Partial derivative

In mathematics, a partial derivative of a function of several variables is its derivative with respect to one of those variables with the others held constant (as opposed to the total derivative, in which all variables are allowed to vary). Partial derivatives are useful in vector calculus and differential geometry.
The partial derivative of a function f with respect to the variable x is written as fx, ∂xf, or ∂f/∂x. The partial-derivative symbol ∂ is a rounded letter, distinguished from the straight d of total-derivative notation. The notation was introduced by Legendre and gained general acceptance after its reintroduction by Jacobi.

In general, the partial derivative of a function f(x1,...,xn) in the direction xi at the point (a1,...,an) is defined to be:

$$\frac{\partial f}{\partial x_i}(a_1,\ldots,a_n) = \lim_{h\to 0}\frac{f(a_1,\ldots,a_i+h,\ldots,a_n) - f(a_1,\ldots,a_n)}{h}.$$

∂ is a rounded d called the partial derivative symbol. To distinguish it from the letter d, ∂ is sometimes pronounced "der", "del", "dah", or "partial" instead of "dee".

$$f(x,y) = x^2 + xy + ay^2 + b \qquad \frac{\partial f}{\partial x} = \qquad \frac{\partial f}{\partial y} =$$

Gradient

The gradient (or gradient vector field) of a scalar function *f*(*x*) with respect to a vector variable $x = (x_1,\ldots,x_n)$ is denoted by ∇f or $\vec{\nabla} f$ where ∇ (the nabla symbol) denotes the vector differential operator, del. The notation $\mathrm{grad}(f)$ is also used for the gradient. The gradient of *f* is defined to be the vector field whose components are the partial derivatives of *f*. That is:

$$\nabla f = \left(\frac{\partial f}{\partial x_1},\ldots,\frac{\partial f}{\partial x_n}\right).$$

(from Wikipedia, modified by kameda 2008/09/08)

Lagrange multipliers

From Wikipedia, the free encyclopedia (rearranged by kameda, 2008/09/08)

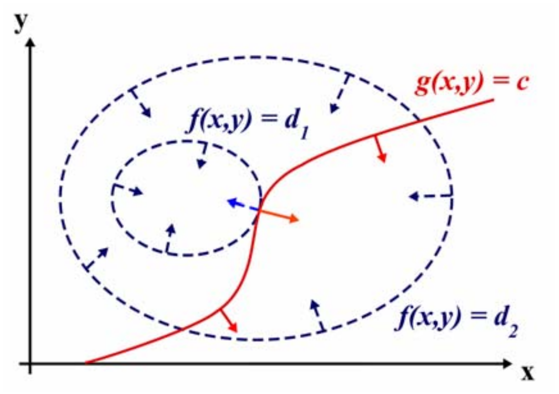


Fig. 1

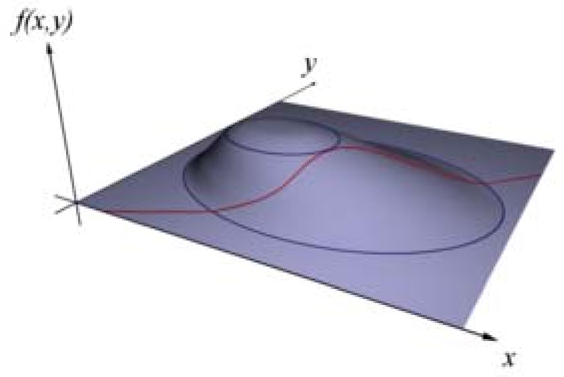


Fig. 2

Fig. 1. Drawn in red is the locus (contour) of points satisfying the constraint $g(x,y) = c$. Drawn in blue are contours of f. Arrows represent the gradient, which points in a direction normal to the contour.
Fig. 2. Same problem as above as three-dimensional visualization, assuming that we want to maximize $f(x,y)$.

In mathematical optimization problems, the method of **Lagrange multipliers**, named after Joseph Louis Lagrange, is a method for finding the extrema of a function of several variables subject to one or more constraints; it is the basic tool in nonlinear constrained optimization.

Simply put, the technique is able to determine where on a particular set of points (such as a circle, sphere, or plane) a particular function is the smallest (or largest).

