Kernel Methods for Statistical Learning

Jakob Verbeek

jakob.verbeek@inria.fr

October 14, 2014

http://lear.inrialpes.fr/people/mairal/teaching/2014-2015/MSIAM

Summary of previous lecture

- We saw how the risk could generally be decomposed as a term of bias/approximation and a term of variance/estimation.

- This decomposition highlights the trade-off that needs to be dealt with in inference. This trade-off is related to the complexity of the set of functions under consideration
  - Sets too simple lead to a large approximation error.
  - Sets too large lead to a large estimation error.

- We defined this notion of complexity more precisely, using Rademacher complexity and VC dimension, and saw it also depended on the number of samples.

- These notions are crucial in modern applications, where we sometimes have few samples in high dimensions.

Plan for this lecture

- With the notion of bias-variance decomposition in mind we now turn to concrete examples of statistical learning methods.
- Focus on penalized empirical risk minimization techniques, which exactly implement the bias-variance trade-off.

- We focus on linear classification models for supervised learning, i.e., inference using labeled data (label in the form of a class).

- If no labeled data is available but we want to estimate and assumed latent structure, we need unsupervised learning techniques (e.g., dimension reduction or clustering).
  - The same notion of bias-variance decomposition also applies in the unsupervised case (we're still estimating models from data).

- Once we have these techniques in place, we will consider kernels as a way to obtain non-linear models.

- First: a brief recap of constrained optimization techniques.

Intermezzo: constrained optimization basics

- We consider equality and inequality constrained optimization over x of a function f(x)

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & h_i(x)=0, & \text{for } i=1,\ldots,m, \\ \text{and} & g_j(x)\leq 0, & \text{for } i=1,\ldots,r, \end{array}$$

- No assumptions on the form of *f, g*, and *h.*

- We will show that the constrained and penalized forms are often equivalent in some sense.

- Let the constrained solution be given by f*, and thus f*=f(x*) for the global constrained minimizer x*.

Lagrangian and dual function

- The Lagrangian of the optimization problem is given by

$$L:\ X\times R^m\times R^r \rightarrow R$$

$$L(x,\lambda,\mu)=f(x)+\sum_{i=1}^{m}\lambda_i h_i(x)+\sum_{j=1}^{r}\mu_j g_j(x)$$

- Lambda and mu known as Lagrange multipliers, or dual variables.

- The Lagrangian dual function is given by

$$q:\ R^m\times R^r \rightarrow R$$

$$\begin{aligned} q(\lambda,\mu) &= inf_x\ L(x,\lambda,\mu) \\ &= inf_x\ \left(f(x)+\sum_{i=1}^{m}\lambda_i h_i(x)+\sum_{j=1}^{r}\mu_j g_j(x)\right) \end{aligned}$$

Properties of the dual function

- The Lagrange dual function q is concave.
  - Even in the original problem is not convex.

$$q: \ R^m \times R^r \rightarrow R$$

$$\begin{aligned} q(\lambda,\mu) &= inf_x \ L(x,\lambda,\mu) \\ &= inf_x \ \left(f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{j=1}^{r} \mu_j g_j(x) \right) \end{aligned}$$

- Proof:
  - For each x the function $(\lambda,\mu) \rightarrow L(x,\lambda,\mu)$ is linear.
  - The pointwise minimum of concave functions is concave, therefore q is concave.

Properties of the dual function

- The dual function yields lower bounds on the optimal value f* of the original problem if *μ* is nonnegative:

$$q(\lambda,\mu) \leq f^*$$
$$\text{for } \mu \geq 0$$

- Let x* be any feasible point, i.e. h(x*)=0 and g(x*)<= 0.
- Then we have for any lambda and non-negative mu:

$$\sum_{i=1}^{m} \lambda_i h_i(x^*) + \sum_{j=1}^{r} \mu_j g_j(x^*) \leq 0$$

$$L(x^*,\lambda,\mu) = f(x^*) + \sum_{i=1}^{m} \lambda_i h_i(x^*) + \sum_{j=1}^{r} \mu_j g_j(x^*) \leq f(x^*)$$

$$q(\lambda,\mu) = inf_x \; f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{j=1}^{r} \mu_j g_j(x) \leq f(x^*)$$

informatics mathematics

Relation primal and dual problem

- For the primal problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & h_i(x)=0, \quad \text{for } i=1,\ldots,m, \\ \text{and} & g_j(x)\leq 0, \quad \text{for } i=1,\ldots,r, \end{array}$$

- The Lagrange dual problem is:

$$\begin{array}{ll} \text{maximize} & q(\lambda,\mu) \\ \text{subject to} & \mu\geq 0 \end{array}$$

where q is the concave Lagrange dual function and lambda and mu are the Lagrange multipliers associated with the (in)equality constraints.

Weak duality

- Let d* be the optimal value of the Lagrange dual problem.
- Each q(λ,μ) is a lower bound of the optimal value of the primal problem.
- By definition d* is the best lower bound that can be obtained.

- Therefore, the following **weak duality always holds**:

$$d^* \leq f^*$$

- This inequality holds when d* or f* are infinite.

- The difference d*-f* is called the **optimal duality gap** of the original problem.

Strong duality

- Strong duality holds if the optimal duality gap is zero, i.e. d*=f*.
- If strong duality holds, then the best lower bound that can be obtained from the Lagrange dual function is tight.
- Strong duality does not hold of general non-linear problems.
- Strong duality usually holds for convex problems.
- Conditions that ensure strong duality for convex problems are called constraint qualification.

Slater's constraint qualification

- **Strong duality holds for a convex problem** (both f and the g's are convex)

$$
\begin{array}{ll}
\text{minimize} & f(x) \\
\text{subject to} & Ax=b, \\
\text{and} & g_j(x)\leq 0, \quad \text{for } i=1,\ldots,r,
\end{array}
$$

if it is strictly feasible,i.e. there exists at least one feasible point that satisfies the constraints.

Dual optimal pairs

- Suppose that
  - strong duality holds,
  - x^* is primal optimal,
  - (λ^*,μ^*) is dual optimal

  then we have

$$
\begin{aligned}
f(x^*) &= q(\lambda^*,\mu^*) \\
&= inf_x \left\{ f(x) + \sum_{i=1}^{m} \lambda_i^* h_i(x) + \sum_{j=1}^{r} \mu_j^* g_j(x) \right\} \\
&\leq f(x^*) + \sum_{i=1}^{m} \lambda_i^* h_i(x^*) + \sum_{j=1}^{r} \mu_j^* g_j(x^*) \\
&\leq f(x^*)
\end{aligned}
$$

- Therefore, both inequalities are in fact equalities.

Complementary slackness

- The second equality

$$f(x^*)+\sum_{i=1}^{m}\lambda_i^* h_i(x^*)+\sum_{j=1}^{r}\mu_j^* g_j(x^*)=f(x^*)$$

shows that for all j:

$$\mu_j^* g_j(x)=0$$

- This property is called **complementary slackness:** either the i-th optimal Lagrange multiplier is zero or the i-th constraint is active at the optimum.

Reminder: Structural Risk Minimization

1) Define nested function sets of increasing complexity.
2) Minimize the empirical risk over each family.
3) Choose the solution giving the best generalization guarantees.

- Define a complexity measure over functions, and consider the classes $H_1 \subseteq H_2 \subseteq \ldots,$

  where $H_j = \{f : \Omega(f) \leq \mu_j\},$ and $\mu_1 < \mu_2 < \ldots$

- Then in step 2 we solve $min_{f \in H_j} \sum_{i=1}^{n} L(y_i, f(x_i)),$

- We minimize the empirical risk while restricting ourselves to sets of functions of increasing complexity.

- This results in constrained optimization problems. Solving these problems for different loss functions and function spaces is an active topic of research.

informatics mathematics

Equivalence with a penalized estimator

- We will mostly discuss penalized estimators

$$min_{f \in H} \sum_{i=1}^{n} L\left(y_i, f\left(x_i\right)\right) + \lambda \Omega\left(f\right)$$

- The first term favors a good fit to the data, the second one favors regularity of *f*.
- We will show that the constrained and penalized forms are often equivalent in some sense.
- The approach will stay the same: we define a regularization function Ω which is relevant for our problem and we compare the generalization performances of the functions obtained for decreasing values of λ.

Equivalence with a penalized estimator

- In some cases, the constrained problem

$$min_{\Omega(f)\leq\mu} \sum_{i=1}^{n} L\left(y_i, f\left(x_i\right)\right),$$

  is equivalent in some sense to the penalized problem

$$min_{f\in H} \sum_{i=1}^{n} L\left(y_i, f\left(x_i\right)\right) + \lambda\Omega\left(f\right)$$

- Any solution of the constrained problem is a solution of the penalized problem, depending on μ and λ .
  - The latter problem is sometimes easier to solve in practice.
  - The estimator obtained from the latter problem sometimes corresponds to a maximum posterior likelihood problem.

informatics mathematics

Equivalence with a penalized estimator

- Consider the case with
  - L convex
  - Ω convex
  - Assume there exists an f with $\Omega(f) < \mu$

- Let us define

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

$$f_\lambda \in arg\,min_f \;\; L(f) + \lambda \Omega(f)$$

$$f_\mu \in arg\,min_{\Omega(f) \le \mu} \;\; L(f)$$

Equivalence with a penalized estimator

- We first show that the solution of the penalized problem

$$f_\lambda = arg\,min_f \;\; L(f) + \lambda\Omega(f)$$

corresponds to a solution of the constrained problem for some mu.

