Support Vector Machines and Hierarchical Clustering

Machine Learning 10601 - Fall 2010

Hai-Son Le

Machine Learning Department
Carnegie Mellon University

hple@cs.cmu.edu

October 18, 2010

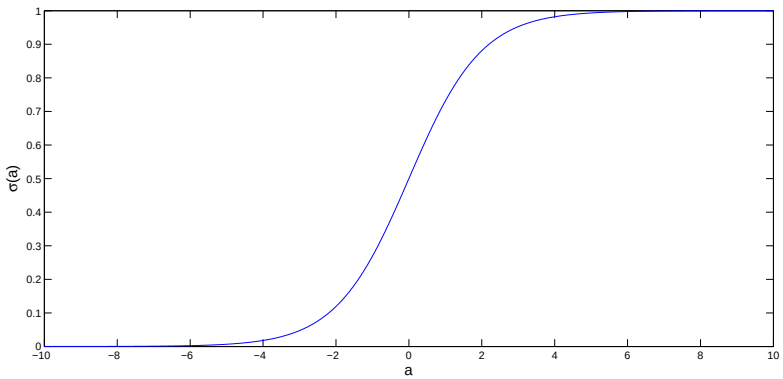
Linear classifiers

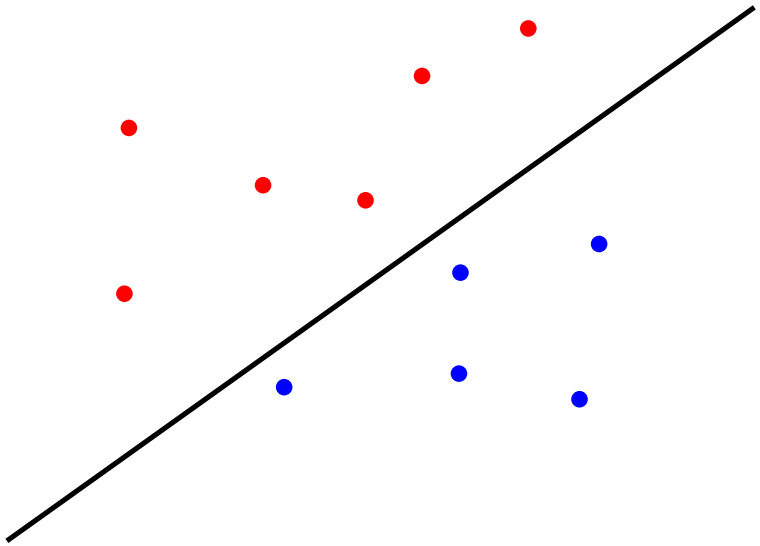
We studied: Logistic Regression

$$p(y = 0|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b) \quad (1)$$

with

$$\sigma(a) = \frac{1}{1 + \exp(-a)} \quad (2)$$

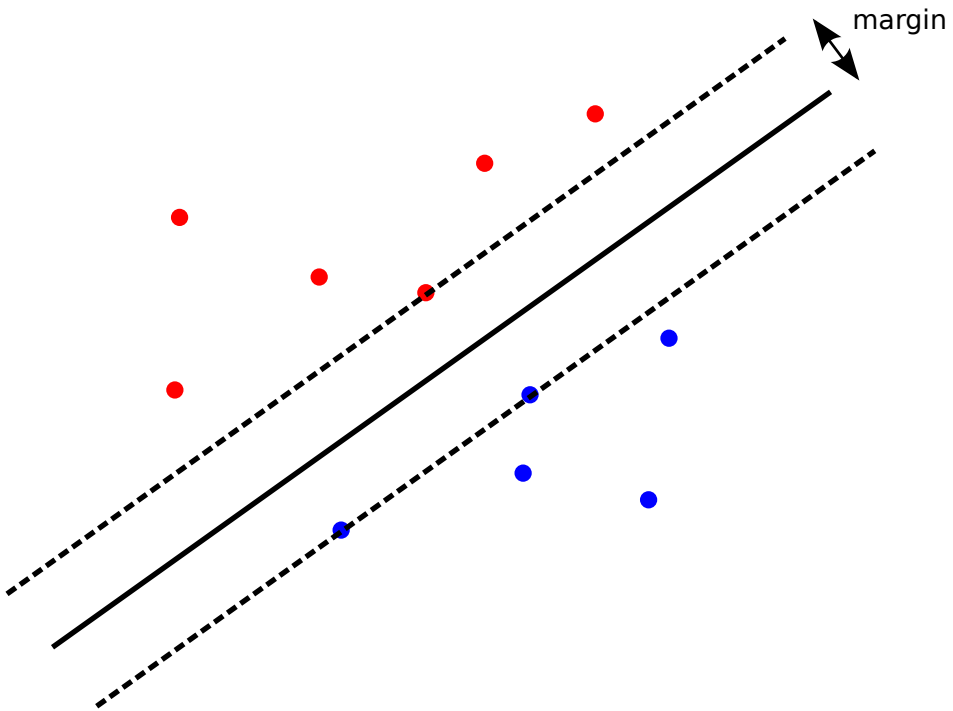




MLE to estimate $\mathbf{w}$, b for logistic regression.

