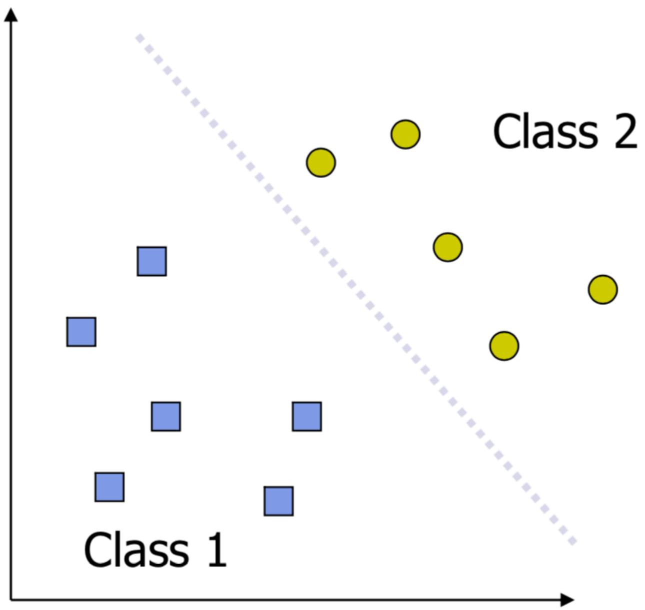
Kernel machines: SVM and duality

Yifeng Tao
School of Computer Science
Carnegie Mellon University

Slides adapted from Eric Xing and Ryan Tibshirani

What is a good decision boundary?

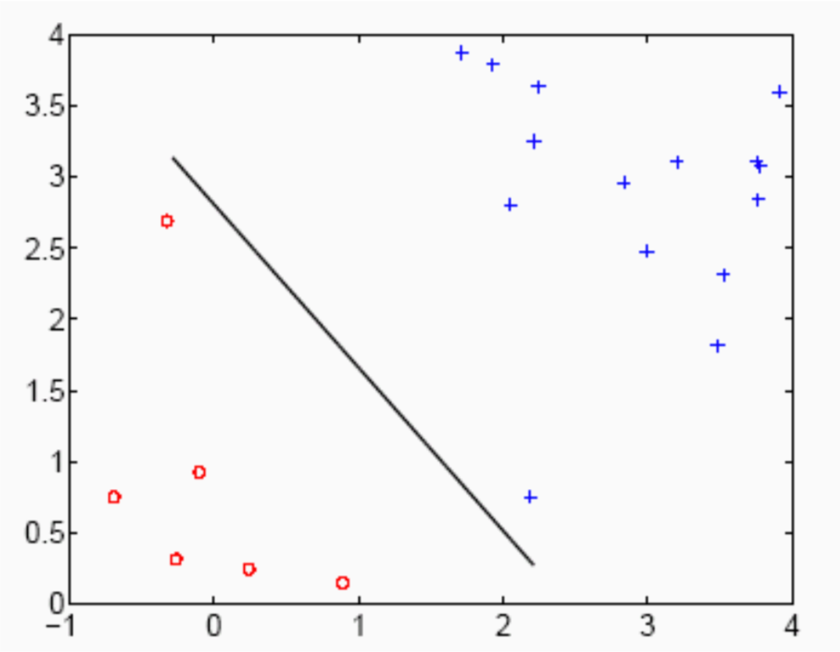
- Consider a binary classification task with y=±1 labels (not 0/1 as before).
- When the training examples are linearly separable, we can set the parameters of a linear classifier so that all the training examples are classified correctly
- Many decision boundaries!
  - Generative classifiers
  - Logistic regressions ...
- Are all decision boundaries equally good?

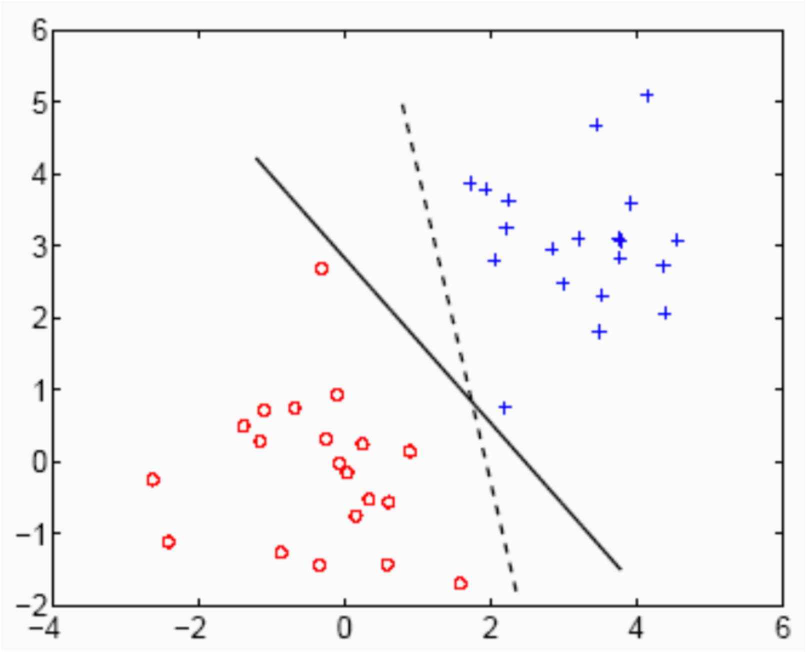


[Slide from Eric Xing]