- Let us constrain the maximum complexity to $\mu = \Omega(f_\lambda)$
  - Clearly the constraint is satisfied for f_λ

- Suppose there exists another function f' with

$$L(f') < L(f_\lambda)$$
$$\Omega(f') \leq \mu$$

then $$L(f') + \lambda\Omega(f') < L(f_\lambda) + \lambda\Omega(f_\lambda)$$

which contradicts the optimality of f_λ for the penalized problem.

- Note that we did not rely on convexity here, result is general.

Equivalence with a penalized estimator

- We now show that the solution of the constrained problem

$$f_\mu = \arg\min_{\Omega(f)\leq\mu} \; L(f)$$

corresponds to a solution of the penalized problmem.

- Let us define the Lagrangian of the constrained problem as

$$L(f,\lambda) = L(f) + \lambda\left(\Omega(f) - \mu\right)$$

- The dual of the constrained problem is $q(\lambda) = \min_f \; L(f,\lambda)$

- Note that $q(\lambda) = \min_f \; L(f,\lambda) = L(f_\lambda, \lambda)$

- By strong duality we have

$$\min_{\Omega(f)\leq\mu} L(f) = \max_{\lambda\geq 0} \; \min_f \; L(f,\lambda) = \max_{\lambda\geq 0} \left(\; L(f_\lambda) + \lambda\left(\Omega(f_\lambda) - \mu\right) \; \right)$$

Equivalence with a penalized estimator

$$min_{\Omega(f)\leq\mu} L(f) = max_{\lambda\geq 0}\ min_f\ L(f,\lambda) = max_{\lambda\geq 0}\left(\ L(f_\lambda) + \lambda(\Omega(f_\lambda) - \mu)\ \right)$$

- In addition, by Slater's conditions again, there exists λ* such that

$$L(f_\mu) = min_{\Omega(f)\leq\mu} L(f) = L(f_{\lambda^*}) + \lambda^*(\Omega(f_{\lambda^*}) - \mu)$$

- By complementary slackness, it is necessary that $\lambda^*\left(\Omega(f_{\lambda^*}) - \mu\right) = 0$

  which implies that $L(f_\mu) = L(f_{\lambda^*})$ and

  - Either λ*=0 and therefore the constrained problem gives the solution to the zero penalty case: $L(f_\mu) + 0\,\Omega(f_\mu) = L(f_{\lambda^*}) + 0\,\Omega(f_{\lambda^*})$
  - Or $\Omega(f_{\lambda^*}) = \mu$ and therefore the constrained problem gives the solution to the penalized case

$$L(f_\mu) + \lambda^*\Omega(f_\mu) = L(f_{\lambda^*}) + \lambda^*\Omega(f_\mu) \leq L(f_{\lambda^*}) + \lambda^*\Omega(f_{\lambda^*})$$

informatics mathematics

Equivalence with a penalized estimator

- In some cases, the constrained problem

$$min_{\Omega(f)\leq\mu} \sum_{i=1}^{n} L(y_i, f(x_i)),$$

  is equivalent in some sense to the penalized problem

$$min_{f\in H} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda\Omega(f)$$

- Any solution of the constrained problem is a solution of the penalized problem, depending on μ and λ .
  - The latter problem is sometimes easier to solve in practice.
  - The estimator obtained from the latter problem sometimes corresponds to a maximum posterior likelihood problem.

An example: the L2 penalty for a linear model

- Let us consider a linear model $f_\theta(x) = \theta^T x, \qquad x \in R^p$
- The penalty function $\Omega(f_\theta) = \|\theta\|_2^2$
- One of the most common penalty functions
  - In support vector machines for classification.
  - In ridge regression.
- Leads to functions with the following type of regularity:
  - Two points that are close in terms of the Euclidean norm have similar function evaluations.
  - Direct consequence of the Cauchy-Schwarz inequality:

$$|f(x) - f(x')| = |\theta^T x - \theta^T x'| = |\theta^T (x - x')| \leq \|\theta\|_2 \|x - x'\|_2$$

informatics mathematics Inria

An example: the L2 penalty for a linear model

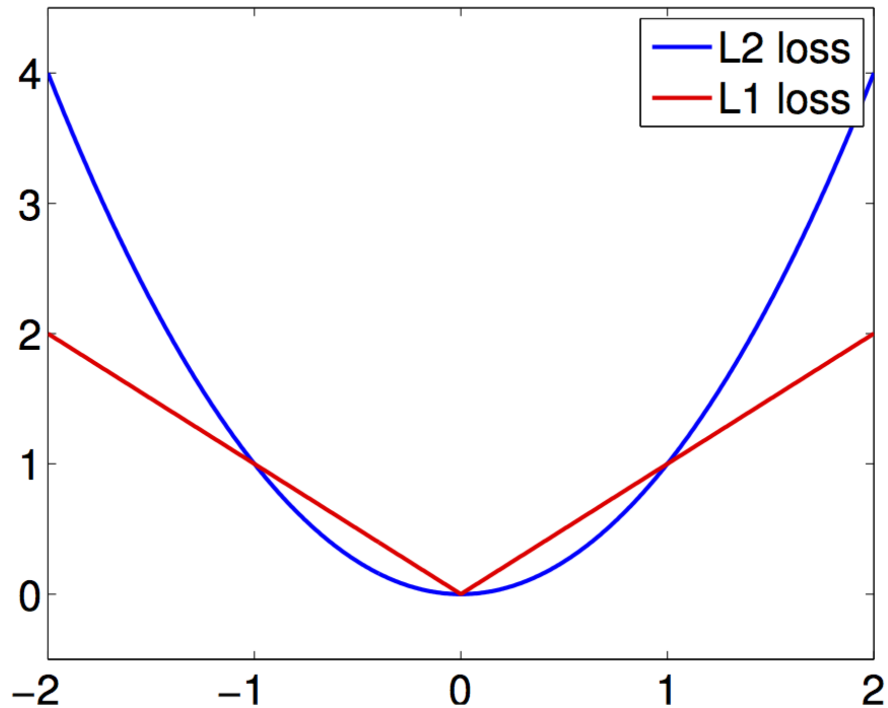
- Let us consider a linear model $f_\theta(x)=\theta^T x, \qquad x \in R^p$
- The penalty function $\Omega(f_\theta)=\|\theta\|_2^2$
- Leads to functions with the following type of regularity:
  - Two points that are close in terms of the Euclidean norm have similar function evaluations.

    $$|f(x)-f(x')| \le \|\theta\|_2 \|x-x'\|_2$$

- This property can limit overfitting, and improve generalization: it makes functions behave similarly over similar, potentially unobserved, data.
- Of course, if there is no good predictor with this kind of regularity, the risk can be high because of the approximation error term.

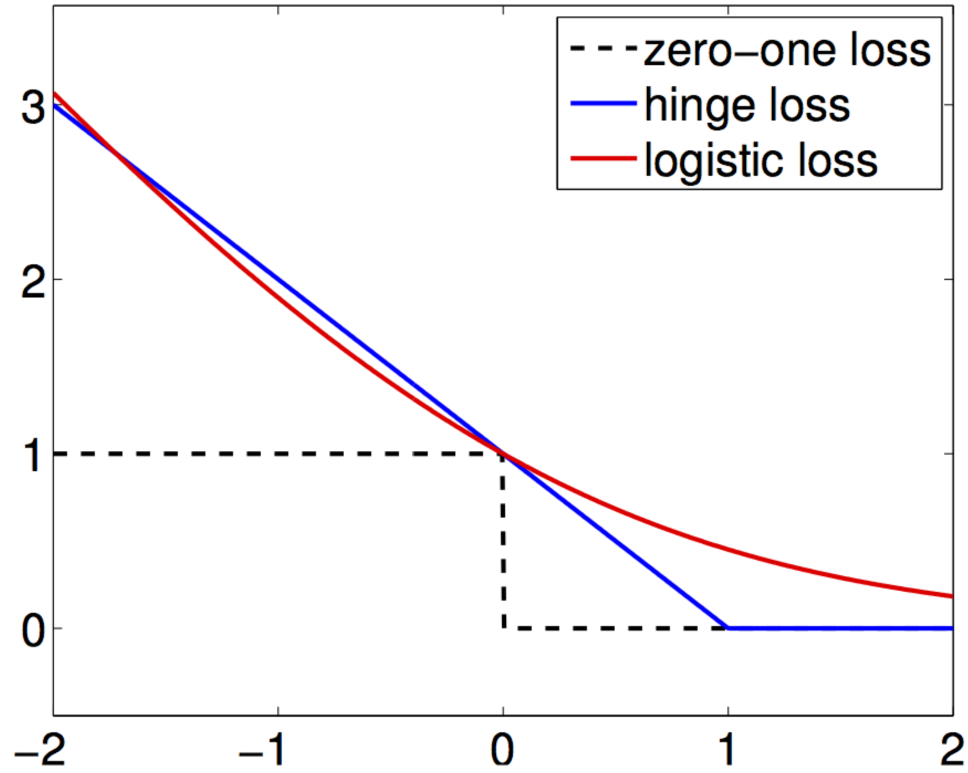
Common loss functions for regression

- L2 loss (considered before): $L(y, f(x)) = (y - f(x))^2$
- L1 loss: $L(y, f(x)) = |y - f(x)|$
  - more robust against large errors
  - Bayes estimator gives median instead of mean



