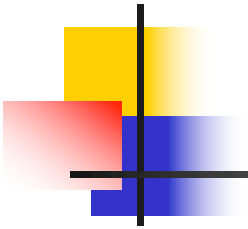


Machine Learning

Support Vector Machines

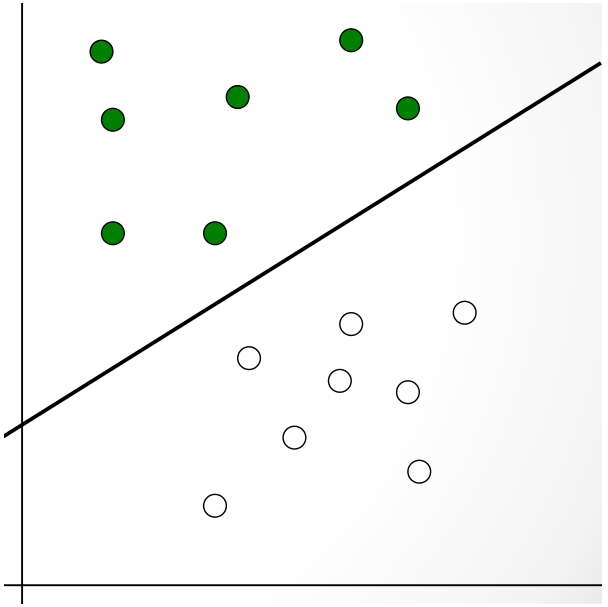
Support Vector Machines

- Both logistic regression and linear discriminant analysis learn a linear discriminant function to separate the data elements of two classes
  - Forms a hyperplane that separates the data points

$$h_\theta(x) = \theta^T x$$

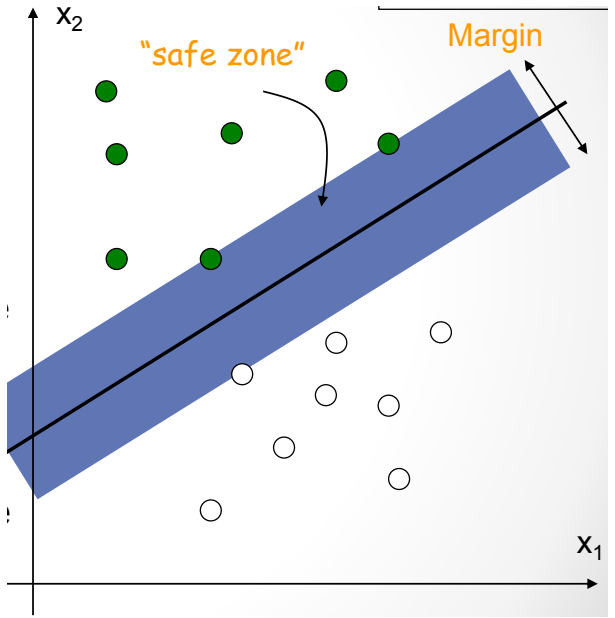
$$h_{w,b}(x) = w^T x + b$$

  - Generally infinite answers



Maximum Margin Classifier

- Classifiers with larger margins have stronger confidence in their predictions
  - Larger margins represent more tolerance to noise
  - Larger margins are more robust to outliers



Maximum Margin Classifier

- Learn a classifier that maximizes the margin
  - Assume classes are labeled 1 and -1

$$y^{(i)} = \begin{cases} 1 & if\ x^{(i)}\ is\ in\ class\ 1 \\ -1 & otherwise \end{cases}$$

  - How do we measure the margin of a classifier ?
    - Functional margin

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + b)$$

      - Has scaling problems if used with a logistic function
    - Geometric margin

$$\gamma^{(i)} = y^{(i)}\left(\frac{w^T x^{(i)}}{\|w\|} + \frac{b}{\|w\|}\right)$$