Margin concept

SVM uses the concept of *margin* to find the decision boundary.



The margin = distance from the decision boundary to the *closest* negative/positive examples.

Intuition

- Test examples tend to be close to training examples.
- Hard examples are those close to the decision boundary.
- Therefore, we would like to find a decision boundary with the *maximum margin*.
- "Minimize the chance of getting hard examples"

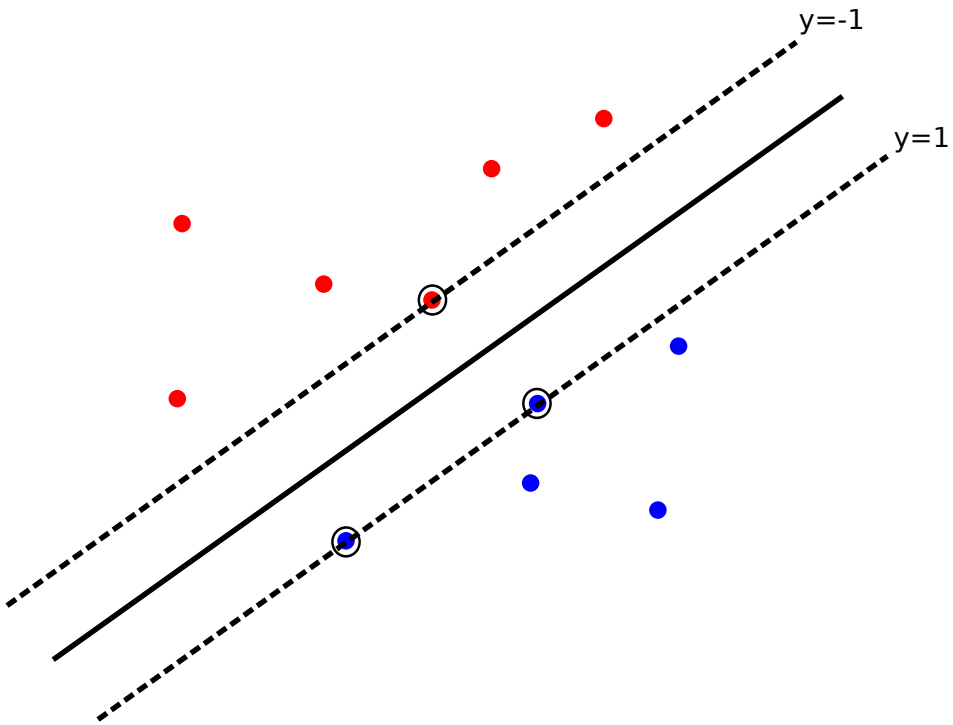
Formalization

Given a set of N training examples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ with target labels $y_1, \ldots, y_N$.

- $y_i \in \{-1, 1\}$. *Before we use* $\{0, 1\}$ *for class labels.*
-

$$y = \begin{cases} -1 & , \text{if } \mathbf{w}^T\mathbf{x} + b < 0 \\ 1 & , \text{if } \mathbf{w}^T\mathbf{x} + b > 0 \end{cases} \tag{3}$$

Formalization



$$d(\mathbf{w_x}, \mathbf{w}^T\mathbf{x} + b = 0) = \frac{|\mathbf{w}^T\mathbf{x_i} + b|}{\sqrt{\mathbf{w}^T\mathbf{w}}} \tag{4}$$

Proof: PS4

Formalization

First attempt:

$$\text{argmax}_{\mathbf{w},b} \frac{1}{\mathbf{w}^T\mathbf{w}} \min_i y_i(\mathbf{w}^T\mathbf{x_i} + b) \tag{5}$$

- $|\mathbf{w}^T\mathbf{x_i} + b|$ substituted by $y_i(\mathbf{w}^T\mathbf{x_i} + b)$. Why?
- Claim: margin $= \frac{1}{\mathbf{w}^T\mathbf{w}} \min_i y_i(\mathbf{w}^T\mathbf{x_i} + b)$

Formalization

We can simplify further.

- Restrict $\min_i y_i(\mathbf{w}^T\mathbf{x_i} + b) = 1$. This does not change the decision boundary!
- The margin is now $\frac{1}{\mathbf{w}^T\mathbf{w}}$.
- For any point: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$. (All points are farther than the margin.)