Common loss functions for classification

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



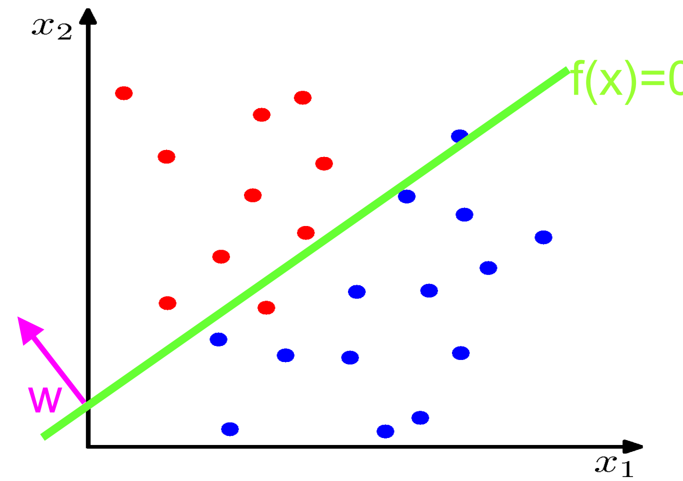
informatics mathematics

Common loss functions for classification

- Assign class label using $y = sign(f(x))$
  - Zero-One loss: $L(y_i, f(x_i)) = [y_i f(x_i) \geq 0]$
  - 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 this leads to convex objective functions, for which global optima can be found.
- Methods based on convex objectives are also simpler to analyze.
- Convexity does, however, not guarantee better performance than non-convex counterparts in practice!

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

Logistic discriminant classifier

- Map linear score function to class probabilities with sigmoid

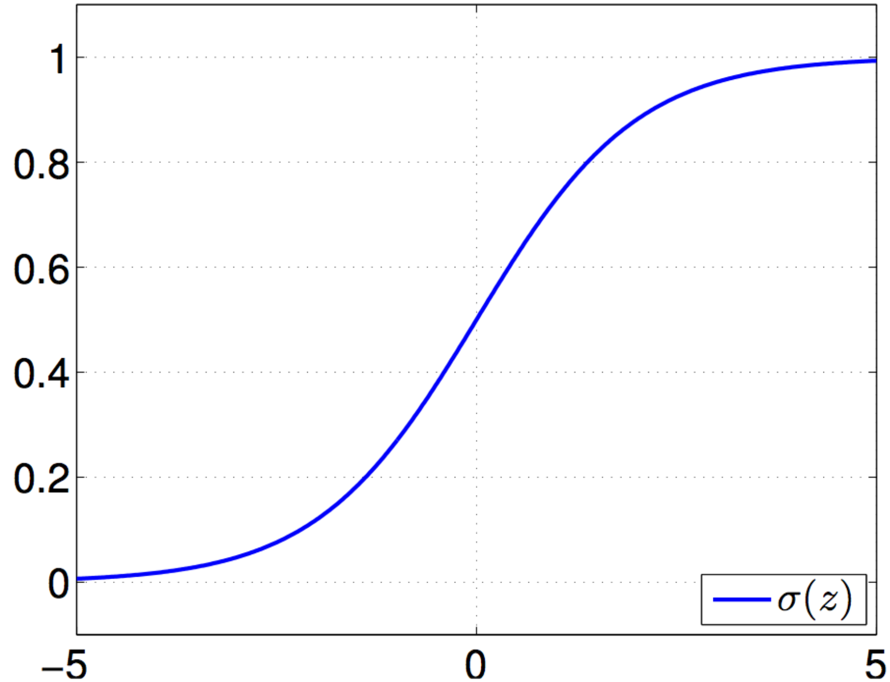
$$p(y=+1|x)=\sigma(w^T x)$$

- For binary classification problem, we have by definition

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

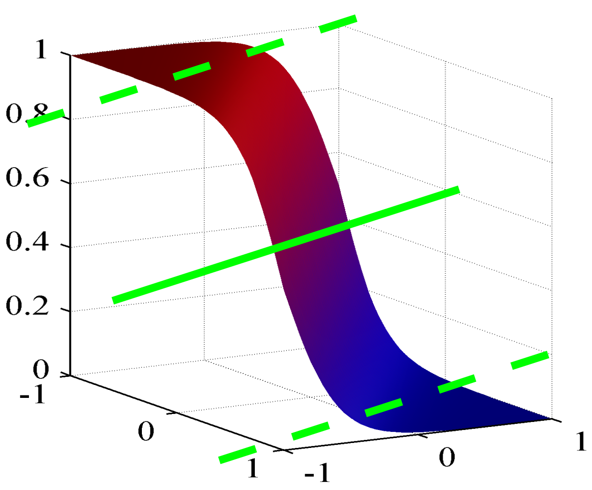
  - Exercise: show that $p(y=-1|x)=\sigma(-w^T x)$

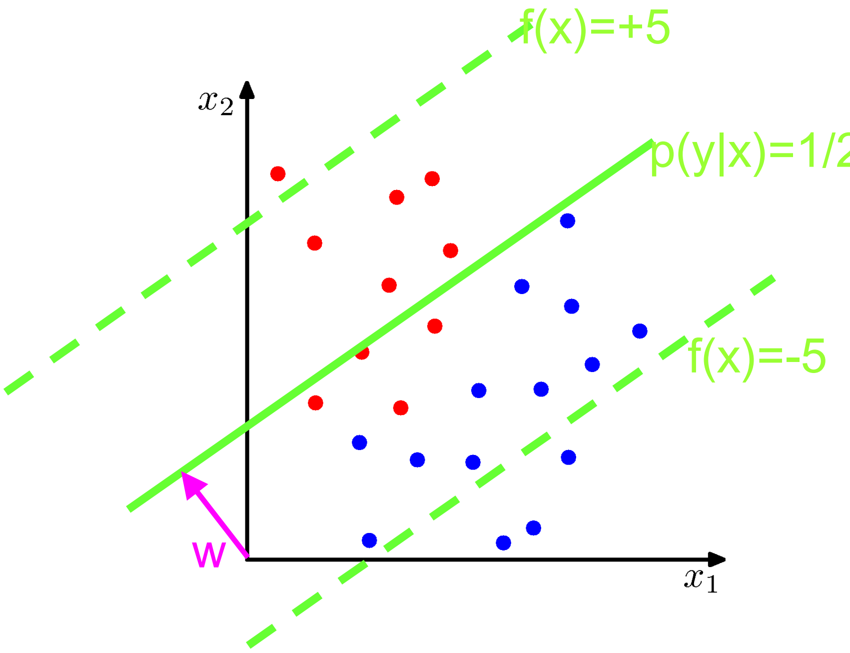
$$\sigma(z)=\frac{1}{1+\exp(-z)}$$



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

$$\log\prod_{i=1}^{n} p(y_i|x_i)=\sum_{i=1}^{n}\log p(y_i|x_i)$$

$$=\sum_{i=1}^{n}\log\sigma(y_i w^T x_i)$$

$$=-\sum_{i=1}^{n}\log\left(1+\exp(-y_i w^T x_i)\right)$$

$$=-L_{\text{logistic}}(y_i, w^T x_i)$$

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

Logistic discriminant estimation

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

$$min_w \sum_{i=1}^{n} L\left(y_i, w^T x_i\right) + \lambda \frac{1}{2} w^T w$$

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

- Exercise 1: derive the gradient $\frac{\partial L\left(y_i, w^T x_i\right)}{\partial w} = -y_i\left(1 - p\left(y_i | x_i\right)\right) x_i$

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

Logistic discriminant estimation

- Solve objective function using first or second order methods

$$min_w \sum_{i=1}^{n} \log\left(1+\exp\left(-y_{iw}^T x_i\right)\right)+\lambda \frac{1}{2} w^T w$$

