Constrained optimization

A general constrained optimization problem has the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \leq 0 \qquad i = 1, \ldots, m \\ & h_i(\mathbf{x}) = 0 \qquad i = 1, \ldots, p \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^d$

The Lagrangian function is given by

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) := f(\mathbf{x}) + \underbrace{\sum_{i=1}^{m} \lambda_i g_i(\mathbf{x})}_{\lambda_i \geq 0} + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x})$$

Primal and dual optimization problems

Primal: $\min_{\mathbf{x}} \max_{\boldsymbol{\lambda},\boldsymbol{\nu}:\lambda_i\geq 0} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})$ ← same solution as constrained version

Dual: $\max_{\boldsymbol{\lambda},\boldsymbol{\nu}:\lambda_i\geq 0} \min_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})$

Weak duality:
$$d^* := \max_{\boldsymbol{\lambda},\boldsymbol{\nu}:\lambda_i\geq 0} \min_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq \min_{\mathbf{x}} \max_{\boldsymbol{\lambda},\boldsymbol{\nu}:\lambda_i\geq 0} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) =: p^*$$

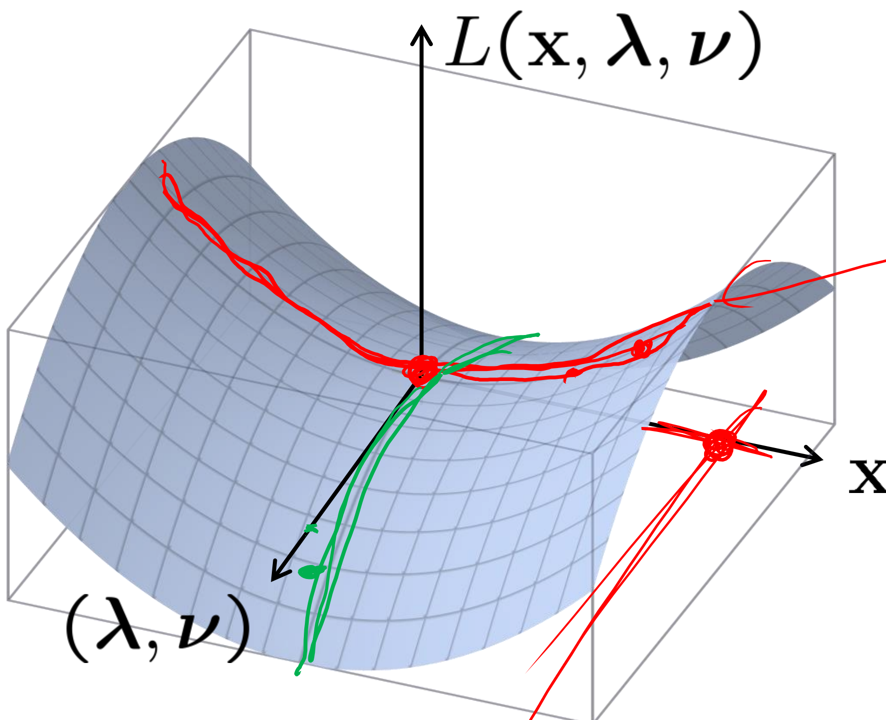
Strong duality: For convex problems with affine constraints
$$d^* = p^*$$

Saddle point property

If $(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$ are primal/dual optimal with zero duality gap, they are a ***saddle point*** of $L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})$, i.e.,

$$L(\mathbf{x}^*, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq L(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*) \leq L(\mathbf{x}, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$$

for all $\mathbf{x} \in \mathbb{R}^d, \ \boldsymbol{\lambda} \in \mathbb{R}^m_+, \ \boldsymbol{\nu} \in \mathbb{R}^p$



$\underline{\max_{\lambda} L(x, \lambda)}$

KKT conditions: The bottom line

If a constrained optimization problem is

- differentiable
- convex

then the KKT conditions are necessary and sufficient for primal/dual optimality (with zero duality gap)

In this case, we can use the KKT conditions to find a solution to our optimization problem

i.e., if we find $(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$ satisfying the conditions, we have found solutions to both the primal and dual problems

The KKT conditions

1. $\nabla f(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(\mathbf{x}^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(\mathbf{x}^*) = 0$

2. $g_i(\mathbf{x}^*) \leq 0, \quad i = 1, \ldots, m$

3. $h_i(\mathbf{x}^*) = 0, \quad i = 1, \ldots, p$

4. $\lambda_i^* \geq 0, \quad i = 1, \ldots, m$

5. $\lambda_i^* g_i(\mathbf{x}^*) = 0 \quad i = 1, \ldots, m$

   (complementary slackness)

Soft-margin classifier

$$\min_{\mathbf{w},b,\xi} \ \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \ \ y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, \dots, n$$

$$\xi_i \geq 0 \quad i = 1, \dots, n$$

This optimization problem is differentiable and convex

- the KKT conditions and necessary and sufficient conditions for primal/dual optimality (with zero duality gap)
- we can use these conditions to find a relationship between the solutions of the primal and dual problems
- the dual optimization problem will be easy to "kernelize"