Examples of Bad Decision Boundaries

- Why we may have such boundaries?
  - Irregular distribution
  - Imbalanced training sizes
  - outliners





[Slide from Eric Xing]

Classification and Margin

- Parameterizing decision boundary
  - Let *w* denote a vector orthogonal to the decision boundary, and *b* denote a scalar "offset" term, then we can write the decision boundary as:

$$w^T x + b = 0$$

- Margin

$w^T x + b > 0$ for all x in class 2

$w^T x + b < 0$ for all x in class 1

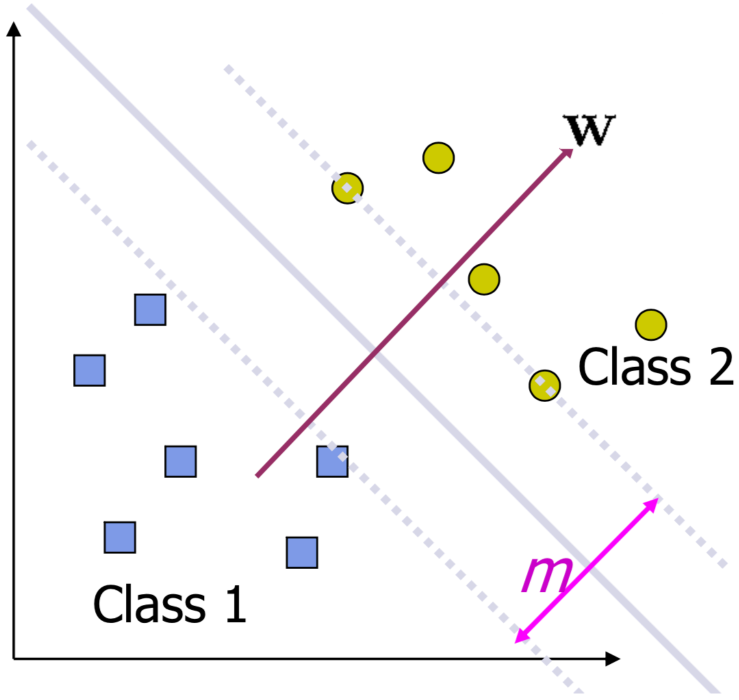
Or more compactly:

$$(w^T x_i + b) y_i > 0$$

The margin between two points

$$\boldsymbol{m} = (w^T x_i + b) - (w^T x_j + b) = w^T (x_i - x_j)$$

if $||w|| = 1$ (normal vector).



[Slide from Eric Xing]

Maximum Margin Classification

- The margin is:

$$m = w^T\left(x_{i^*} - x_{j^*}\right)$$

- Here is our Maximum Margin Classification problem:

$$\begin{aligned} \max_{w,b} \quad & m \\ \text{s.t} \quad & y_i(w^T x_i + b) \geq m, \quad \forall i \\ & \|w\| = 1 \end{aligned}$$

- Equivalently, we can instead work on this (*w*→ *w*/||*w*||, *m*→*m*/||*w*||):

$$\begin{aligned} \max_{w,b} \quad & \frac{m}{\|w\|} \\ \text{s.t} \quad & y_i(w^T x_i + b) \geq m, \quad \forall i \end{aligned}$$

[Slide from Eric Xing]

Maximum Margin Classification

- The optimization problem:

$$\max_{w,b} \quad \frac{m}{\|w\|}$$
$$\text{s.t} \quad y_i(w^T x_i + b) \geq m, \quad \forall i$$

- But note that the magnitude of *m* merely scales *w* and *b*, and does not change the classification boundary at all!
- So we instead work on this cleaner problem (*w*→*w/m*, *b*→*b/m*):

$$\max_{w,b} \quad \frac{1}{\|w\|}$$
$$\text{s.t} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i$$

- The solution to this leads to the famous **Support Vector Machines** --believed by many to be the best "off-the-shelf" supervised learning algorithm.

[Slide from Eric Xing]

Support vector machine

- A convex quadratic programming problem with linear constrains:

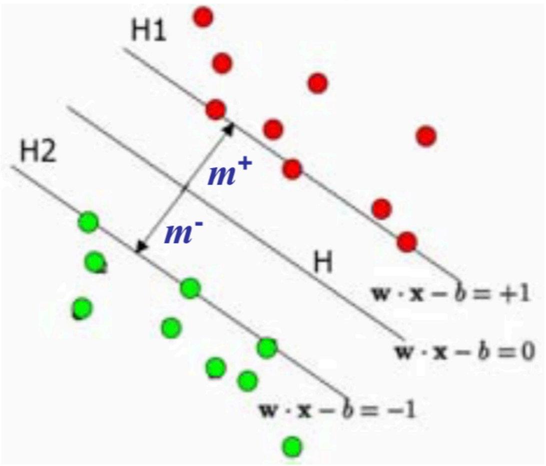
$$\max_{w,b} \quad \frac{1}{\|w\|}$$

$$\text{s.t} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i$$

  - The attained margin is now given by *1/||w||*.
  - Only a few of the classification constraints are relevant → **support vectors**
- Constrained optimization
  - We can directly solve this using commercial quadratic programming (QP) code
  - But we want to take a more careful investigation of Lagrange duality, and the solution of the above is its dual form.

  → deeper insight: support vectors, kernels ...