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

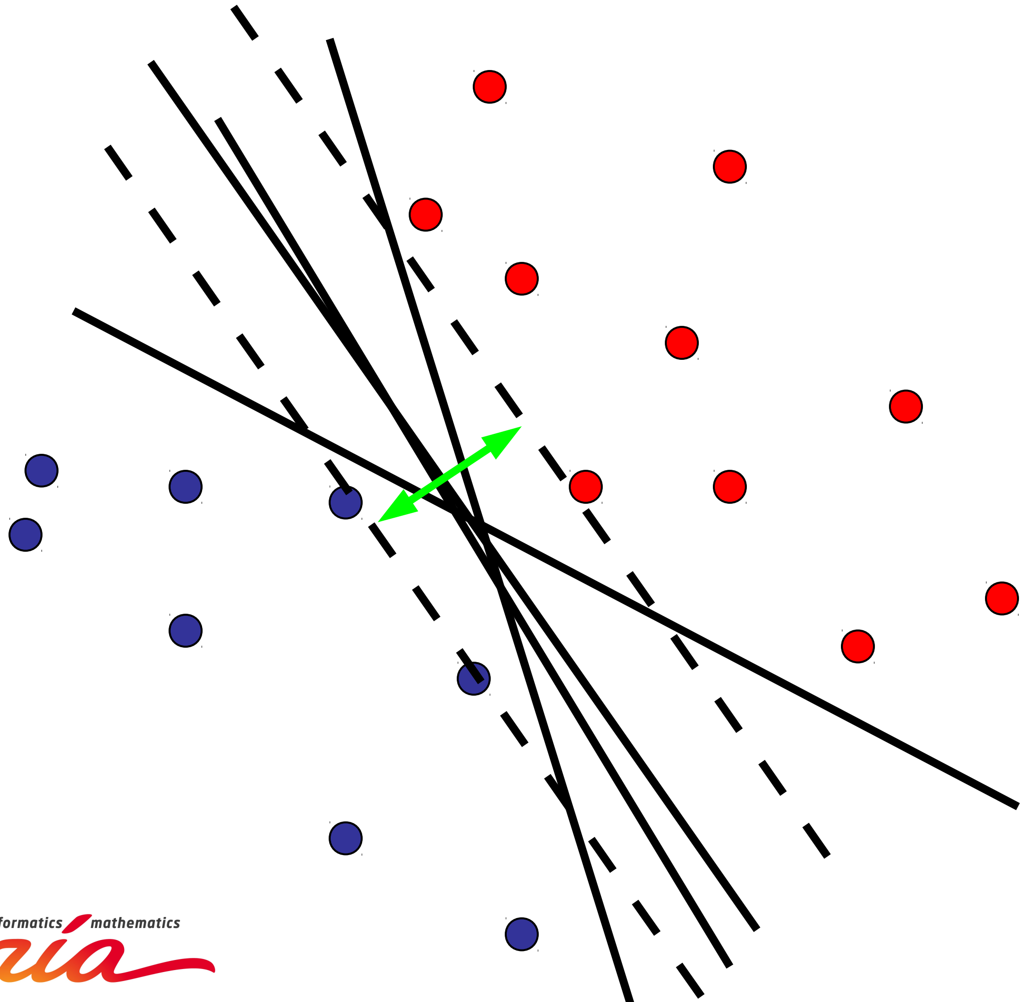
- Recall the gradient $$\frac{\partial L\left(y_i, w^T x_i\right)}{\partial w} = -y_i\left(1-p\left(y_i | x_i\right)\right) x_i$$

- 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



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

Support vector machines

- Witout loss of generality, define function value at the margin as +/- 1
- Now constrain w to that all points fall on correct side of the margin:

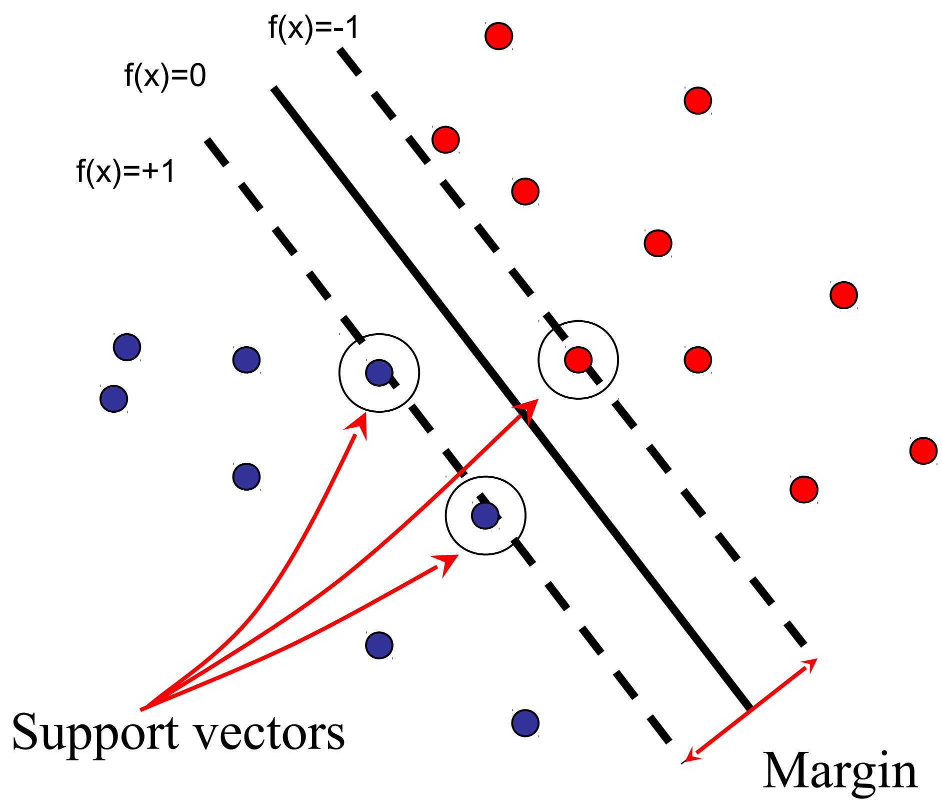
$$y_i\left(w^T x_i + b\right) \geq 1$$

- 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.