Maximum Margin Classifier

- From the margin of a data point we can formulate the margin of the classifier on a training set

$$\gamma = \min_i \gamma^{(i)}$$

- Margin provides a new performance function

$$\begin{aligned} \max_{\gamma,w,b} \quad & \gamma \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq \gamma, \quad i = 1, \ldots, n \\ & ||w|| = 1. \end{aligned}$$

  - Constrained optimization problem

Maximum Margin Classifier

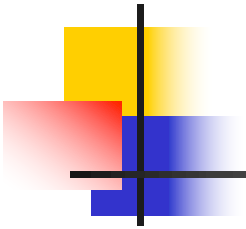
- Non-convex constraints are difficult so we reformulate this to

$$\max_{\gamma,w,b} \quad \frac{\hat{\gamma}}{||w||}$$
$$\text{s.t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq \hat{\gamma}, \quad i = 1, \ldots, n$$

  - Since we can scale functional margins, we can make it 1 and replace maximizing 1/||x|| with minimizing $||x||^2$

$$\min_{\gamma,w,b} \quad \frac{1}{2}||w||^2$$
$$\text{s.t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \ldots, n$$

- Solvable using Quadratic Programming



Constrained Optimization

- Constrained optimization problems are harder to solve than the previous unconstrained ones
  - Could solve the previous problem using Quadratic Programming
  - There is a more efficient solution for this formulation that requires some optimization background

Lagrange Multipliers

- Lagrange multipliers allow the transfer of some constrained optimization problems into unconstrained ones
- Consider a constrained optimization problem with equality constraints

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

  - There is an unconstrained Lagrangian version

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

    - Stationary point of this is necessary for original optimum

Lagrange Duality

- Lagrange duality extends this under some conditions to solving optimization problems with inequalities

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

  - From this we can formulate 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 relation to the original problem is more complex

Lagrange Duality

- Lagrange primal

$$\theta_{\mathcal{P}}(w) = \max_{\alpha,\beta\,:\,\alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta)$$

  - The Lagrange primal has interesting properties

$$\theta_{\mathcal{P}}(w) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{otherwise.} \end{cases}$$

  - Minimizing the primal leads to the same result as the original problem

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

Lagrange Duality

- Lagrange dual

$$\theta_{\mathcal{D}}(\alpha, \beta) = \min_{w} \mathcal{L}(w, \alpha, \beta)$$

  - Maximizing the dual has an interesting relation to minimizing the primal

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

    - And under certain conditions we get

$$d^* = p^*$$

      - Equality is fulfilled if

$$\mathcal{L}(w^*, \alpha^*, \beta^*) = \min_{w} \mathcal{L}(w, \alpha^*, \beta^*) = \max_{\alpha, \beta : \alpha_i \geq 0} \mathcal{L}(w^*, \alpha, \beta)$$

Lagrange Duality

- If $g_i(w)$ are convex and strictly feasible and $h_i(x)$ are affine we get the Karush-Kuhn-Tucker (KKT) conditions we can pick
  - Solution w^* solves the primal and we get

$$\frac{\partial}{\partial \beta_i}\mathcal{L}(w^*, \alpha^*, \beta^*) = 0, \quad i = 1, \ldots, l$$

$$g_i(w^*) \leq 0, \quad i = 1, \ldots, k$$

  - α^* and β^* solve the dual and we get

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

$$\alpha^* \geq 0, \quad i = 1, \ldots, k$$

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

Lagrange Duality

- From these we can see that in this case we also get our equality condition $d^* = p^*$
  - The dual complementarity condition provides additional insights

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

    - For all active inequality constraints (i.e. constraints where $\alpha_i^* > 0$) we have

$$g_i(w^*) = 0$$

      - Correspond to points that lie on the margin (support vectors)
        - Number of support vectors is often much smaller than the number of data points
        - Minimum is $d+1$ for d dimensional input space