Formalization

$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} \quad \text{such that } y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1, \forall i \tag{6}$$

This is a Quadratic Programming!

Quadratic Programming (QP)

Minimizing a quadratic function subject to a set of linear equality and inequality constraints.

$$\begin{aligned} \min_{\mathbf{x}} \ & \frac{1}{2}\mathbf{x}^T P\mathbf{x} + \mathbf{q}^T\mathbf{x} + r \\ \text{such that } & G\mathbf{x} \preceq h \\ & A\mathbf{x} = b \end{aligned} \tag{7}$$

There are generic solvers for QP. We can also solve using Gradient Descent, but slow!!!

Convex Programming and Duality

- QP belongs to a larger class of Convex Programming.

$$\begin{aligned} &\min_x f(x) \\ \text{such that } &g_i(x) \leq 0 \\ &h_i(x) = 0 \end{aligned} \tag{8}$$

  where $f(x), g_i(x)$ are *convex* functions and $h_i(x)$ are affine ($h_i(x) = a^T x - b$).
- Good news: Convex Programming has a unique local minima = global minima!

Convex Programming and Duality

- Lagrangian multipliers convert the constrained optimization to a non-constrained optimization.

$$\max_{\lambda,\nu} \min_{x} L(x, \lambda, \nu) = f(x) + \sum_i \lambda_i g_i(x) + \sum_i \nu h_i(x) \tag{9}$$

  such that $\lambda \geq 0$ and $\nu \geq 0$.
- We can transform the original *min* optimzation into *max* optimization, called a dual problem.
- Why? Dual problem offers a different view of the problem. Often this yields better insight!

Dual problem of SVM

$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} \quad \text{such that } y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1, \forall i \tag{10}$$

Applying Lagrangian multipliers α_i for each i-th constraint.

$$\max_{\alpha} \min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_i \alpha_i \Big(y_i(\mathbf{w}^T\mathbf{x_i} + b) - 1 \Big) \tag{11}$$

Dual problem of SVM

Simplifying (Problem set), we get the dual problem.

$$\begin{aligned} &\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ &\text{such that } \sum_i \alpha_i y_i = 0 \\ &\alpha_i \geq 0, \forall i \end{aligned} \tag{12}$$

Observations

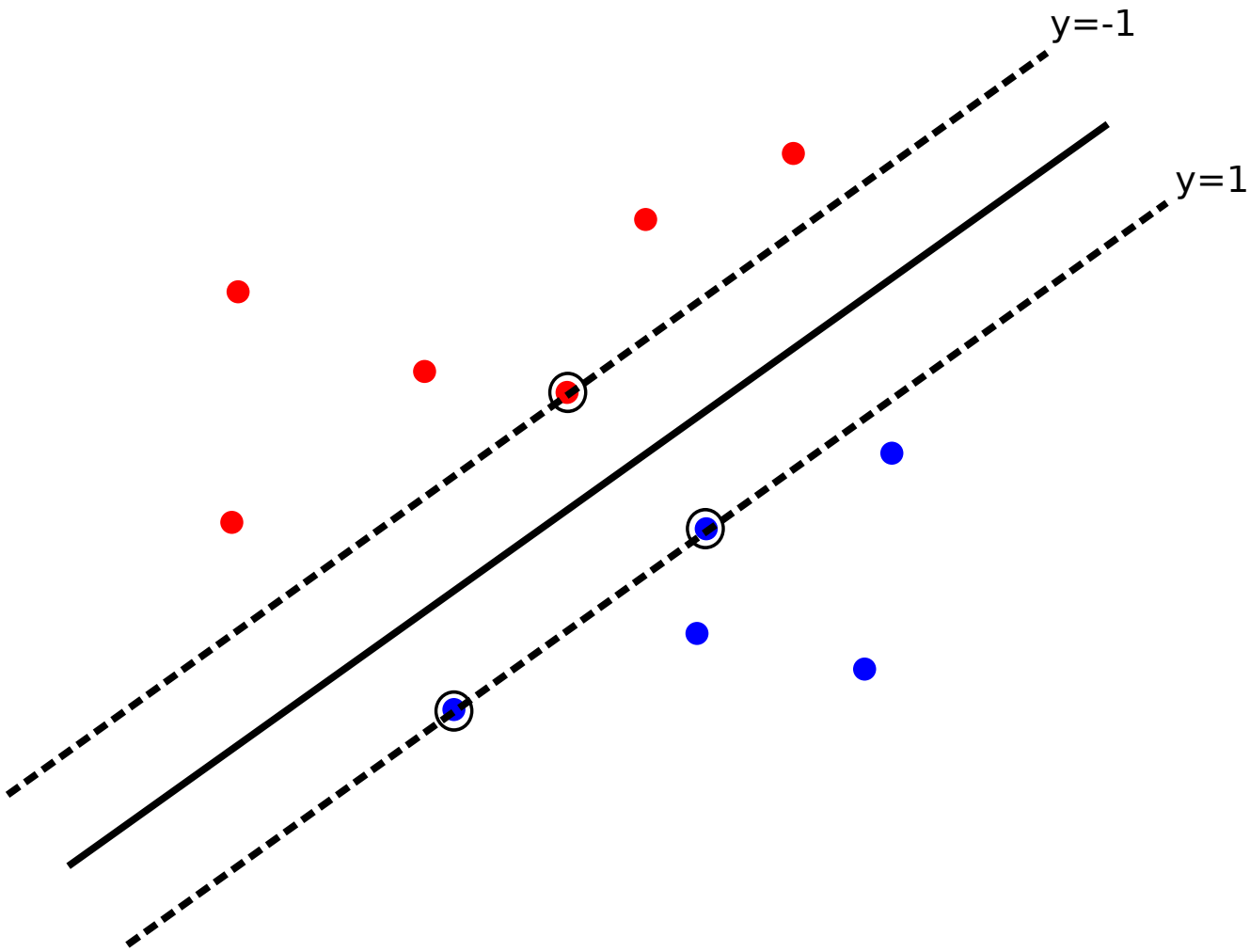
Furthermore, the optimal solution satisfies:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \tag{13}$$