More formally, Lagrange multipliers compute the stationary points of the constrained function. By Fermat's theorem, extrema occur either at these points, or on the boundary, or at points where the function is not differentiable.

It reduces finding stationary points of a constrained function in n variables with k constraints to finding stationary points of an unconstrained function in $n+k$ variables. The method introduces a new unknown scalar variable (called the Lagrange multiplier) for each constraint, and defines a new function (called the **Lagrangian**) in terms of the original function, the constraints, and the Lagrange multipliers.

Introduction

Consider a two-dimensional case. Suppose we have a function $f(x,y)$ we wish to maximize or minimize subject to the constraint

$$g(x,y) = c,$$

where *c* is a constant. We can visualize contours of *f* given by

$$f(x,y) = d_n$$

for various values of d_n, and the contour of *g* given by $g(x,y) = c$.

Suppose we walk along the contour line with $g = c$. In general the contour lines of *f* and *g* may be distinct, so traversing the contour line for $g = c$ could intersect with or cross the contour lines of *f*. This is equivalent to saying that while moving along the contour line for $g = c$ the value of *f* can vary. Only when the contour line for $g = c$ touches contour lines of *f* tangentially, we do not increase or decrease the value of *f* — that is, when the contour lines touch but do not cross.

This occurs exactly when the tangential component of the total derivative vanishes: $df_\parallel = 0$, which is at the constrained stationary points of *f* (which include the constrained local extrema, assuming *f* is differentiable). Computationally, this is when the gradient of *f* is normal to the constraint(s): when $\nabla f = \lambda \nabla g$ for some scalar λ (where ∇ is the gradient). Note that the constant λ is required because, even though the directions of both gradient vectors are equal, the magnitudes of the gradient vectors are most likely not equal.

A familiar example can be obtained from weather maps, with their contour lines for temperature and pressure: the constrained extrema will occur where the superposed maps show touching lines (isopleths).

Geometrically we translate the tangency condition to saying that the gradients of *f* and *g* are parallel vectors at the maximum, since the gradients are always normal to the contour lines. Thus we want points (x,y) where $g(x,y) = c$ and

$$\nabla_{x,y} f = \lambda \nabla_{x,y} g,$$

where

$$\nabla_{x,y} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right).$$

To incorporate these conditions into one equation, we introduce an auxiliary function

$$F(x,y,\lambda) = f(x,y) + \lambda \left(g(x,y) - c \right),$$

and solve

$$\nabla_{x,y,\lambda} F(x, y, \lambda) = 0.$$

Justification

As discussed above, we are looking for stationary points of *f* seen while travelling on the level set *g*(*x*,*y*) = *c*. This occurs just when the gradient of *f* has no component tangential to the level sets of *g*. This condition is equivalent to $\nabla_{x,y} f(x,y) = \lambda \nabla_{x,y} g(x,y)$ for some λ. Stationary points (*x*,*y*, λ) of *F* also satisfy *g*(*x*,*y*) = *c* as can be seen by considering the derivative with respect to λ.

Caveat: extrema versus stationary points

Be aware that the solutions are the ***stationary points*** of the Lagrangian *F*, and are saddle points: they are not necessarily *extrema* of *F*. *F* is unbounded: given a point (*x*,*y*) that doesn't lie on the constraint, letting $\lambda \to \pm\infty$ makes *F* arbitrarily large or small. However, under certain stronger assumptions, as we shall see below, the **strong Lagrangian principle** holds, which states that the maxima of *f* maximize the Lagrangian globally.

A more general formulation: The weak Lagrangian principle

Denote the objective function by $f(\mathbf{x})$ and let the constraints be given by $g_k(\mathbf{x}) = 0$, perhaps by moving constants to the left, as in $h_k(\mathbf{x}) - c_k = g_k(\mathbf{x})$. The domain of *f* should be an open set containing all points satisfying the constraints. Furthermore, *f* and the g_k must have continuous first partial derivatives and the gradients of the g_k must not be zero on the domain.[1] Now, define the Lagrangian, Λ, as

$$\Lambda(\mathbf{x}, \boldsymbol{\lambda}) = f + \sum_k \lambda_k g_k.$$

k is an index for variables and functions associated with a particular constraint, *k*.
$\boldsymbol{\lambda}$ without a subscript indicates the vector with elements λ_k, which are taken to be independent variables.