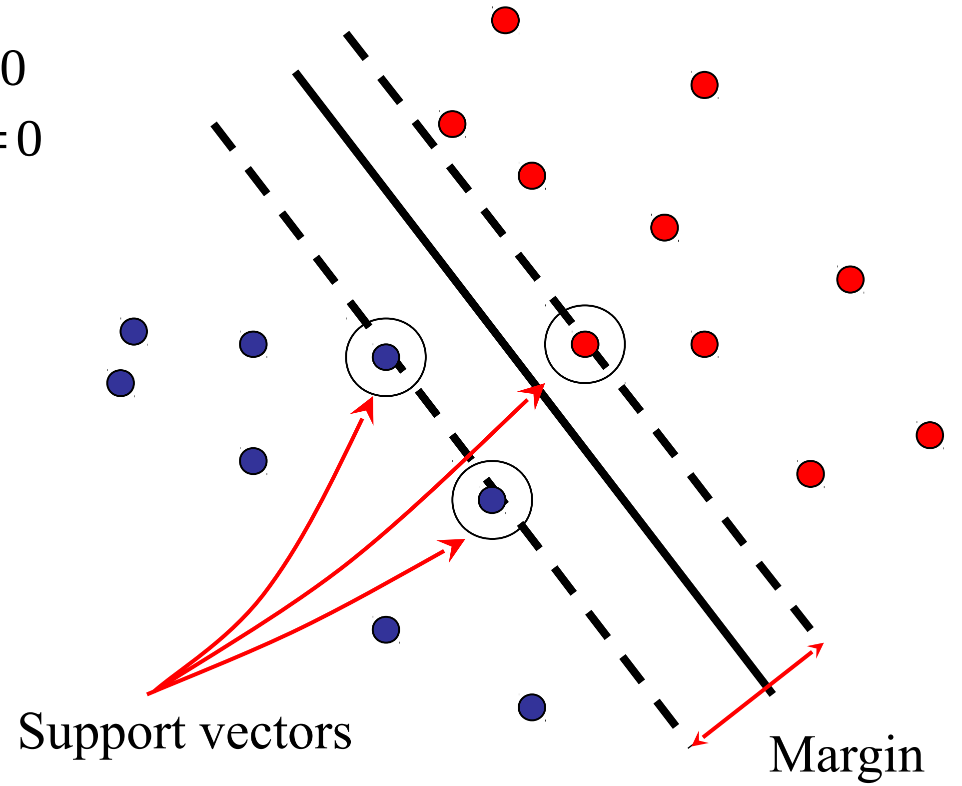
Support vector machines

- 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

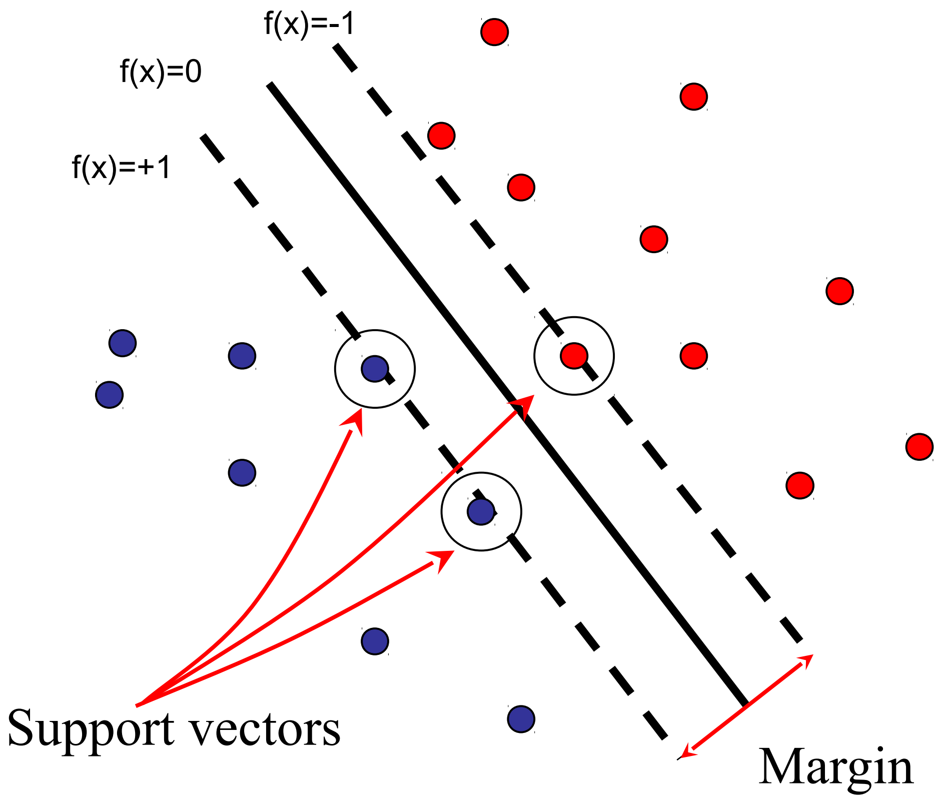
$$\|x - z\|_2 = \|x - (x - \alpha w)\|_2$$
$$\|\alpha w\|_2 = \alpha \|w\|_2$$
$$\frac{\|w\|_2}{\|w\|_2^2} = \frac{1}{\|w\|_2}$$



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

$$argmin_{w,b} \frac{1}{2} w^T w$$
$$\text{subject to } \; y_i(w^T x_i + b) \geq 1$$



Support vector machines: optimization

- The primal version of the optimization problem:

$$argmin_w \frac{1}{2} w^T w$$
$$\text{subject to } y_i \left(w^T x_i + b \right) \geq 1$$

- For each constraint, i.e. for each data point, we introduce a corresponding dual variable alpha, which leads to the Lagrangian:

$$L(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i \left(y_i \left(w^T x_i + b \right) - 1 \right)$$

  - Note sign-swap of constraint terms, since here we have larger-equal, rather than smaller equal as in the general presentation.

Support vector machines: optimization

$$L(w,b,\alpha)=\frac{1}{2}w^T w-\sum_{i=1}^{n}\alpha_i\left(y_i\left(w^T x_i+b\right)-1\right)$$

- The Lagrangian is convex and quadratic in w.
- It is minimized w.r.t. w for:

$$\nabla_w L=w-\sum_{i=1}^{n}\alpha_i y_i x_i=0$$

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

- The Lagrangian is affine in b.
- It has minimum minus infinity, except when:

$$\nabla_b L=\sum_{i=1}^{n}\alpha_i y_i=0$$

informatics mathematics

Support vector machines: optimization

- We therefore obtain the Lagrange dual function:

$$q(\alpha)=\inf_{w,b}\ L(w,b,\alpha)$$

$$=\inf_{w,b}\frac{1}{2}w^T w-\sum_{i=1}^{n}\alpha_i\left(y_i\left(w^T x_i+b\right)-1\right)$$

$$=\inf_{w,b}\frac{1}{2}w^T w-w^T\sum_{i=1}^{n}\alpha_i y_i x_i-b\sum_{i=1}^{n}\alpha_i y_i+\sum_{i=1}^{n}\alpha_i$$

$$=\begin{cases}\text{if } \sum_{i=1}^{n}\alpha_i y_i=0: & \sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}y_i y_j\alpha_i\alpha_j x_i^T x_j\\ \text{otherwise} & : -\infty\end{cases}$$

- The dual problem is:

$$\begin{array}{ll}\text{maximize} & q(\alpha)\\ \text{subject to} & \alpha\geq 0\end{array}$$

Support vector machines: optimization

- The dual problem is therefore equal to

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j$$

subject to

$$\alpha \geq 0, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

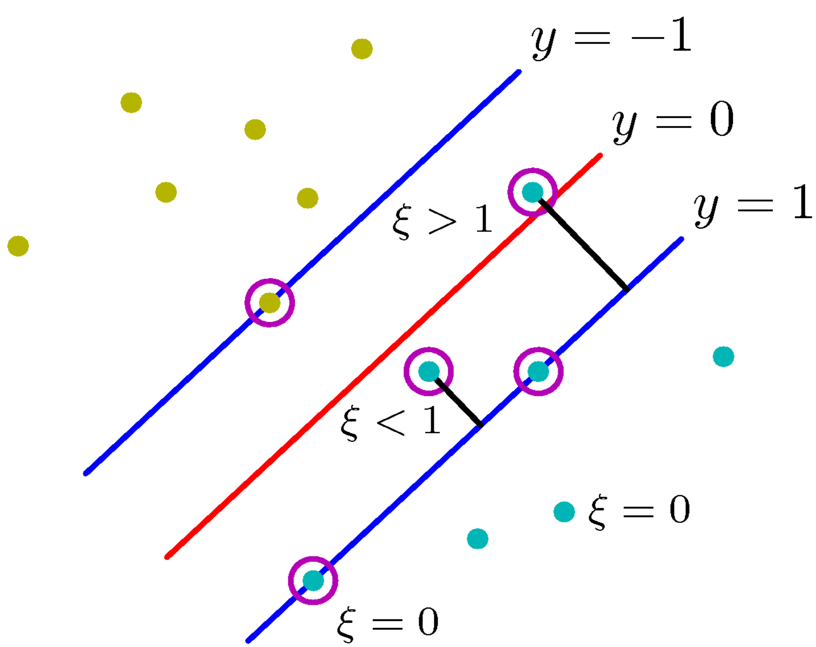
- This is a quadratic program with n variables, with simple linear constraints.
- Note that the data is accessed only in terms of pairwise dot-products.
- Less variables to solve with respect to primal problem, if we have less data points than dimensions.

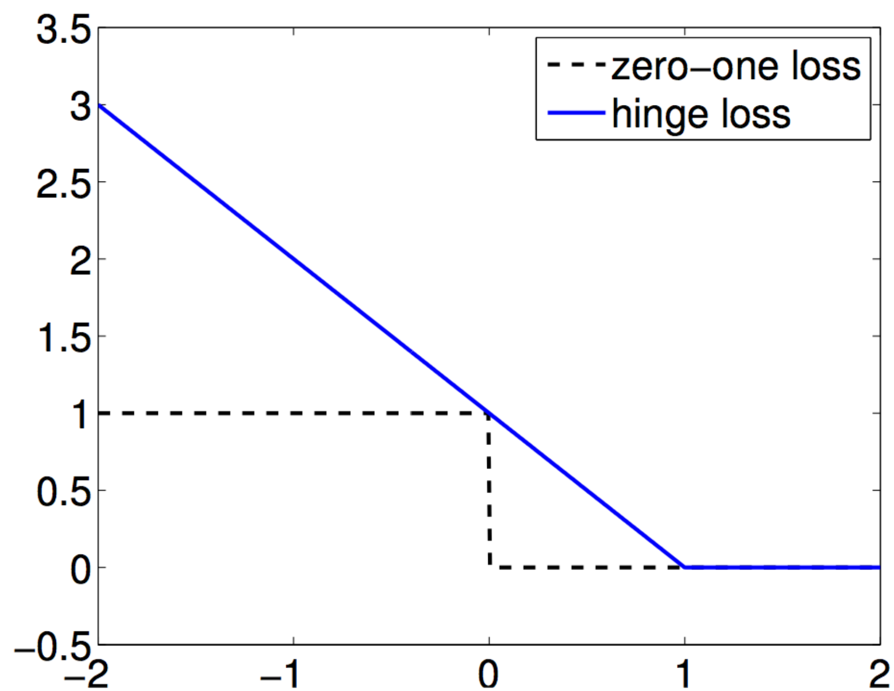
Support vector machines: inseperable classes