  → more efficient algorithm



[Slide from Eric Xing]

Lagrangian Duality

- The Primal Problem

Primal:

$$\min_w \quad f(w)$$
$$\text{s.t.} \quad g_i(w) \leq 0, \quad i = 1, \ldots, k$$
$$h_i(w) = 0, \quad i = 1, \ldots, l$$

The generalized Lagrangian:

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

the α's ($\alpha_i \geq 0$) and βs are called the Lagarangian multipliers

Lemma:

$$\max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{o/w} \end{cases}$$

A re-written Primal:

$$\min_w \max_{\alpha, \beta, \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

[Slide from Eric Xing]

Lagrangian Duality

- Recall the Primal Problem:

$$\min_w \max_{\alpha,\beta,\alpha_i \geq 0} \mathcal{L}(w,\alpha,\beta)$$

- The Dual Problem:

$$\max_{\alpha,\beta,\alpha_i \geq 0} \min_w \mathcal{L}(w,\alpha,\beta)$$

- **Theorem (weak duality):**

$$d^* = \max_{\alpha,\beta,\alpha_i \geq 0} \min_w \mathcal{L}(w,\alpha,\beta) \leq \min_w \max_{\alpha,\beta,\alpha_i \geq 0} \mathcal{L}(w,\alpha,\beta) = p^*$$

- **Theorem (strong duality):**

$$d^* = p^*$$

Next page: Slater's condition

[Slide from Eric Xing]

Slater's condition

$$\begin{array}{ll} \min_w & f(w) \\ \text{s.t.} & g_i(w) \leq 0, \quad i = 1, \ldots, k \\ & h_i(w) = 0, \quad i = 1, \ldots, l \end{array}$$

If the primal is a convex problem (i.e., f and g_i are convex, h_i are affine), and there exists at least one strictly feasible w, meaning

$$g_i(w) < 0, \text{ and } h_i(w)=0$$

then strong duality holds.

- For SVM primal problem, it's easy to see Slater's condition satisfied, therefore strong duality holds.

$$\begin{array}{ll} \min_{w,b} & \frac{1}{2} w^T w \\ \text{s.t} & 1 - y_i(w^T x_i + b) \leq 0, \quad \forall i \end{array} \qquad (*)$$

[Slide from Ryan Tibshirani]

The KKT conditions

- Under the strong duality, the following "Karush-Kuhn-Tucker" (KKT) conditions satisfies:

Stationary: $$\frac{\partial}{\partial w_i}\mathcal{L}(w,\alpha,\beta)=0, \quad i=1,\ldots,n$$

Primal feasibility: $$\frac{\partial}{\partial \beta_i}\mathcal{L}(w,\alpha,\beta)=0, \quad i=1,\ldots,l$$ → $h_i(w)=0$

Complementary slackness: $$\alpha_i g_i(w)=0, \quad i=1,\ldots,k$$

Primal feasibility: $$g_i(w)\leq 0, \quad i=1,\ldots,k$$

Dual feasibility: $$\alpha_i \geq 0, \quad i=1,\ldots,k$$

[Slide from Eric Xing]

KKT conditions

- Always sufficient: If w^* and α^*, β^* satisfy the KKT conditions, strong duality holds.
- Necessary under strong duality: If w^* and α^*, β^* are primal and dual solutions, with zero duality gap, then w^* and α^*, β^* satisfy the KKT conditions
  - Assumes nothing a priori about convexity of the problem
  - It is not necessary even in convex problem:

$$\min x$$

$$\text{s.t.} \quad x^2 \leq 0.$$

[Slide from Ryan Tibshirani]

Solving optimal margin classifier

- Recall our opt problem:

$$\max_{w,b} \quad \frac{1}{\|w\|}$$
$$\text{s.t} \quad y_i(w^T x_i + b) \ge 1, \quad \forall i$$

- This is equivalent to

$$\min_{w,b} \quad \frac{1}{2} w^T w$$
$$\text{s.t} \quad 1 - y_i(w^T x_i + b) \le 0, \quad \forall i \qquad (*)$$

- Write the Lagrangian:

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

Recall that (*) can be reformulated as $\min_{w,b} \max_{\alpha_i \ge 0} \mathcal{L}(w,b,\alpha)$

Now we solve its **dual problem**: $\max_{\alpha_i \ge 0} \min_{w,b} \mathcal{L}(w,b,\alpha)$

[Slide from Eric Xing]

The Dual Problem

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

$$\max_{\alpha_i\geq 0}\min_{w,b}\mathcal{L}(w,b,\alpha)$$

- We minimize L with respect to w and b first:

$$\nabla_w\mathcal{L}(w,b,\alpha)=w-\sum_{i=1}^{m}\alpha_i y_i x_i=0, \qquad (*)$$

$$\nabla_b\mathcal{L}(w,b,\alpha)=\sum_{i=1}^{m}\alpha_i y_i=0, \qquad (**)$$

Note that **(*)** implies: $$w=\sum_{i=1}^{m}\alpha_i y_i x_i \qquad (***)$$

- Plus (***) back to L, and using (**), we have:

$$\mathcal{L}(w,b,\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i^T\mathbf{x}_j)$$

[Slide from Eric Xing]

The Dual problem

- Now we have the following dual opt problem:

$$\max_\alpha \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, k$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- This is,(again,) a **quadratic programming** problem.
  - A global maximum of α_i can always be found.
  - But what's the big deal?
  - Note two things:

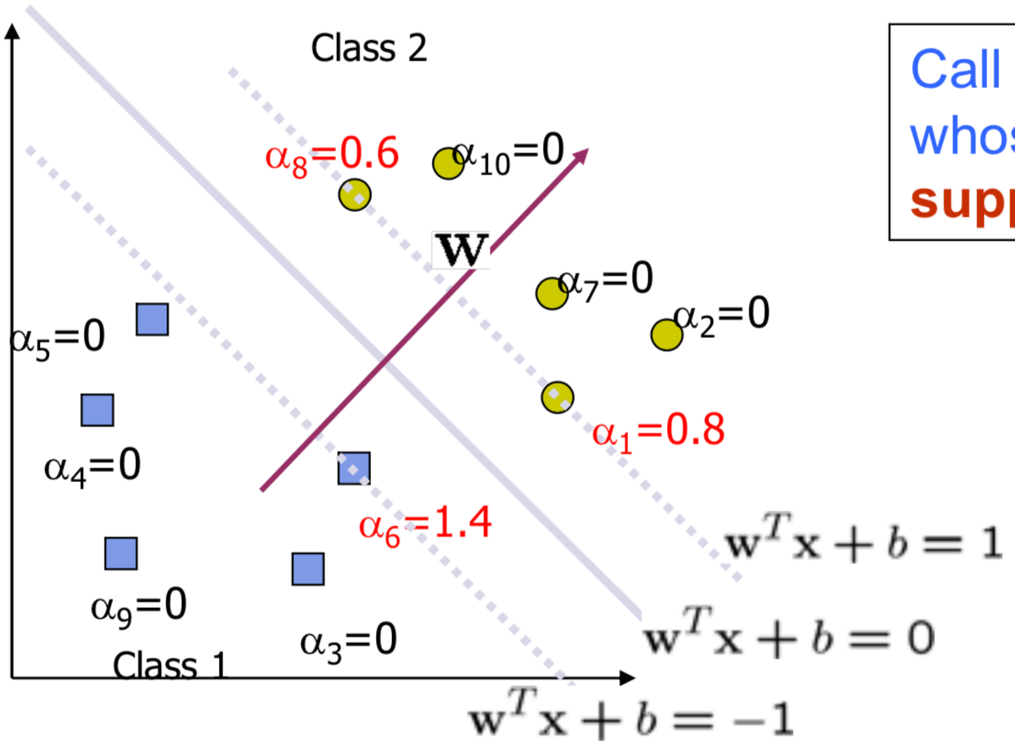
1. w can be recovered by $w = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$ See next …
2. The "kernel" $\mathbf{x}_i^T \mathbf{x}_j$ More later …

[Slide from Eric Xing]

Support vectors

- Note the KKT condition --- only a few α_i's can be nonzero!

$$\alpha_i g_i(w) = 0, \quad i = 1, \ldots, k$$



Call the training data points whose α_i's are nonzero the **support vectors** (SV)

[Slide from Eric Xing]

Support vector machines

- Once we have the Lagrange multipliers $\{\alpha_i\}$, we can reconstruct the parameter vector *w* as a weighted combination of the training examples:

$$w = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$$

- For testing with a new data ***z***
- Compute

$$w^T z + b = \sum_{i \in SV} \alpha_i y_i \left(\mathbf{x}_i^T z \right) + b$$

and classify ***z*** as class 1 if the sum is positive, and class 2 otherwise

- Note: *w* need not be formed explicitly

[Slide from Eric Xing]

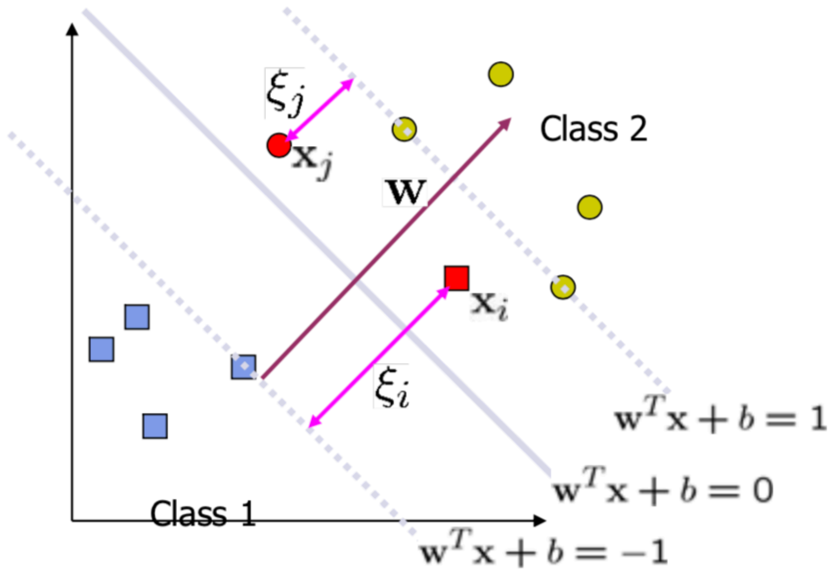
Interpretation of support vector machines

- The optimal ***w*** is a linear combination of a small number of data points. This “sparse” representation can be viewed as data compression as in the construction of kNN classifier
- To compute the weights $\{\alpha_i\}$, and to use support vector machines we need to specify only the inner products (or kernel) between the examples $\mathbf{x}_i^T \mathbf{x}_j$
- We make decisions by comparing each new example ***z*** with only the support vectors:

$$y^* = \text{sign}\left(\sum_{i \in SV} \alpha_i y_i \left(\mathbf{x}_i^T z\right) + b\right)$$