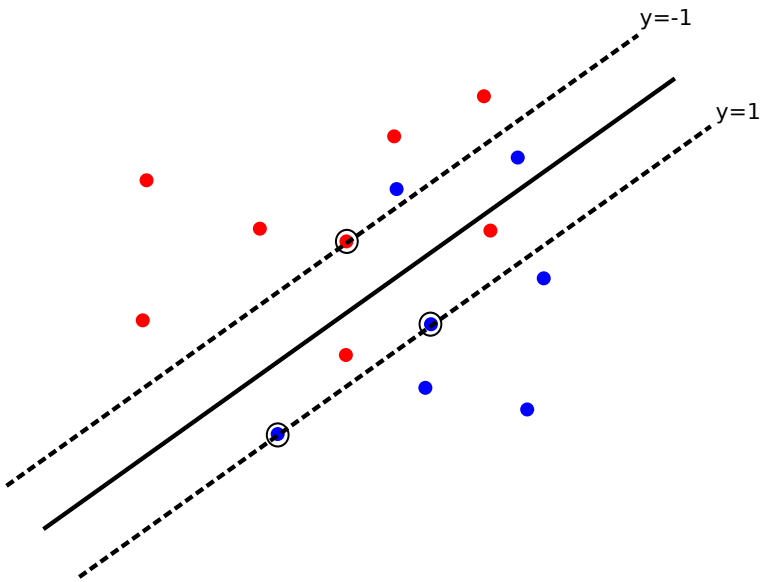
$$b = y_i - \mathbf{w}^T \mathbf{x}_i \tag{14}$$

- $\mathbf{w}$ only depends on the example i whose $\alpha_i > 0$.
- We call these examples: *support vectors*.
- Adding to the training set examples that are outside of the margin, does not affect the solution!
- To classify a new example, we only need to consider the *support vectors*!