- 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

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

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

- Transformation into 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

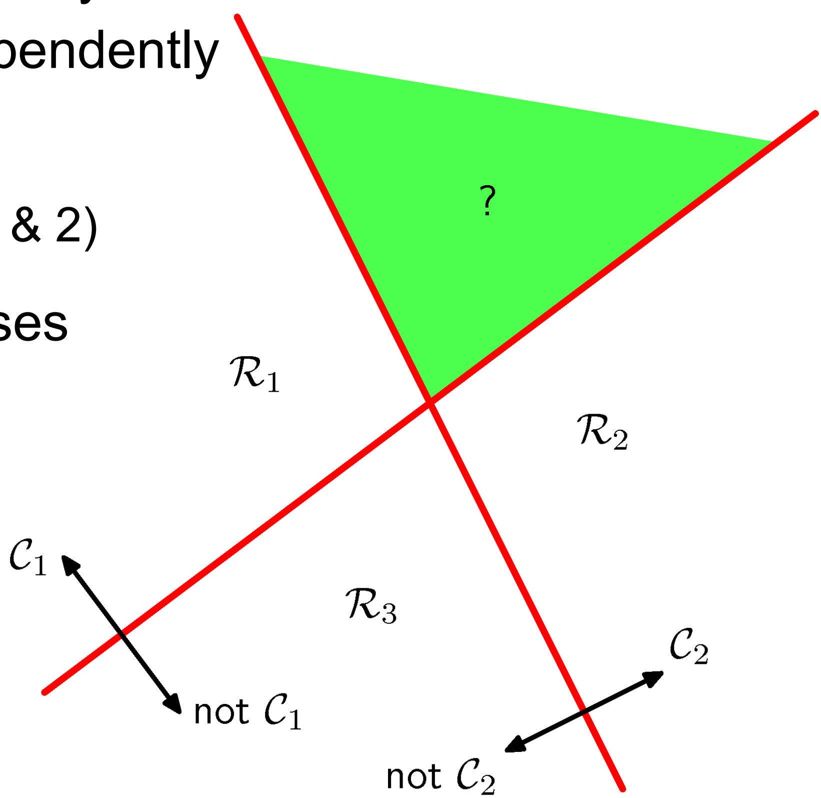
$$\min_{w,b,\{\xi_i\}} \quad \lambda \frac{1}{2} w^T w \; + \; \sum_i \xi_i$$

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

- Solution of the quadratic program has again the property that w is a linear combination of the data points.

Dealing with more than two classes

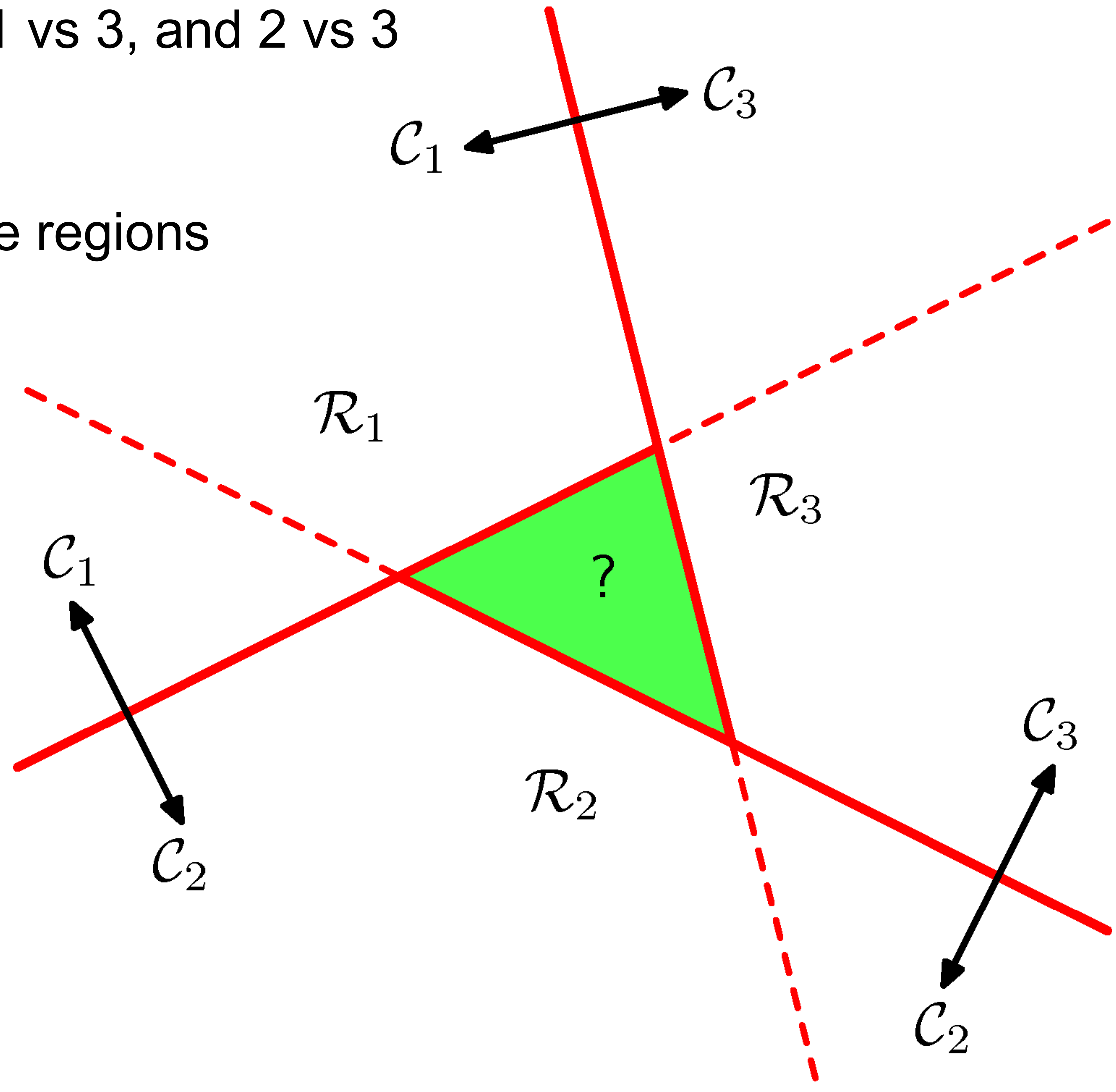
- So far, we have only considered the, useful, case for two classes
  - E.g., is this email spam or not ?
- Many practical problems have more classes
  - E.g., which fruit is placed on the supermarket weight scale: apple, orange, or banana ?
- First idea: construction from multiple binary classifiers
  - 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



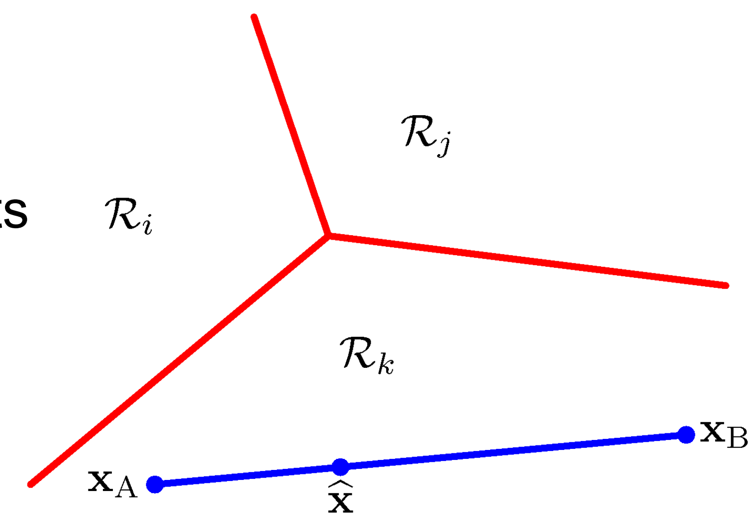
Dealing with more than two classes

- 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 assgined to that class.

informatics mathematics

Multi-class logistic discriminant classifier

- Map score functions to class probabilities with "soft-max"

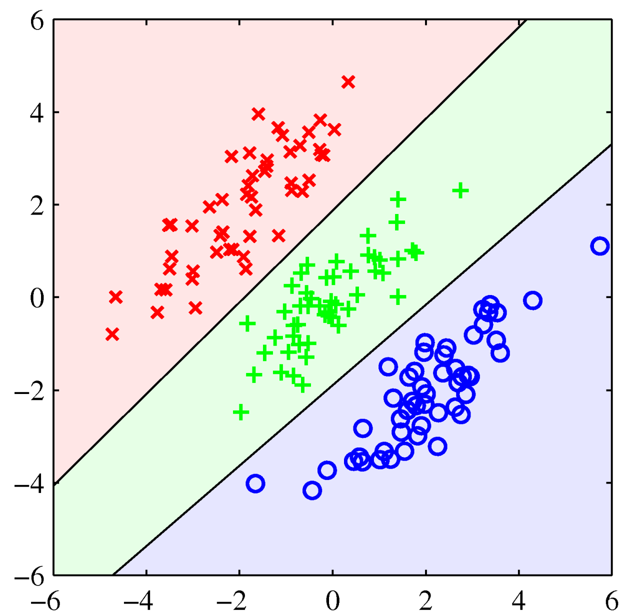
$$f_k(x) = w_k^T x \qquad\qquad p(y=c|x) = \frac{\exp(f_c(x))}{\sum_{k=1}^{K} \exp(f_k(x))}$$