Forming the Lagrangian

Begin by converting our problem to the standard form

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \ \ y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \qquad i = 1, \ldots, n$$

$$\xi_i \geq 0 \qquad i = 1, \ldots, n$$

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

2n

$$\text{s.t.} \ \ 1 - \xi_i - y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq 0 \qquad i = 1, \ldots, n$$

$$-\xi_i \leq 0 \qquad i = 1, \ldots, n$$

Forming the Lagrangian

The Lagrangian function is then given by

Lagrange multipliers/dual variables

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$+ \sum_{i=1}^{n}\alpha_i\underbrace{(1 - \xi_i - y_i(\mathbf{w}^T\mathbf{x}_i + b))}_{g_i(x)} - \sum_{i=1}^{n}\beta_i\underbrace{\xi_i}_{g_i(x)}$$

Soft-margin dual

The Lagrangian dual is thus

$$L_D(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \min_{\mathbf{w}, b, \boldsymbol{\xi}} L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

and the dual optimization problem is

$$\max_{\boldsymbol{\alpha}, \boldsymbol{\beta} : \alpha_i, \beta_i \geq 0} L_D(\boldsymbol{\alpha}, \boldsymbol{\beta})$$

Let's compute a simplified expression for $L_D(\boldsymbol{\alpha}, \boldsymbol{\beta})$

How?

Using the KKT conditions!

Taking the gradient

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$+ \sum_{i=1}^{n}\alpha_i(1 - \xi_i - y_i(\mathbf{w}^T\mathbf{x}_i + \cancel{b})) - \sum_{i=1}^{n}\beta_i\xi_i$$

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathbf{w} - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i = 0 \Rightarrow w = \sum_i \alpha_i y_i x_i$$

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = -\sum_{i=1}^{n}\alpha_i y_i = 0$$

$$\frac{\partial}{\partial \xi_i} L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{C}{n} - \alpha_i - \beta_i = 0$$

Plugging this in

The dual function is thus

$$L_D(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_i \alpha_i$$

And the dual optimization problem can be written as

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \quad -\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_i \alpha_i$$

$$\text{s.t.} \quad \sum_i \alpha_i y_i = 0$$

$$0 \le \alpha_i \le \frac{C}{n}$$

$$\alpha_i + \beta_i = \frac{C}{n} \qquad \alpha_i, \beta_i \ge 0 \qquad i = 1, \ldots, n$$

Soft-margin dual quadratic program

We can eliminate β to obtain

$$\max_{\boldsymbol{\alpha}} \quad -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j + \sum_i \alpha_i$$

$$\text{s.t. } \sum_i \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq \frac{C}{n} \qquad i = 1,\ldots,n$$

$\langle w^*, x\rangle + b^*$

box constraints

Note: Input patterns are only involved via ***inner products***

Recovering $\mathbf{w}^*$

Given $\boldsymbol{\alpha}^*$ (the solution to the soft-margin dual), can we recover the optimal $\mathbf{w}^*$ and b^*?

Yes! Use the KKT conditions

From KKT condition 1, we know that

$$\mathbf{w}^* - \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i = 0$$

And thus the optimal normal vector is just a linear combination of our input patterns

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i$$

b^* is a little less obvious – we'll return to this in a minute

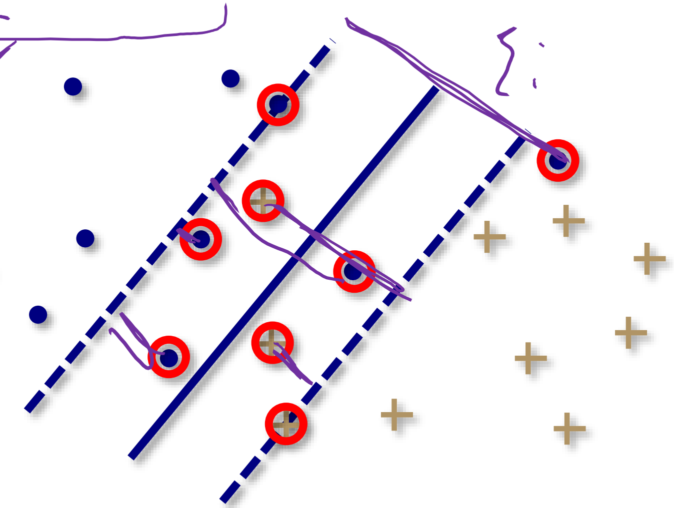
Support vectors

From KKT condition 5 (complementary slackness) we also have that for all i,

$$\alpha_i^* \underbrace{\left(1 - \xi_i^* - y_i\left(\mathbf{w}^{*T}\mathbf{x}_i + b^*\right)\right)}_{\neq 0} = 0$$

The $\mathbf{x}_i$ for which $y_i(\mathbf{w}^{*T}\mathbf{x}_i + b^*) = 1 - \xi_i^*$ are called ***support vectors***

These are the points on or inside the margin of separation



Useful fact:

By the KKT conditions, $\alpha_i^* \neq 0$ if and only if $\mathbf{x}_i$ is a support vector!