Recap



Not linearly separable :(

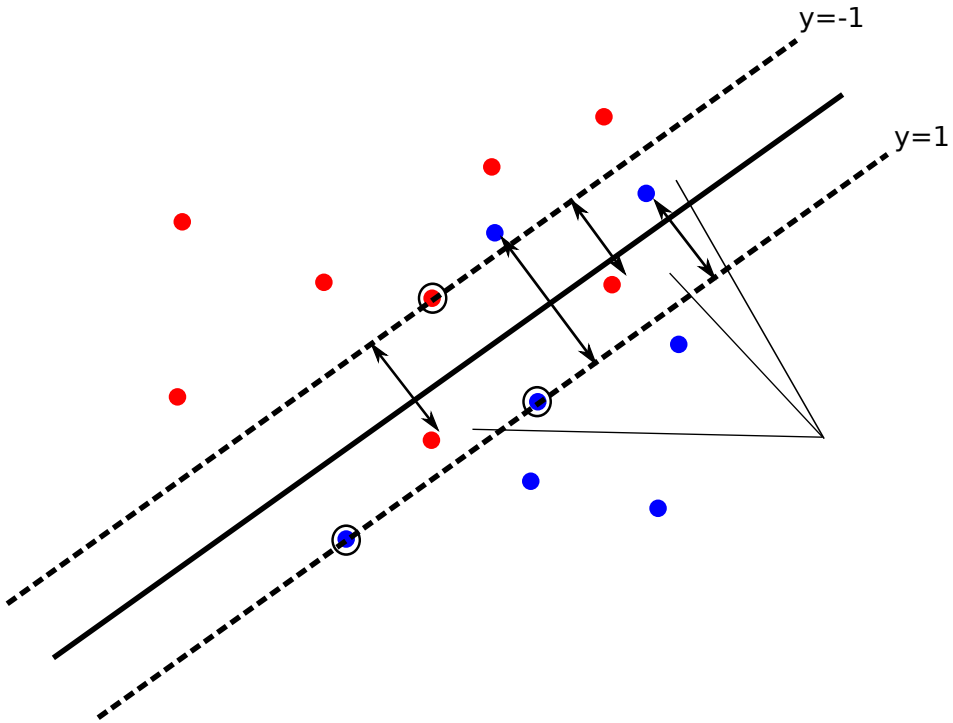


We need to compromise:

- Allow some training examples to be within the margin.
- But we want to minimize the number of such examples.

Formalization

- Introduce variables ϵ_i = the violation of the i-th example. $\epsilon_i \geq 0$.



Formalization

$$\begin{aligned} &\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \epsilon_i \\ &\text{such that } y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \epsilon_i, \forall i \\ &\epsilon_i \geq 0, \forall i \end{aligned} \tag{15}$$

where C is some constant.

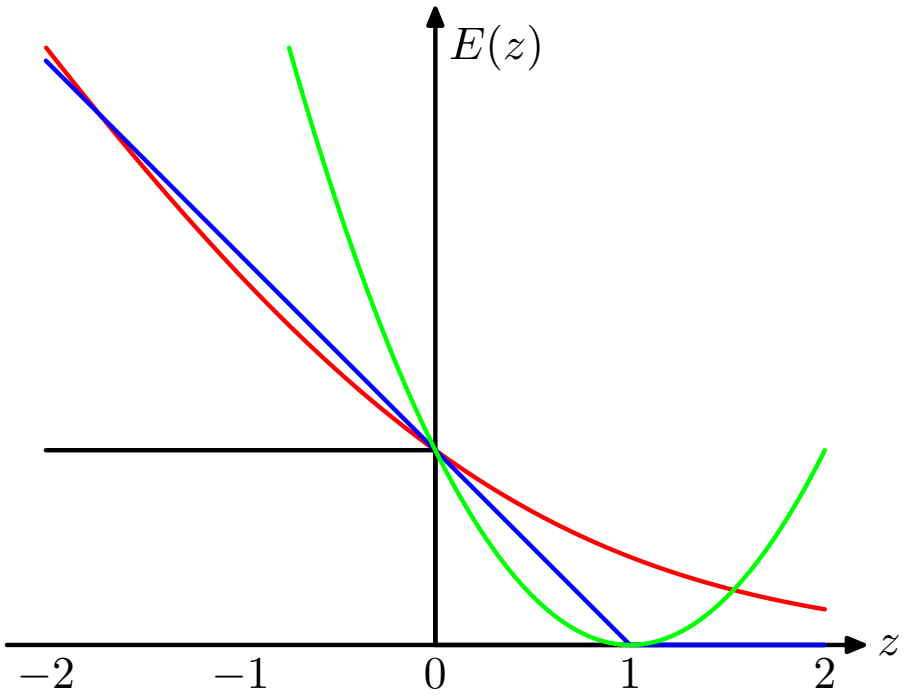
Dual problem

We can use the Lagrangian multipliers to get the dual problem

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$
$$\text{such that } \sum_i \alpha_i y_i = 0 \tag{16}$$
$$0 \leq \alpha_i \leq C, \forall i$$

SVM v.s. other classifiers

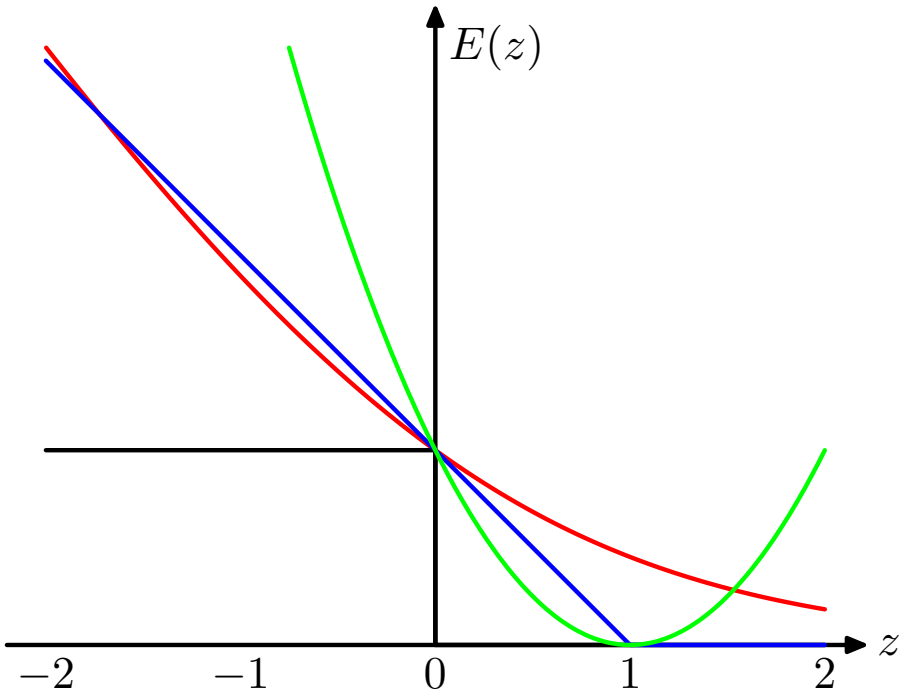
Another way to justify SVM is in term of the loss function!



