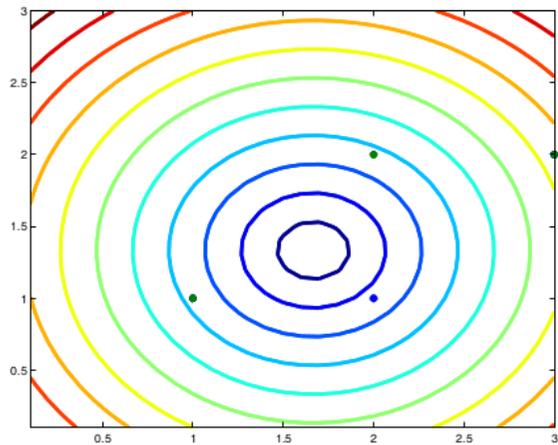
Gaussian kernel,
with $\sigma = 1$



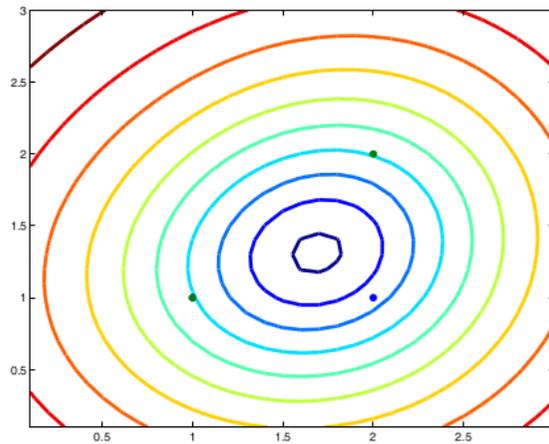
Gaussian kernel,
with $\sigma = 0.2$

2D illustration

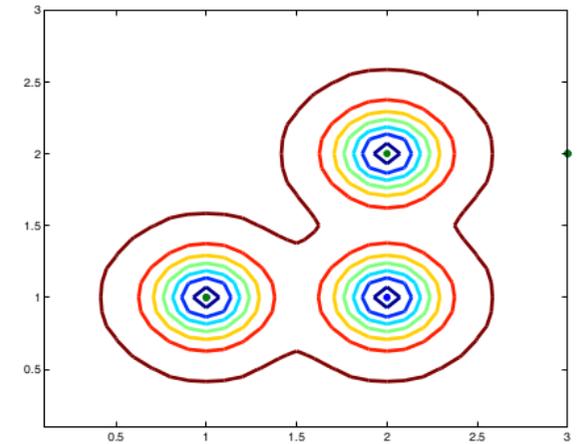
- Let $S = \{ (1,1)', (1,2)', (2,2)' \}$, plot $f(x) = d(x,S)$.



Linear kernel



Gaussian kernel,
with $\sigma=1$



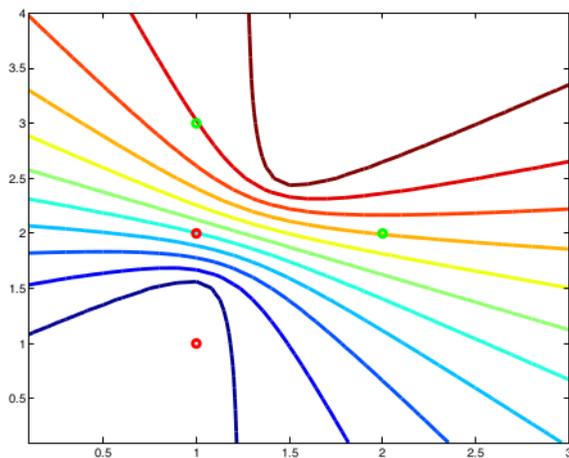
Gaussian kernel,
with $\sigma=0.2$

Application to discrimination

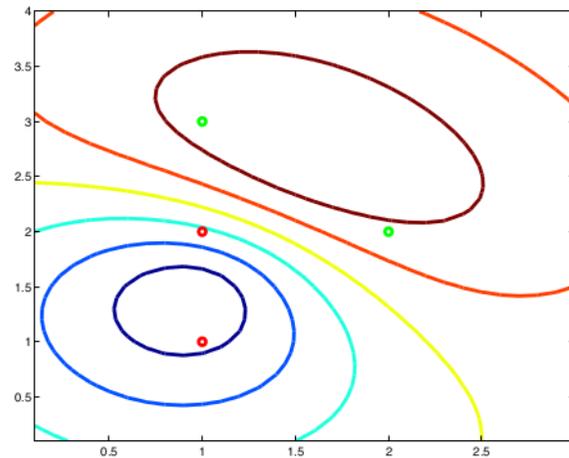
- Consider a set of points from positive class $P = \{ (1,1)', (1,2)' \}$
- And a set of points from the negative class $N = \{ (1,3)', (2,2)' \}$
- Plot $f(x) = d_k(x, P)^2 - d_k(x, N)^2$

$$= \|\varphi(x) - m_P\|_H^2 - \|\varphi(x) - m_N\|_H^2$$

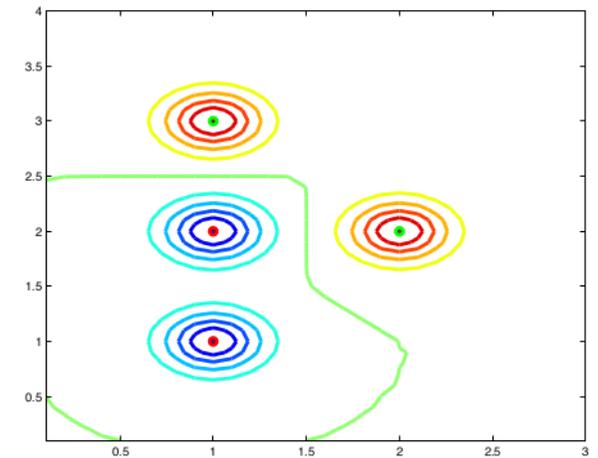
$$= \frac{2}{n} \sum_{x_i \in N} k(x, x_i) - \frac{2}{n} \sum_{x_i \in P} k(x, x_i) + \text{constant}$$