Empirical fact

It has been widely demonstrated (empirically) that in typical learning problems, only a small fraction of the training input patterns are support vectors

Thus, support vector machines produce a hyperplane with a ***sparse*** representation

$$\mathbf{w}^* = \sum_{\text{support vectors}} \alpha_i^* y_i \mathbf{x}_i$$

This is advantageous for efficient storage and evaluation

What about b^*?

Another consequence of the KKT conditions (condition 5) is that for all i, $\beta_i^* \xi_i^* = 0$

Since $\alpha_i^* + \beta_i^* = \frac{C}{n}$, this implies that if $\alpha_i^* < \frac{C}{n}$, then $\xi_i^* = 0$

Recall that if $\alpha_i^* > 0$ we also have that $\mathbf{x}_i$ is a support vector, and hence

$$y_i({\mathbf{w}^*}^T \mathbf{x}_i + b^*) = 1 - \xi_i^*$$

How can we combine these two facts to determine b^* ?

Recovering b^*

For any i such that $0 < \alpha_i^* < \frac{C}{n}$, we have

$$y_i(\mathbf{w}^{*T}\mathbf{x}_i + b^*) = 1$$

$$\Downarrow$$

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

In practice, it is common to average over several such i to counter numerical imprecision

Support vector machines

Given an inner product kernel k, we can write the SVM classifier as

$$\widehat{h}(\mathbf{x}) = \text{sign}\left(\sum_i \alpha_i^* y_i k(\mathbf{x}, \mathbf{x}_i) + b^*\right)$$

where $\boldsymbol{\alpha}^*$ is the solution of

$$\max_{\boldsymbol{\alpha}} \quad -\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + \sum_i \alpha_i$$

$$\text{s.t.} \quad \sum_i \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq \frac{C}{n} \quad i = 1, \ldots, n$$

and $b^* = y_i - \sum_j \alpha_j^* y_j k(\mathbf{x}_i, \mathbf{x}_j)$ for some i s.t. $0 < \alpha_i^* < \frac{C}{n}$

Remarks

- The final classifier depends only on the $\mathbf{x}_i$ with $\alpha_i > 0$, i.e., the ***support vectors***
- The size (number of variables) of the dual QP is n, independent of the kernel k, the mapping Φ, or the space $\mathcal{F}$
  - remarkable, since the dimension of $\mathcal{F}$ can be ***infinite***
- The soft-margin hyperplane was the first machine learning algorithm to be “kernelized”, but since then the idea has been applied to many, many other algorithms
  - kernel ridge regression
  - kernel PCA
  - …

Solving the quadratic program

How can we actually compute the solution to

$$\max_{\alpha} \quad -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j q_{ij} + \sum_i \alpha_i$$

$$\text{s.t.} \quad \sum_i \alpha_i y_i = 0, \quad 0 \le \alpha_i \le \frac{C}{n} \quad i = 1, \ldots, n$$

where $q_{ij} := y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$?

There are several general approaches to soling quadratic programs, and many can be applied to solve the SVM dual

We will focus on a particular example that is very efficient and capitalizes on some of the unique structure in the SVM dual, called ***sequential minimal optimization (SMO)***

Sequential minimal optimization

SMO is an example of a ***decomposition*** algorithm

Sequential minimal optimization

Initialize: $\boldsymbol{\alpha} = 0$

Repeat until stopping criteria satisfied

(1) Select a pair $i, j, 1 \leq i, j \leq n$

(2) Update α_i and α_j by optimizing the dual QP, holding all other $\alpha_k, k \neq i, j$ fixed

The reason for decomposing this to a two-variable subproblem is that this subproblem can be solved ***exactly*** via a simple ***analytic*** update

The update step

Choose α_i and α_j to solve

$$\max_{\alpha_i, \alpha_j} \quad -\frac{1}{2}\left(\alpha_i^2 q_{ii} + \alpha_j^2 q_{jj} + 2\alpha_i\alpha_j q_{ij}\right) + c_i\alpha_i + c_j\alpha_j$$

$$\text{s.t. } \alpha_i y_i + \alpha_j y_j = -\sum_{k \neq i,j} \alpha_k y_k$$

$$0 \leq \alpha_i, \alpha_j \leq \frac{C}{n}$$

where $c_i = 1 - \frac{1}{2}\sum_{k \neq i,j} \alpha_k q_{ik}$ and similarly for c_j

SMO in practice

- Several strategies have been proposed for selecting (i, j) at each iteration
- Typically based on heuristics (often using the KKT conditions) that predict which pair of variables will lead to the largest change in the objective function
- For many of these heuristics, the SMO algorithm is proven to converge to the global optimum after finitely many iterations
- The running time is $O(n^3)$ in the worst case, but tends to be more like $O(n^2)$ in practice

Alternative algorithms

SMO is one of the predominant strategies for training an SVM, but there are important alternatives to consider on very large datasets

- modern variants for solving the dual based on stochastic gradient descent
  - closely related to SMO
- directly optimizing the primal
  - makes most sense when the dimension of the feature space is small compared to the size of the dataset
  - some algorithms very similar to PLA and stochastic gradient descent version of logistic regression