- $z = (\mathbf{w}^T \mathbf{x_i} + b) - y_i$
- $E(z) =$ error
- 0-1 loss?

SVM v.s. other classifiers

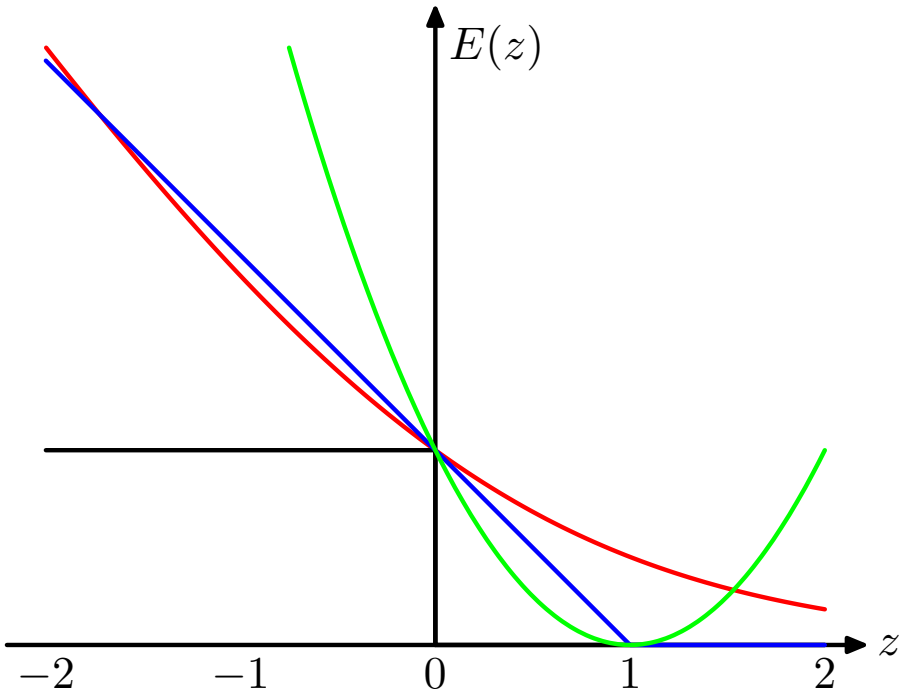
Another way to justify SVM is in term of the loss function!



- $z = (\mathbf{w}^T \mathbf{x_i} + b) - y_i$
- $E(z) =$ error
- 0-1 loss? black line
- squared loss?

SVM v.s. other classifiers

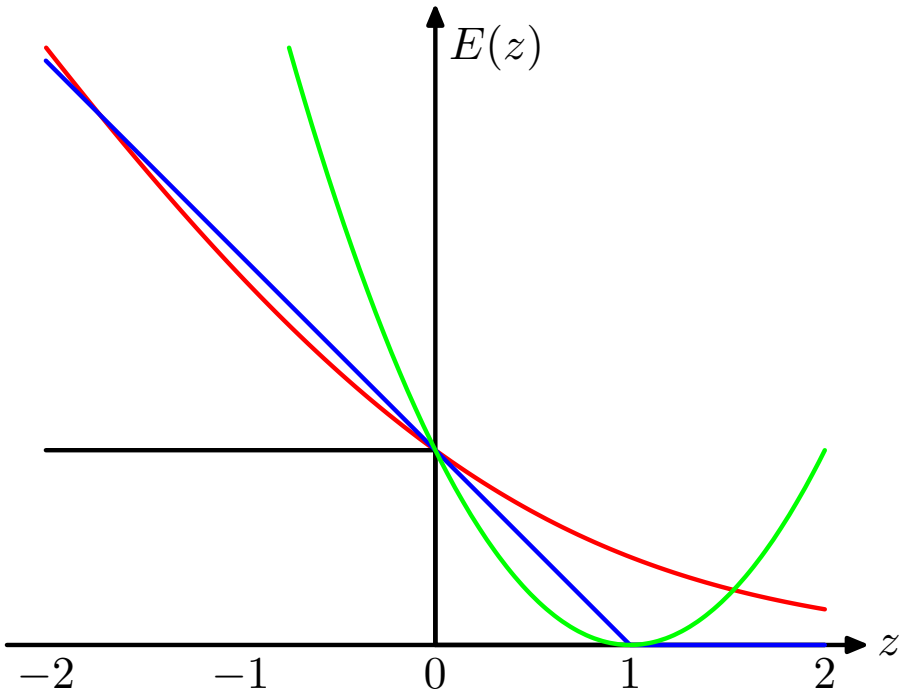
Another way to justify SVM is in term of the loss function!