Observe that both the optimization criteria and constraints $g_k(x)$ are compactly encoded as stationary points of the Lagrangian:

$\nabla_{\mathbf{x}} \Lambda = \mathbf{0}$ if and only if $\nabla_{\mathbf{x}} f = -\sum_k \lambda_k \nabla_{\mathbf{x}} g_k,$

$\nabla_{\mathbf{x}}$ means to take the gradient only with respect to each element in the vector $\mathbf{x}$, instead of all variables.

and

$$\nabla_\lambda \Lambda = \mathbf{0}$$ implies $g_k = 0$.

Collectively, the stationary points of the Lagrangian,

$$\nabla \Lambda = \mathbf{0},$$

give a number of unique equations totaling the length of $\mathbf{x}$plus the length of λ.

Interpretation of λ_i

Often the Lagrange multipliers have an interpretation as some salient quantity of interest. To see why this might be the case, observe that:

$$\frac{\partial \Lambda}{\partial g_k} = \lambda_k.$$

So, λ_k is the rate of change of the quantity being optimized as a function of the constraint variable. As examples, in Lagrangian mechanics the equations of motion are derived by finding stationary points of the action, the time integral of the difference between kinetic and potential energy. Thus, the force on a particle due to a scalar potential, $F = -\nabla V$, can be interpreted as a Lagrange multiplier determining the change in action (transfer of potential to kinetic energy) following a variation in the particle's constrained trajectory. In economics, the optimal profit to a player is calculated subject to a constrained space of actions, where a Lagrange multiplier is the value of relaxing a given constraint (e.g. through bribery or other means).

Examples

Very simple example

Suppose you wish to maximize $f(x,y) = x + y$ subject to the constraint $x^2 + y^2 = 1$. The constraint is the unit circle, and the level sets of f are diagonal lines (with slope −1). Then, where is the maximum and how much is it?

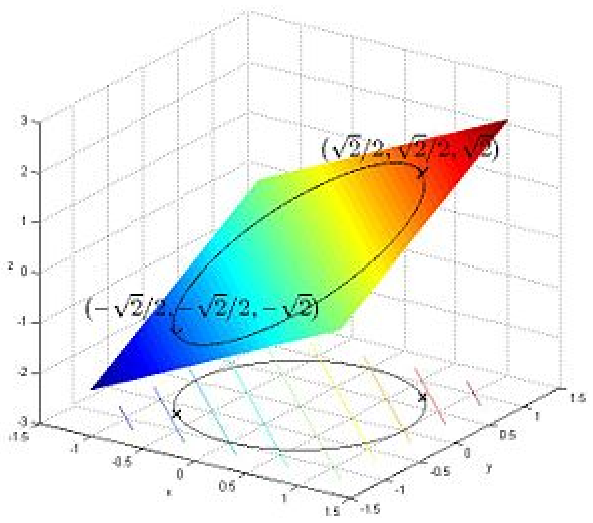


Fig. 3. Illustration of the constrained optimization problem.

Simple example

Suppose you want to find the maximum values for

$$f(x,y) = x^2 y$$

with the condition that the *x* and *y* coordinates lie on the circle around the origin with radius $\sqrt{3}$, that is,

$$x^2 + y^2 = 3.$$

Again, where is the maximum and how much is it?

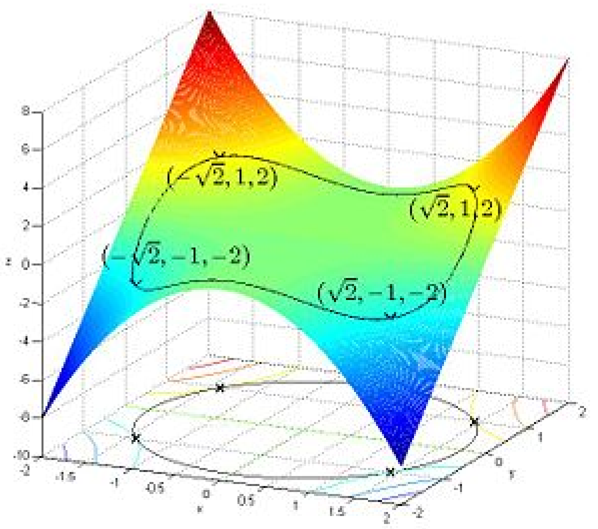


Fig. 4. Illustration of the constrained optimization problem.