Linear kernel



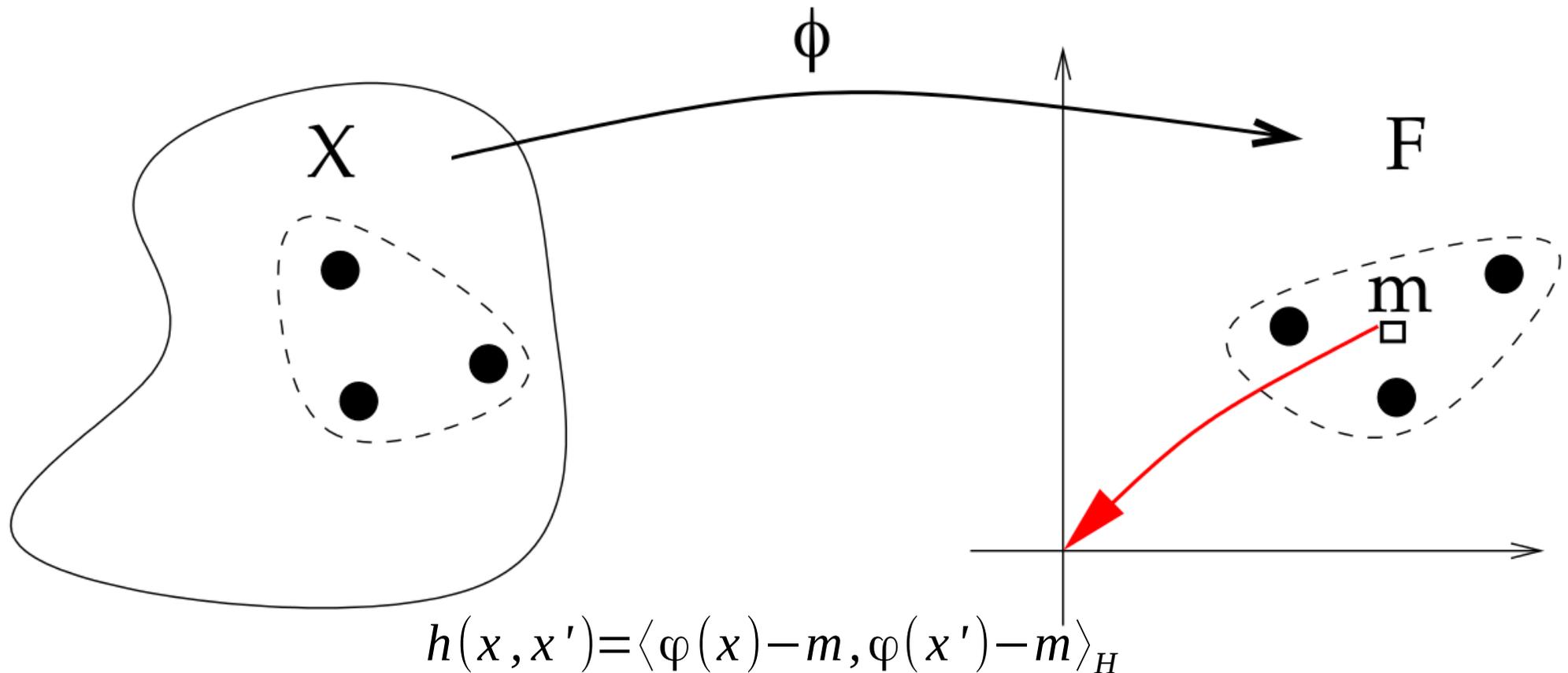
Gaussian kernel,
with $\sigma=1$



Gaussian kernel,
with $\sigma=0.2$

Example 3: centering data in feature space

- Let S be a set of n points in X .
- Let K be the kernel matrix generated by the p.d. kernel $k(\cdot, \cdot)$.
- Let m be the barycenter in the feature space of the points in S .
- How to compute the kernel matrix when the points are centered on m ?



Example 3: centering data in feature space

- Substitution of the barycenter gives

$$\begin{aligned}h(x_i, x_j) &= \langle \varphi(x_i) - m, \varphi(x_j) - m \rangle_H \\ &= \langle \varphi(x_i), \varphi(x_j) \rangle_H - \langle m, \varphi(x_i) + \varphi(x_j) \rangle_H + \langle m, m \rangle_H \\ &= k(x_i, x_j) - \frac{1}{n} \sum_{k=1}^n (k(x_i, x_k) + k(x_k, x_j)) + \frac{1}{n^2} \sum_{k,l=1}^n k(x_k, x_l)\end{aligned}$$

- Or, in matrix notation we get

$$H = K - KU - UK + UKU = (I - U)K(I - U)$$

where for all i, j :

$$U_{i,j} = 1/n$$