- $z = (\mathbf{w}^T \mathbf{x_i} + b) - y_i$
- $E(z) =$ error
- 0-1 loss? black line
- squared loss? green line
- logistic regression loss?

SVM v.s. other classifiers

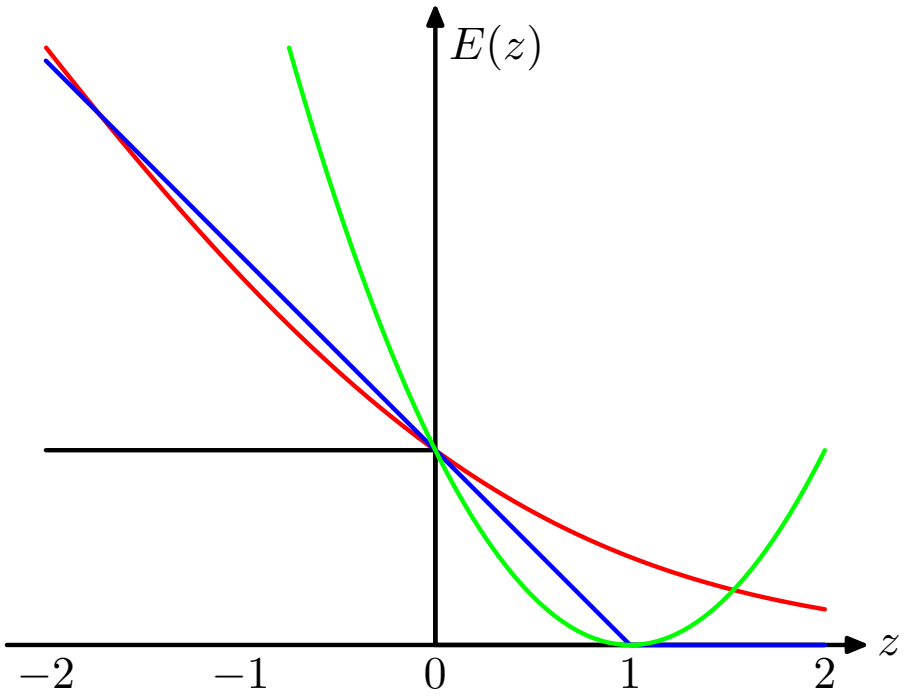
Another way to justify SVM is in term of the loss function!



- $z = (\mathbf{w}^T \mathbf{x_i} + b) - y_i$
- $E(z) =$ error
- 0-1 loss? black line
- squared loss? green line
- logistic regression loss? red line
- SVM loss?

SVM v.s. other classifiers

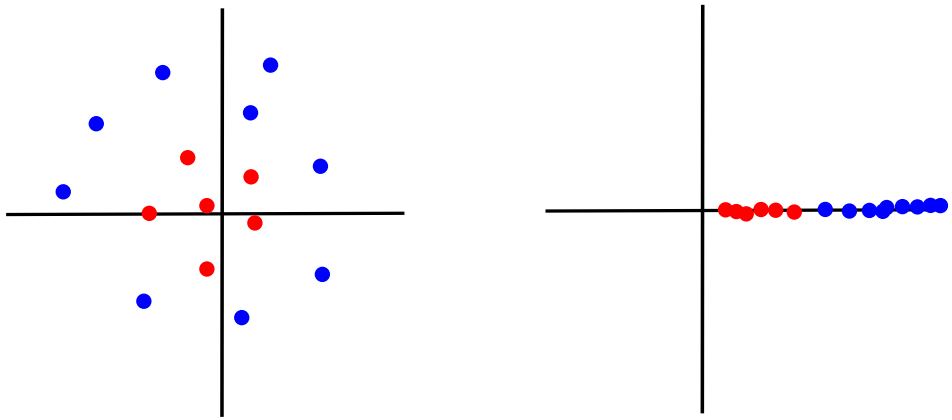
Another way to justify SVM is in term of the loss function!