Suppose you want to find the maximum values for

$$f(x,y) = x^2 y$$

with the condition that the x and y coordinates lie on the circle around the origin with radius $\sqrt{3}$, that is,

$$x^2 + y^2 = 3.$$

Again, where is the maximum and how much is it?

Example: entropy

Suppose we wish to find the discrete probability distribution with maximal information entropy. Then

$$f(p_1, p_2, \ldots, p_n) = -\sum_{k=1}^{n} p_k \log_2 p_k.$$

Of course, the sum of these probabilities equals 1, so our constraint is $g(\mathbf{p}) = 1$ with

$$g(p_1, p_2, \ldots, p_n) = \sum_{k=1}^{n} p_k.$$

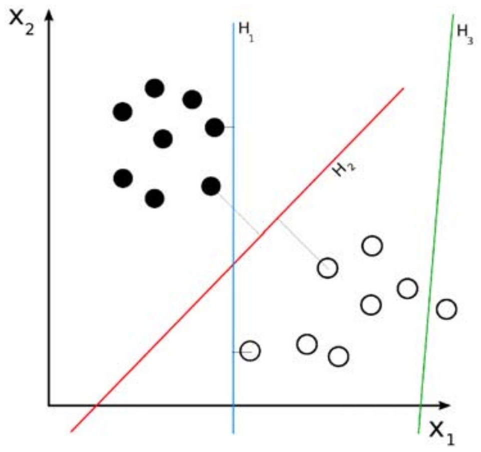
We can use Lagrange multipliers to find the point of maximum entropy (depending on the probabilities).

Support vector machine

From Wikipedia, the free encyclopedia (rearranged by kameda, 2008/09/08)

Support vector machines (**SVMs**) are a set of related supervised learning methods used for classification and regression. Viewing input data as two sets of vectors in an *n*-dimensional space, an SVM will construct a separating hyperplane in that space, one which maximizes the *margin* between the two data sets. To calculate the margin, two parallel hyperplanes are constructed, one on each side of the separating hyperplane, which are "pushed up against" the two data sets. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the neighboring datapoints of both classes, since in general the larger the margin the better the generalization error of the classifier.

Motivation



H3 doesn't separate the 2 classes. H1 does, with a small margin and H2 with the maximum margin.

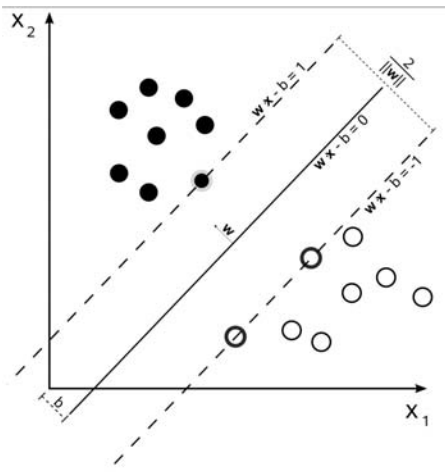
Classifying data is a common need in machine learning. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a *new* data point will be in. In the case of support vector machines, a data point is viewed as a *p*-dimensional vector (a list of *p* numbers), and we want to know whether we can separate such points with a $p - 1$-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. However, we are additionally interested in finding out if we can achieve maximum separation (margin) between the two classes. By this we mean that we pick the hyperplane so that the distance from the hyperplane to the nearest data point is maximized. That is to say that the nearest distance between a point in one *separated* hyperplane and a point in the other *separated* hyperplane is maximized. Now, if such a hyperplane exists, it is clearly of interest and is known as the *maximum-margin hyperplane* and such a linear classifier is known as a **maximum margin classifier**.