[Slide from Eric Xing]

Non-linearly Separable Problems



- We allow "error" ξ_i in classification; it is based on the output of the discriminant function $\boldsymbol{w}^T\boldsymbol{x}+b$
- ξ_i approximates the number of misclassified samples

[Slide from Eric Xing]

Soft Margin Hyperplane

- Now we have a slightly different opt problem:

$$\min_{w,b} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i$$

$$\text{s.t} \quad \begin{array}{l} y_i (w^T x_i + b) \geq 1 - \xi_i, \quad \forall i \\ \xi_i \geq 0, \quad \forall i \end{array}$$

- ξ_i are "slack variables" in optimization
- Note that ξ_i=0 if there is no error for $\mathbf{x}_i$
- ξ_i is an upper bound of the number of errors
- C : tradeoff parameter between error and margin

[Slide from Eric Xing]

The Optimization Problem

- The dual of this new constrained optimization problem is

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, k$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound C on α_i now.
- Once again, a QP solver can be used to find α_i.

[Slide from Eric Xing]

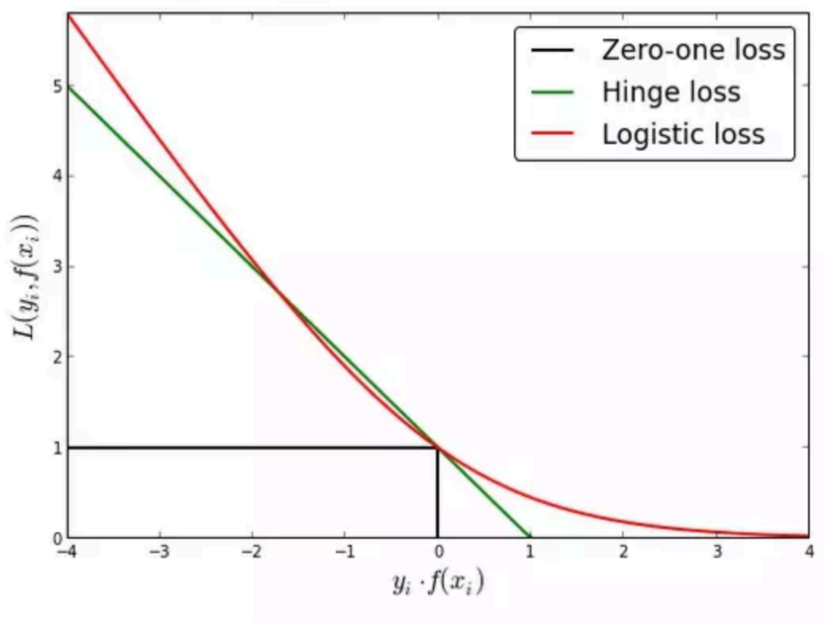
SVM vs logistic regression

- SVM with soft margin and logistic regression with L_2 regularization

$$\min_w \lambda\|w\|^2 + \sum_i \max\{0, 1 - y_i w^T x_i\}$$

$$\min_w \lambda\|w\|^2 + \sum_i \log(1 + \exp(1 - y_i w^T x_i))$$

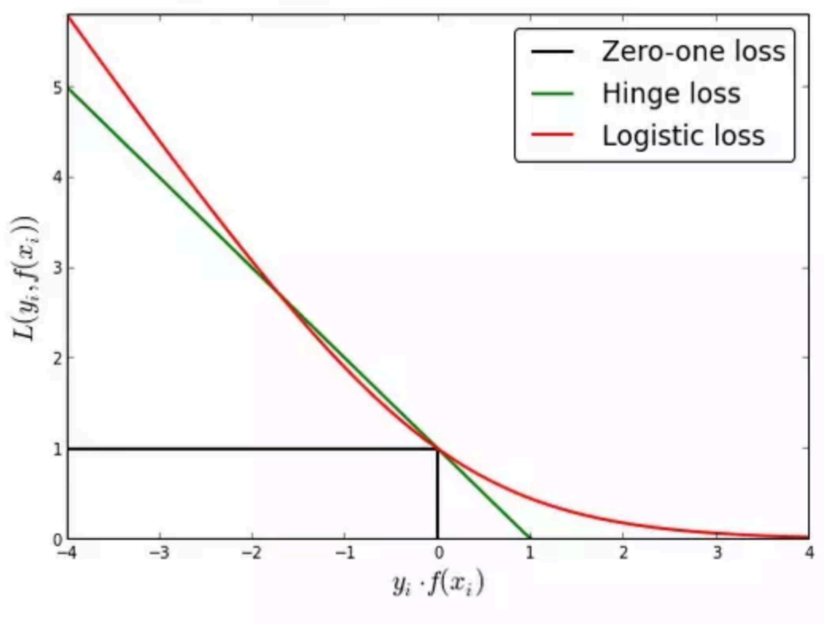
- They only differ in the loss functions -- SVM minimizes hinge loss while logistic regression minimizes logistic loss



[Figure from https://towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f]

SVM vs logistic regression

- Logistic loss diverges faster than hinge loss. So, in general, it will be more sensitive to outliers.
- Logistic loss does not go to zero even if the point is classified sufficiently confidently. This might lead to minor degradation in accuracy.