Maximum Margin Classifier

- Applying this to our maximum margin problem

$$\min_{\gamma,w,b} \quad \frac{1}{2}||w||^2$$
$$\text{s.t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \ldots, m$$

  - We can rewrite the constraints as

$$g_i(w) = -y^{(i)}(w^T x^{(i)} + b) + 1 \leq 0$$

  - And get a Lagrangian

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

Maximum Margin Classifier

- Using the inner function of the dual we can derive functions for w^* and b^*

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

$$\frac{\partial}{\partial b} \mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

  - Placing them into the Lagrangian results in

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

    - This is a function in only the parameters α_i on the sum of inner products of the data points

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

Maximum Margin Classifier

- From this we get the Lagrangian dual problem as

$$\begin{aligned} \max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle. \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned}$$

  - This problem needs to be solved for α^*

Maximum Margin Classifier

- Once solved for α^* the remaining parameters can be computed
  - w^* can be computed from the inner dual function

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$

  - b^* can be computed from the primal problem

$$b^* = -\frac{\max_{i:y^{(i)}=-1} w^{*T} x^{(i)} + \min_{i:y^{(i)}=1} w^{*T} x^{(i)}}{2}$$

- Once this is solved we can make predictions by evaluating the point against the line

$$w^T x + b = \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b$$

Sequential Minimal Optimization

- An efficient way to solve for α^* *is SMO*
  - Iteratively optimizes over pairs of parameters where at least one violates the conditions, keeping the other parameters fixed
    - From the constraints we get for α_1 and α_2

$$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = -\sum_{i=3}^{m} \alpha_i y^{(i)} = \zeta$$

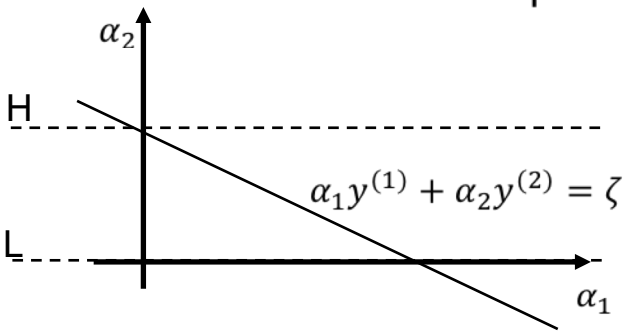
    - We can rewrite *W(α)* as

$$W(\alpha_1, \alpha_2, \ldots, \alpha_m) = W((\zeta - \alpha_2 y^{(2)}) y^{(1)}, \alpha_2, \ldots, \alpha_m)$$

      - Using $\alpha_2^{new,\bar{u}nclipped}$ as the unconstrained optimization result and considering that both α_1 and α_2 have to be at least 0 (limiting the parameter line to the upper right quadrant)

Sequential Minimal Optimization

- Since both α_1 and α_2 have to be non-negative, we can establish bounds L and H for the line to keep it in the right upper quadrant



- Clipping now moves α_2 to the closes legal point

$$\alpha_2^{new} = \begin{cases} H & \text{if } \alpha_2^{new,unclipped} > H \\ \alpha_2^{new,unclipped} & \text{if } L \leq \alpha_2^{new,unclipped} \leq H \\ L & \text{if } \alpha_2^{new,unclipped} < L \end{cases}$$

- The efficiency of SMO depends in part on how well the pairs of parameters α_1 and α_2 can be picked
  - Once all parameters α fulfill the KKT conditions, a solution is found

Support Vector Machines

- Everything so far required linearly separable date to make the constraints feasible
  - To make it practical we need to make it able to deal non-separable data and with outliers
    - Introduce a regularization term to deal with constraint violations, resulting in a modified version of the original

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m \\ & \xi_i \geq 0, \quad i = 1, \ldots, m. \end{aligned}$$