  - 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

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

- 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

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

- This objective function is also convex in the w vectors

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

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

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

- 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

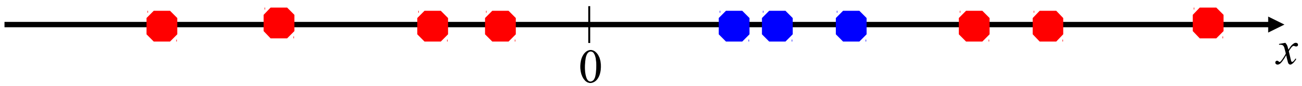
$$w = \sum_{i=1}^{n} \alpha_i x_i$$

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

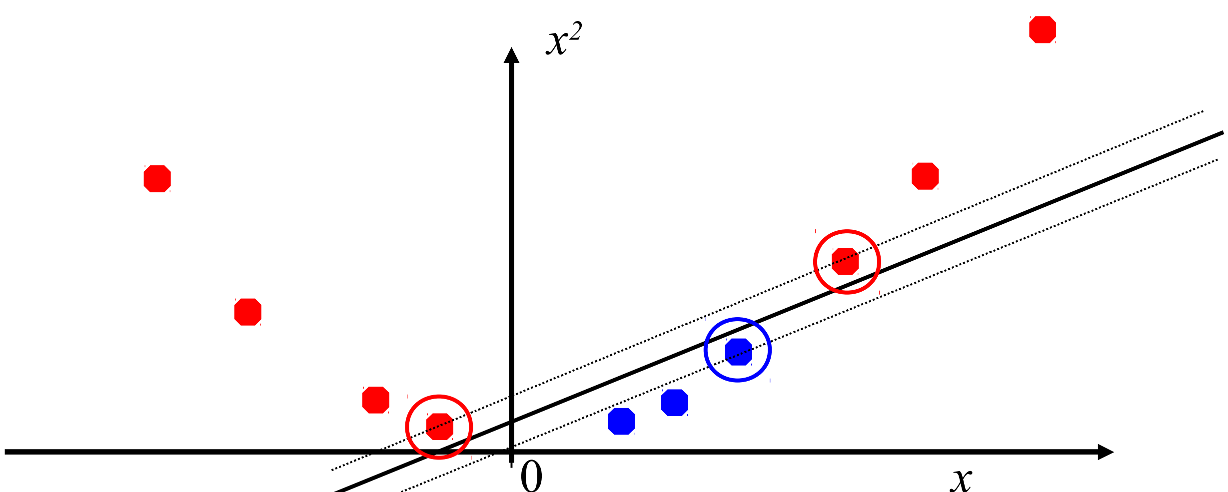
$$f(x) = w^T x + b$$
$$= \sum_{i=1}^{n} \alpha_i \left(x_i^T x \right) + b$$

Nonlinear Classification

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



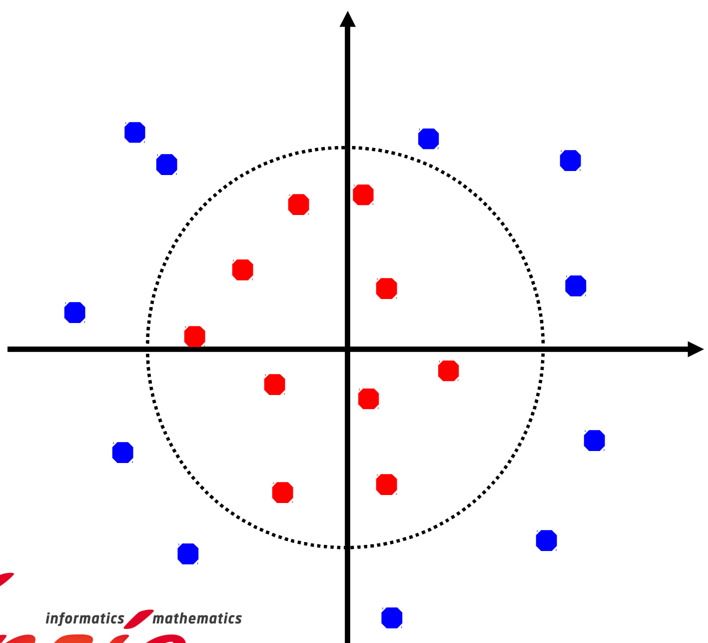
- Similar to what we considered last week for regression with higher-order polynomials, we can do linear classification on non-linear features. For example augment map the data to R^2 by adding x^2.



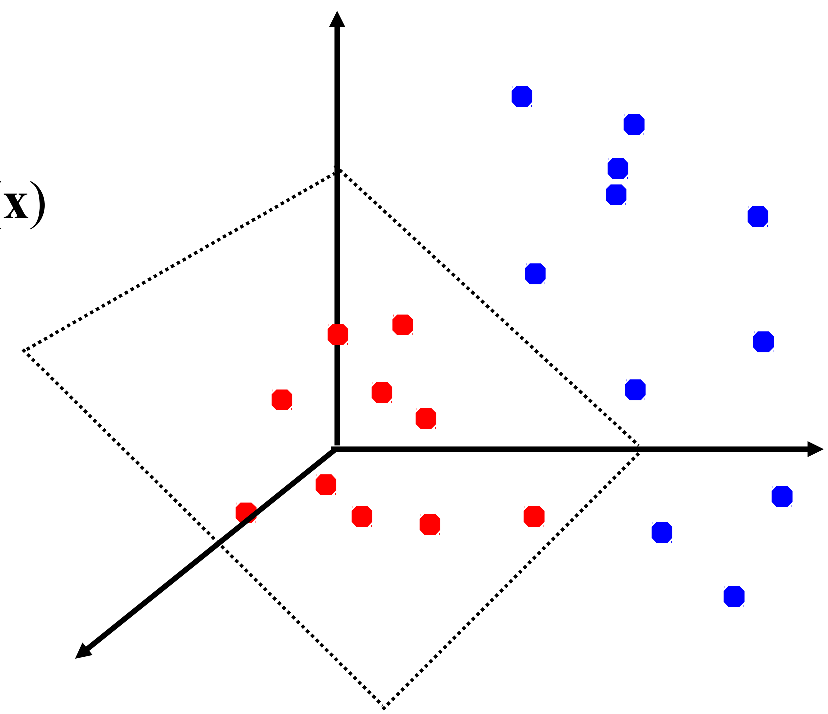
Slide credit: Andrew Moore

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 ?



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



informatics mathematics

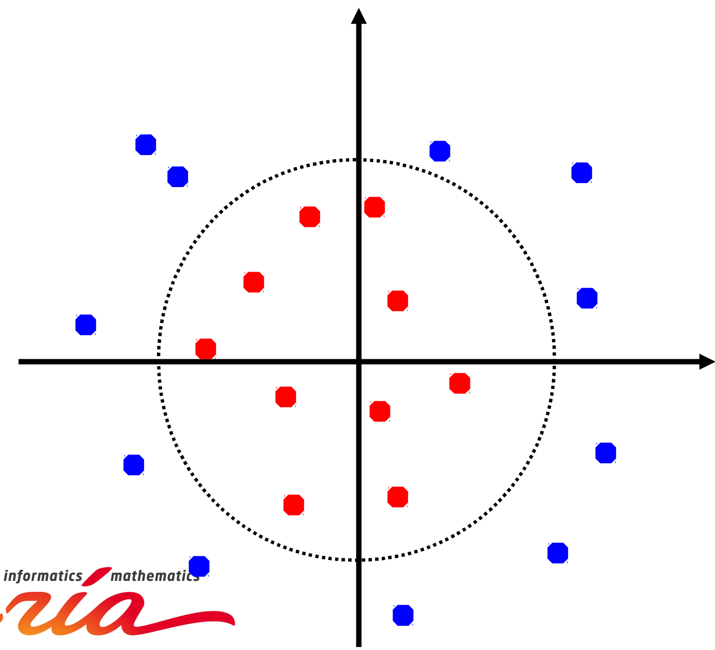
Slide credit: Andrew Moore

Non-linear feature mappings for classification

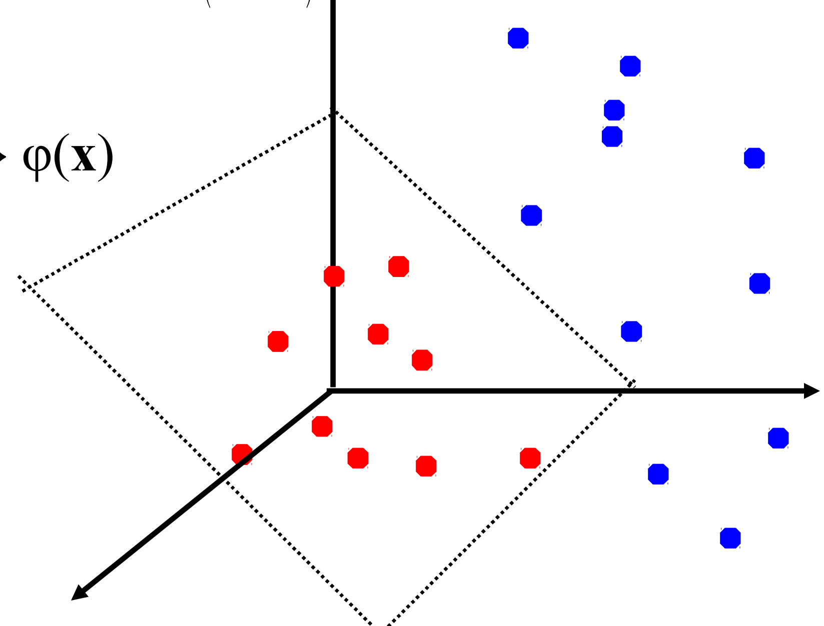
- Remember that for classification we only need dot-products.
- Let's calculate the dot-product explicitly for our example.
  - New dot-product easily computed from the original one.