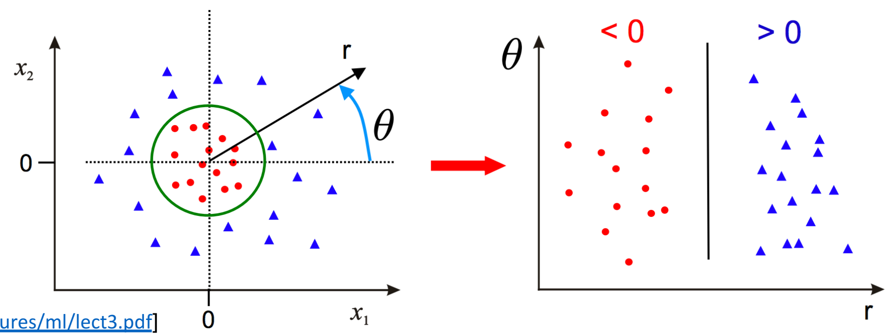


Loss functions

[Figure from https://towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f]

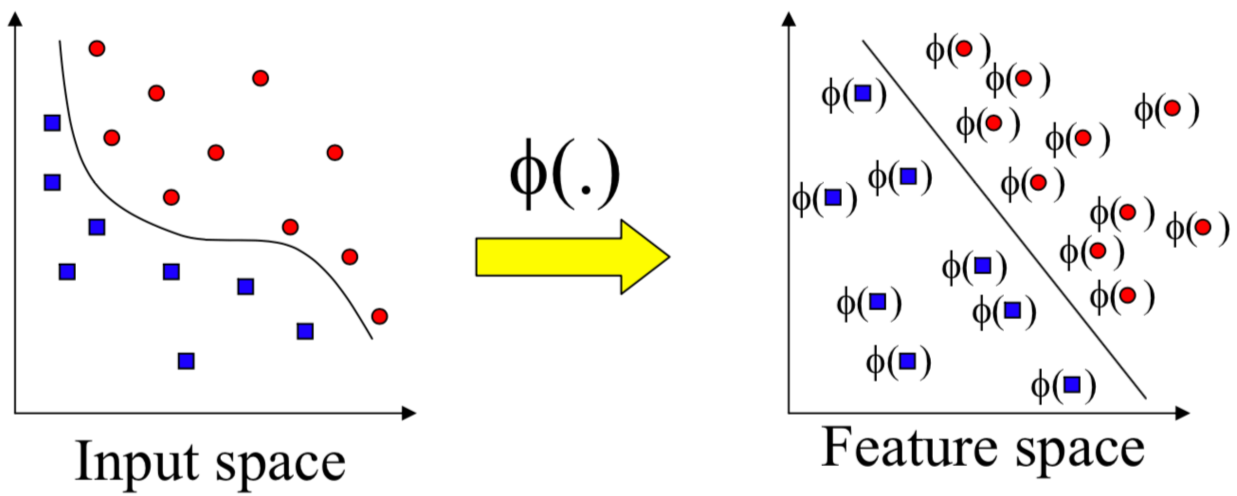
Extension to Non-linear Decision Boundary

- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become nonlinear?
- Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space the point $\mathbf{x}_i$ are located
  - Feature space: the space of $\varphi(\mathbf{x}_i)$ after transformation
- Why transform?
  - Linear operation in the feature space is equivalent to non-linear operation in input space
  - Classification can become easier with a proper transformation. In the XOR problem, for example, adding a new feature of x_1 x_2 make the problem linearly separable



[Slide from Eric Xing and http://www.robots.ox.ac.uk/~az/lectures/ml/lect3.pdf]

Transforming the Data



Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

[Slide from Eric Xing]

The Kernel Trick

- Recall the SVM optimization problem

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, k$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- The data points only appear as inner product
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function *K* by $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

[Slide from Eric Xing]

An Example for feature mapping and kernels

Consider an input $\mathbf{x}=[x_1,x_2]$

Suppose $\phi(.)$ is given as follows

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = 1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2$$

An inner product in the feature space is

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} x_1' \\ x_2' \end{bmatrix}\right) \right\rangle =$$

So, if we define the **kernel function** as follows, there is no need to carry out $\phi(.)$ explicitly

$$K(\mathbf{x},\mathbf{x}') = \left(1 + \mathbf{x}^T\mathbf{x}'\right)^2$$

[Slide from Eric Xing]

More examples of kernel functions

Linear kernel (we've seen it)

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

Polynomial kernel (we just saw an example)

$$K(\mathbf{x}, \mathbf{x}') = \left(1 + \mathbf{x}^T \mathbf{x}'\right)^p$$

where p = 2, 3, … To get the feature vectors we concatenate all pth order polynomial terms of the components of x (weighted appropriately)

Radial basis kernel

$$K(x, y) = \exp\left(-\frac{\|x - y\|_2^2}{2\sigma^2}\right) = \phi(x)^T \phi(y)$$

In this case the feature space consists of functions and results in a non-parametric classifier.

$$\phi(x) = e^{-x^2/2\sigma^2}\left[1, \sqrt{\frac{1}{1!\sigma^2}}x, \sqrt{\frac{1}{2!\sigma^4}}x^2, \sqrt{\frac{1}{3!\sigma^6}}x^3, \ldots\right]^T$$

[Slide from Eric Xing and https://stats.stackexchange.com/questions/69759/feature-map-for-the-gaussian-kernel]