      - This added a set of additional inequality constraints

Support Vector Machines

- This results in a Lagrangian

$$\mathcal{L}(w, b, \xi, \alpha, r) = \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i \left[y^{(i)} (x^T w + b) - 1 + \xi_i \right] - \sum_{i=1}^{m} r_i \xi_i$$

  - And slightly modified dual and KKT constraints

$$\max_\alpha \quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

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

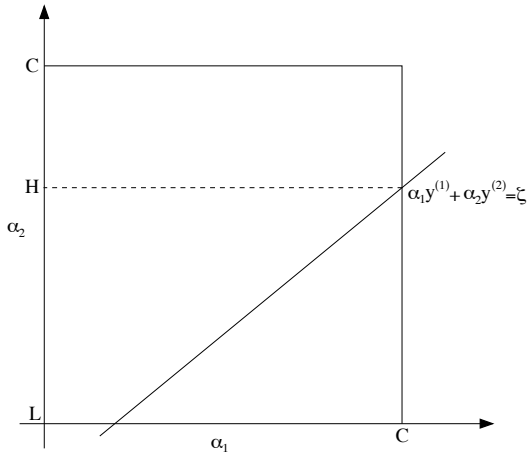
$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0,$$

$$\begin{aligned} \alpha_i = 0 \quad &\Rightarrow \quad y^{(i)} (w^T x^{(i)} + b) \ge 1 \\ \alpha_i = C \quad &\Rightarrow \quad y^{(i)} (w^T x^{(i)} + b) \le 1 \\ 0 < \alpha_i < C \quad &\Rightarrow \quad y^{(i)} (w^T x^{(i)} + b) = 1 \end{aligned}$$

    - Has the concept of active constraints (support vectors)

Support Vector Machines

- The optimization problem is still solvable using the SMO algorithm
  - Clipping now requires α_1 and α_2 to lie within a box spanned by 0 and C



Kernels and the "Kernel Trick"

- So far we have strictly dealt with linear Support Vector Machines
  - Can address this using non-linear features
    - Since in the entire algorithm for SVM only uses x in inner products $\langle x^{(i)}, x^{(j)} \rangle$ we can apply the "kernel trick"
      - A kernel function is any function of the form
        $$K(x, z) = \phi(x)^T \phi(z)$$
      - The "kernel trick" refers to the fact that in any algorithms that only uses the inner product of the data items we can replace the inner product of the features with the kernel function
        - Under certain conditions the value of a kernel function is significantly cheaper than computing the feature vector allowing higher dimensional feature spaces to be used

Kernels and the "Kernel Trick"

- Example Kernels
  - Square kernel $K(x,z) = (x^T z)^2 = \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) = \phi(x)^T \phi(z)$
    - Represents a n^2-dimensional feature vector

$$\phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix}$$

    - *K(x,z)* can be computed in *O(n)* time
  - Similar applies to general polynomial kernels

$$K(x,z) = (x^T z + c)^d$$

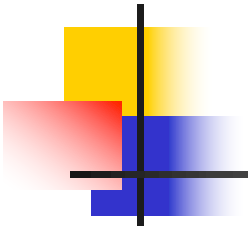
Kernels and the "Kernel Trick"

- Gaussian kernel $K(x,z) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$
  - Corresponds to a feature vector of infinite dimension
- A Kernel matrix is the square matrix of values of the kernel function applied to all pairs of data points

$$K_{ij} = K(x^{(i)}, x^{(j)})$$

  - A function is a valid (Mercer) kernel function if and only if for any data set the Kernel matrix is
    - Symmetric: $K_{i,j} = K_{j,i}$
    - Positive semi-definite $z^T K z \geq 0$

Kernel-Base Support Vector Machines



- The "kernel trick" can be applied to any algorithm that uses the data only in the form of an inner product
- The kernel trick allows SVMs to operate efficiently in very high-dimensional non-linear feature spaces expressed by appropriate kernel functions
  - Together with appropriate heuristics to make the SMO algorithm efficient, this makes SVMs very efficient non-linear classifiers