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

$$\begin{aligned} k(x, z) = \varphi(x)^T \varphi(z) &= ? \\ &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 \\ &= (x_1 z_1 + x_2 z_2)^2 \\ &= (x^T z)^2 \end{aligned}$$



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



Non-linear feature mappings for classification

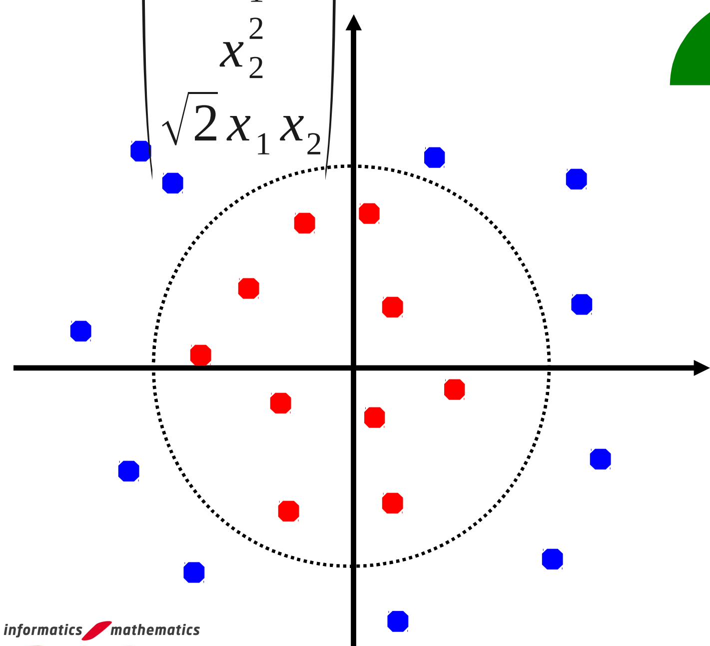
- Suppose we also want to keep the original features to still be able to implement linear functions
  - Again efficient computation in 6d, roughly at cost of 2d dot-product

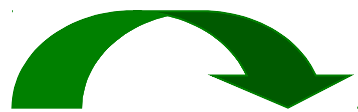
$$\varphi(x) = \begin{pmatrix} 1 \\ \sqrt{2}\, x_1 \\ \sqrt{2}\, x_2 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}\, x_1 x_2 \end{pmatrix}$$

$$k(x, y) = \varphi(x)^T \varphi(y) = ?$$

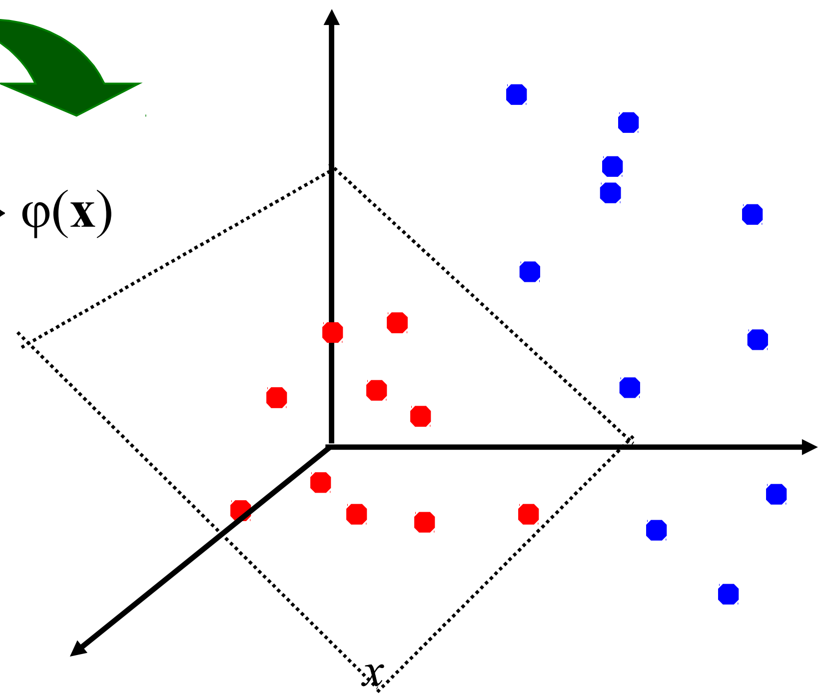
$$= 1 + 2\mathrm{x}^T y + \left(x^T y\right)^2$$

$$= \left(x^T y + 1\right)^2$$





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


0

Slide credit: Andrew Moore

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 ?

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

  - 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 $\left(x^T y\right)^2=\left(x_1 y_1+\ldots+x_D y_D\right)^2$

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

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

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

$$\varphi(x)=\left(1,\underbrace{\sqrt{2}x_1,\sqrt{2}x_2,\ldots,\sqrt{2}x_D}_{\text{Original features}},\underbrace{x_1^2,x_2^2,\ldots,x_D^2}_{\text{Squares}},\underbrace{\sqrt{2}x_1x_2,\ldots,\sqrt{2}x_1x_D,\ldots,\sqrt{2}x_{D-1}x_D}_{\text{Products of two distinct elements}}\right)^T$$

Nonlinear classification with kernels

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

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

- We will see that conversely, if a kernel is positive definite then it computes an inner product in some feature space, possibly with large number or infinite number of dimensions.

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

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

Summary linear 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 support vectors, may cost lots of memory.
  - Computing kernel between new data point and support vectors may be computationally expensive
- Kernel functions can also be used for other linear data analysis
  - Principle component analysis, k-means, CCA, regression, ...

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 x n matrix $[K]_{ij} = K(x_i, x_j)$

- Always an n x 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:

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

  and which satisfies

  $$\forall n \in N$$
  $$\forall (x_{1,} \dots, x_n) \in R^n \quad \text{and} \quad (a_{1,} \dots, a_n) \in R^n$$
  $$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j K(x_i, x_j) \geq 0$$

- 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”

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

- Proof
  - Symmetry: $K(x, x') = x^T x' = (x')^T x = K(x', x)$
  - Positive definiteness:

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

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.

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

- Proof
  - Symmetry: $K(x, x') = \langle \varphi(x), \varphi(x') \rangle_H = \langle \varphi(x'), \varphi(x) \rangle_H = K(x', x)$
  - Positive definiteness:

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

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.

Some definitions

- An **inner product** on an R-vector space H is a mapping

$$H \times H \rightarrow R$$
$$(f, g) \rightarrow \langle f, g \rangle_H$$

  that is bilinear, symmetric, and such that $\langle f, f \rangle_H > 0$ for all $f \in H \setminus 0$

- A vector space endowed with an inner product is called **pre-Hilbert.** It is endowed with a norm defined by the inner product as

$$\|f\|_H = \langle f, f \rangle_H^2$$

- A **Hilbert space** is a pre-Hilbert space complete for the norm defined by the inner product.
  - In other words: any Cauchy series of points in H, has a limit in H.
  - A series $f_{1,} f_{2,} f_{3,} \ldots$ is Cauchy if for any $\epsilon > 0$ there exists some N, such that for any $m, n > N$ we have that $\|f_n - f_m\|_H < \epsilon$

Proof, for the case of finite sets X

- Suppose X is a finite set of size n: $X = \{x_{1,} x_{2,} \dots, x_n\}$

- Any positive definite kernel $K: \ X \times X \rightarrow R$ is entirely defined by the n x n symmetric positive semidefinite matrix $[K]_{ij} = K(x_i, x_j)$

- The kernel matrix can therefore be diagonalized on an orthonormal basis of eigenvectors with non-negative eigenvalues

$$K = U \Lambda U^T$$

  - Eigenvectors are the columns of U.
  - Eigenvalues in the diagonal matrix lambda.

- Therefore $K(x_i, x_j) = [U \Lambda U^T]_{ij} = \sum_{l=1}^{N} \lambda_l u_l(i) u_l(j)$

$$= \langle \varphi(x_i), \varphi(x_j) \rangle$$

with $\varphi(x_i) = \left(\sqrt{\lambda_1} u_1(i), \sqrt{\lambda_2} u_2(i) \dots, \sqrt{\lambda_n} u_n(i) \right)^T$

informatics mathematics Inria