Kernelized SVM

- Training:

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, k$$

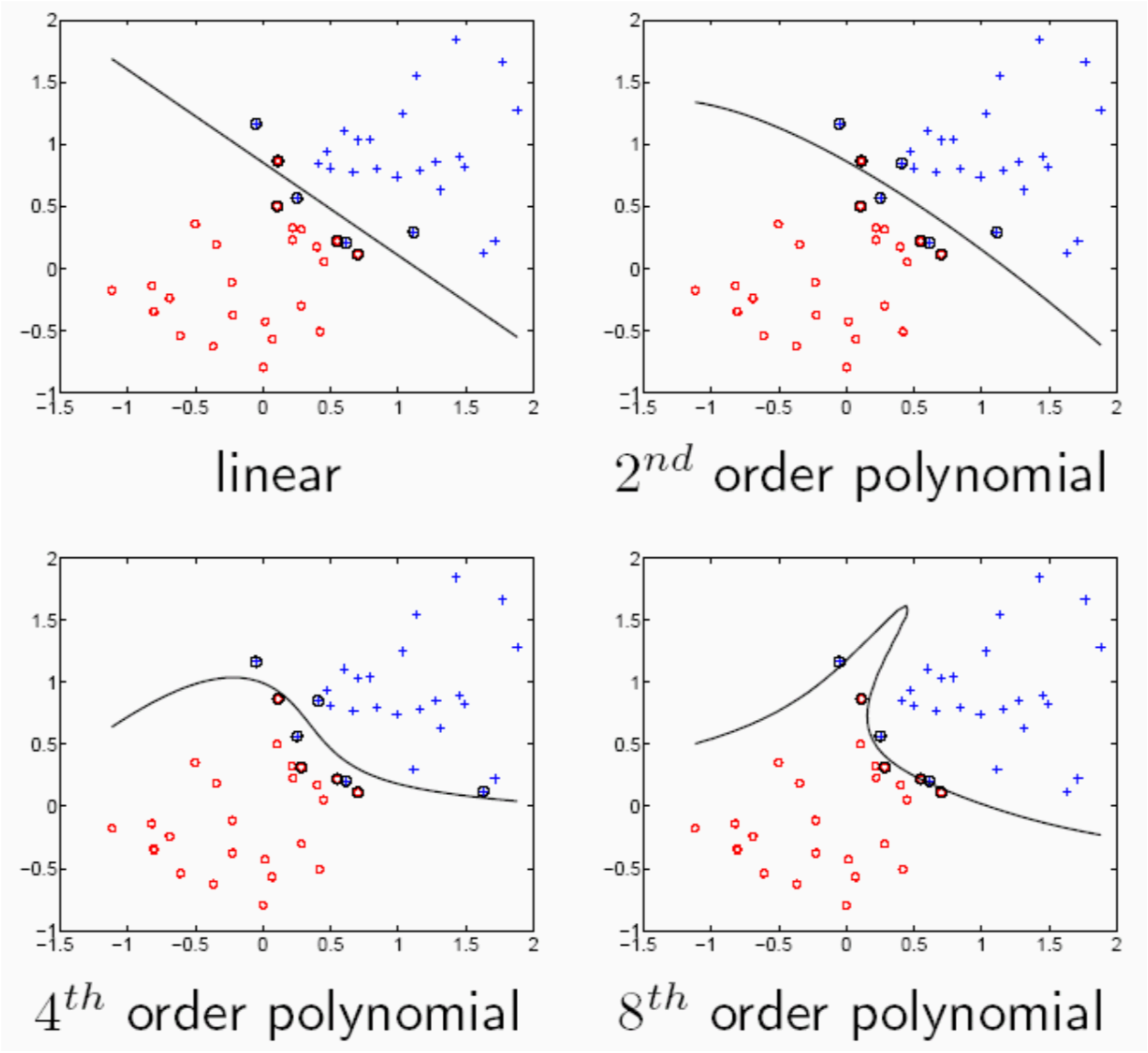
$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- Using:

$$y^* = \text{sign}\left(\sum_{i \in SV} \alpha_i y_i K(\mathbf{x}_i, z) + b \right)$$