- $z = (\mathbf{w}^T \mathbf{x_i} + b) - y_i$
- $E(z) =$ error
- 0-1 loss? black line
- squared loss? green line
- logistic regression loss? red line
- SVM loss? blue line

Kernel trick

- We can transform the original feature space $\mathbf{x} \in R^n$ into a new feature space $\phi(\mathbf{x})$.
- $\phi(x)$ is called the mapping function.
- For example, $\phi(x_1, x_2) = x_1^2 + x_2^2$



Kernel trick

- One more reason to use SVM: Training and predicting only depend on $\mathbf{x}_i^T \mathbf{x}_j$

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \tag{17}$$

For a new example $\mathbf{x}^*$:

$$\mathbf{w}^T \mathbf{x}^* = \sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x}^* \tag{18}$$

- We don't have to specify $\phi(\mathbf{x})$ *explicitly* as long as we can calculate $k(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2)$.

Polynomial kernel

- $\phi(\mathbf{x})$ can be infinite! Choosing the right kernel is an art and also problem dependent.
- Polynomial kernel of degree d: $k(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^T \mathbf{x}_2 + 1)^d$
- The space is very high dimensional:

$$\phi(\mathbf{x}) = \begin{pmatrix} \dots \\ x_1^{i_1} x_2^{i_2} \dots x_n^{i_n} \\ \dots \end{pmatrix} \tag{19}$$

  with $\sum_j i_j \leq d$.
- This allows modeling feature conjunctions up to the order of the polynomial.

Gaussian kernel

- $$k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\frac{(\mathbf{x}_1 - \mathbf{x}_2)^T(\mathbf{x}_1 - \mathbf{x}_2)}{2\sigma^2}) \quad (20)$$
- σ^2 is chosen to model the separation between classes.
- This allows modeling Gaussian-like decision boundary. Very popular in real world data.

String kernel

- Applications in bioinformatics: gene sequence, protein sequence,..
- String kernels usually count the number of matches between two strings, where the definition of a match varies from kernel to kernel.
- For ex, the *p-spectrum kernel*:
- It is a count of how many contiguous substrings of length p the two strings have in common.

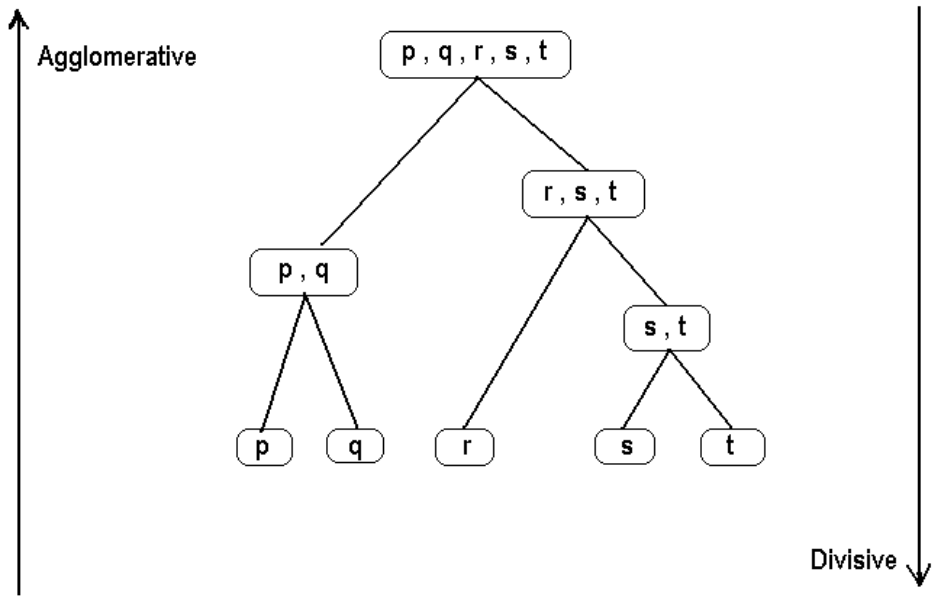
More kernels

- Time series kernel.
- Wavelet kernel.
- ANOVA kernel.

Beyond the scope of this class.
As always, there is an active area called kernel learning where a kernel is learnt to improve classification or regression accuracy.

Hierarchical clustering

We want to find a binary tree that "explains" the similarity between objects.



Hierarchical clustering

- We need a similarity function to compare objects.

  For ex: Euclidean distance, Pearson correlation, cosine correlation,...
- We also need a similarity function to compare clusters of objects.

  There are 3 type of linkages:
  - Complete linkage: distance of the two farthest points.
  - Single linkage: shortest distance between any two points.
  - Average linkage: averaged distances between all points.

Agglomerative clustering

- A greedy algorithm.
- At each step, combine 2 *closest* clusters together.
- The height of each internal node = distance between two children of the node.
- You will implement single linkage for Problem set 4.