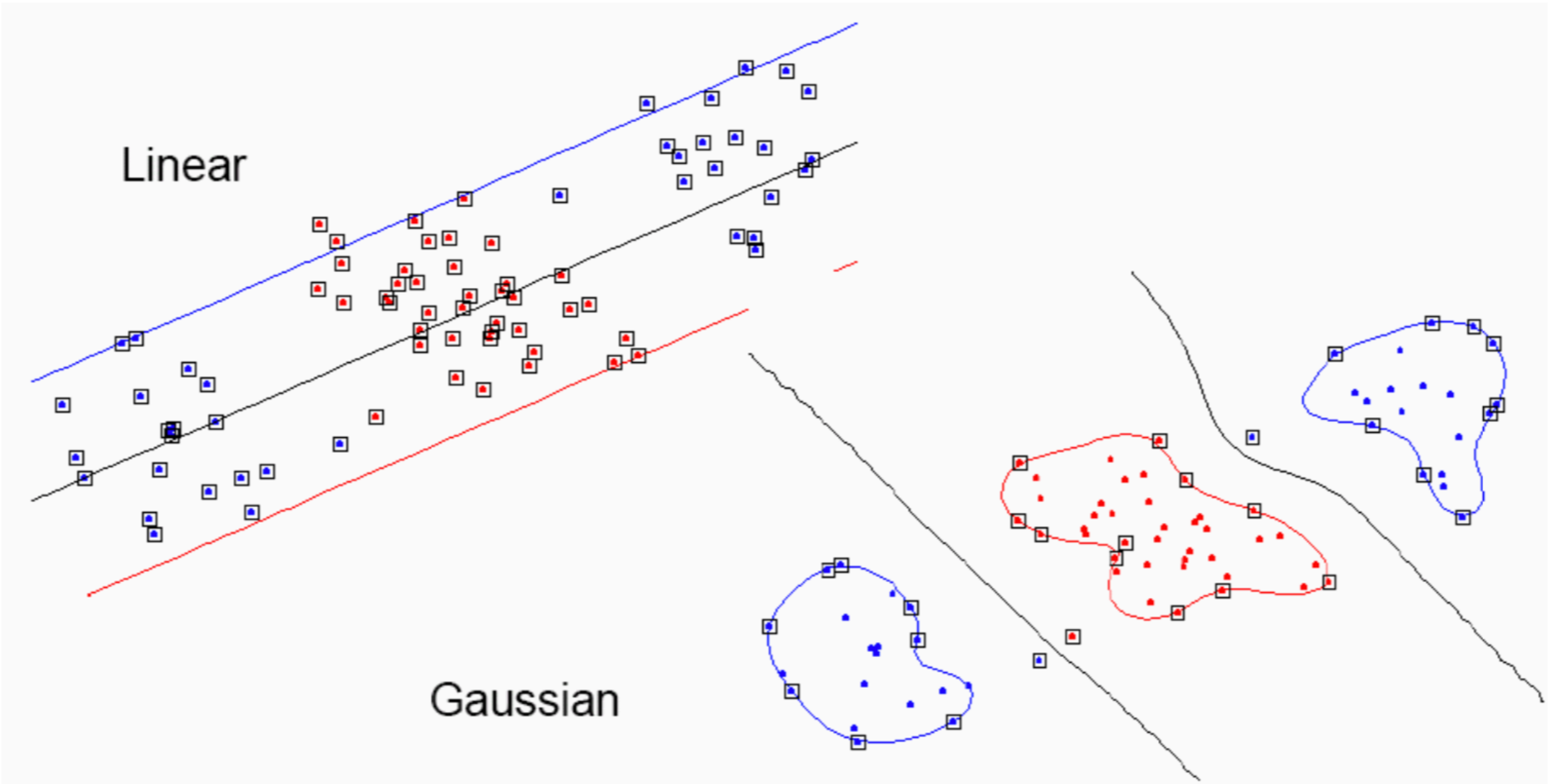
[Slide from Eric Xing]

SVM examples



[Slide from Eric Xing]

Examples for Non Linear SVMs – Gaussian Kernel

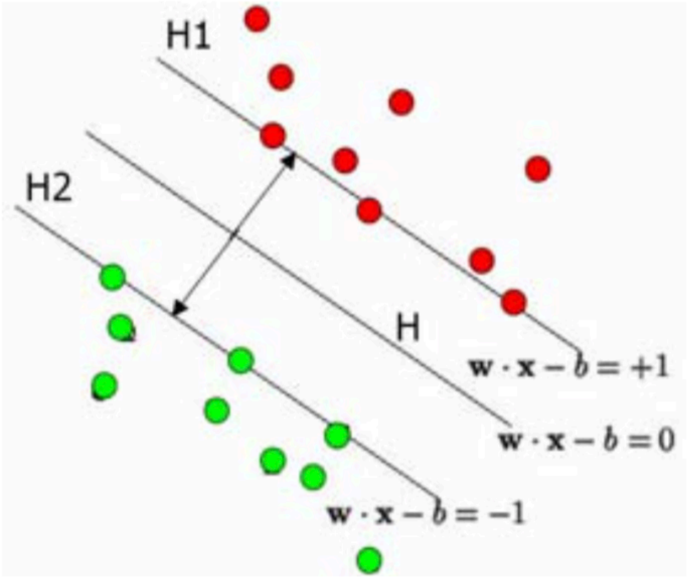


[Slide from Eric Xing]

Cross-validation error

- The leave-one-out cross-validation error does not depend on the dimensionality of the feature space but only on the # of support vectors!

$$\text{Leave-one-out CV error} = \frac{\#\ \text{support vectors}}{\#\ \text{of training examples}}$$



[Slide from Eric Xing]

Take home message

- SVM is a maximum margin classifier
- Lagrangian, weak and strong duality
  - Weak duality always holds
  - Slater’s condition is sufficient for strong duality
- KKT and solutions to primal and dual problems
  - KKT conditions are always sufficient
  - KKT conditions are necessary under strong duality
- Extension to the vanilla SVM
  - Soft margin to solve problems of outliers in SVM
  - Kernel tricks to treat non-linear decision boundary in SVM

References

- Eric Xing, Tom Mitchell. 10701 Introduction to Machine Learning: http://www.cs.cmu.edu/~epxing/Class/10701-06f/
- Eric Xing, Ziv Bar-Joseph. 10701 Introduction to Machine Learning: http://www.cs.cmu.edu/~epxing/Class/10701/
- Ryan Tibshirani. 10725 Convex Optimization: http://www.stat.cmu.edu/~ryantibs/convexopt/