Formalization

We are given some training data, a set of points of the form

$$\mathcal{D} = \{(\mathbf{x}_i, c_i) | \mathbf{x}_i \in \mathbb{R}^p, c_i \in \{-1, 1\}\}_{i=1}^{n}$$

where the c_i is either 1 or −1, indicating the class to which the point $\mathbf{x}_i$ belongs. Each $\mathbf{x}_i$ is a p-dimensional real vector. We want to give the maximum-margin hyperplane which divides the points having $c_i = 1$ from those having $c_i = -1$. Any hyperplane can be written as the set of points $\mathbf{x}$ satisfying



Maximum-margin hyperplane and margins for a SVM trained with samples from two classes. Samples on the margin are called the support vectors.

$$\mathbf{w} \cdot \mathbf{x} - b = 0.$$

The vector $\mathbf{w}$ is a normal vector: it is perpendicular to the hyperplane. The parameter $\frac{b}{\|\mathbf{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$.

We want to choose the $\mathbf{w}$ and b to maximize the margin, or distance between the parallel hyperplanes that are as far apart as possible while still separating the data. These hyperplanes can be described by the equations

$$\mathbf{w} \cdot \mathbf{x} - b = 1 \text{ and } \mathbf{w} \cdot \mathbf{x} - b = -1.$$

Note that if the training data are linearly separable, we can select the two hyperplanes of the margin in a way that there are no points between them and then try to maximize their distance. By using geometry, we find the distance between these two hyperplanes is $\frac{2}{\|\mathbf{w}\|}$, so we want to minimize $\|\mathbf{w}\|$. As we also have to prevent data points falling into the margin, we add the following constraint: for each i either

$$\mathbf{w} \cdot \mathbf{x}_i - b \geq 1 \text{ for } \mathbf{x}_i \text{ for the first class or}$$

$\mathbf{w}\cdot\mathbf{x}_i - b \le -1$ for $\mathbf{x}_i$ of the second.

This can be rewritten as:

$$c_i(\mathbf{w}\cdot\mathbf{x}_i - b) \ge 1, \quad \text{for all } 1 \le i \le n. \qquad (1)$$

We can put this together to get the optimization problem:

choose $\mathbf{w}, b$ to minimize $\|\mathbf{w}\|$

subject to $c_i(\mathbf{w}\cdot\mathbf{x_i} - b) \ge 1, \quad \text{for all } 1 \le i \le n$

Primal Form

The optimization problem presented in the preceding section is hard because it depends on the absolute value of |w|. The reason being that it is a non-convex optimization problem, which are known to be much more difficult to solve than convex optimization problems. Fortunately it is possible to alter the equation by substituting ||w|| with $\frac{1}{2}\|\mathbf{w}\|^2$ without changing the solution (the minimum of the original and the modified equation have the same w and b). This is a quadratic programming (QP) optimization problem. More clearly,

minimize $\frac{1}{2}\|\mathbf{w}\|^2$, subject to $c_i(\mathbf{w}\cdot\mathbf{x_i} - b) \ge 1, \quad 1 \le i \le n$.

The factor of 1/2 is used for mathematical convenience. This problem can now be solved by standard quadratic programming techniques and programs.

Dual Form

Writing the classification rule in its unconstrained dual form reveals that the maximum margin hyperplane and therefore the classification task is only a function of the *support vectors*, the training data that lie on the margin. The dual of the SVM can be shown to be:

$$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j c_i c_j \mathbf{x}_i^T \mathbf{x}_j$$ subject to $\alpha_i \geq 0$, and

$$\sum_{i=1}^{n} \alpha_i c_i = 0$$

where the α terms constitute a dual representation for the weight vector in terms of the training set:

$$\mathbf{w} = \sum_i \alpha_i c_i \mathbf{x}_i$$

Extensions to the linear SVM

Soft margin

In 1995, Corinna Cortes and Vladimir Vapnik suggested a modified maximum margin idea that allows for mislabeled examples.[2] If there exists no hyperplane that can split the "yes" and "no" examples, the *Soft Margin* method will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. This work popularized the expression *Support Vector Machine* or *SVM*. The method introduces slack variables, ξ_i, which measure the degree of misclassification of the datum x_i

$$c_i(\mathbf{w}\cdot\mathbf{x_i} - b) \geq 1 - \xi_i \quad 1 \leq i \leq n \qquad (2).$$

The objective function is then increased by a function which penalises non-zero ξ_i, and the optimisation becomes a trade off between a large margin, and a small error penalty. If the penalty function is linear, the equation (3) now transforms to

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \quad \text{such that} \quad c_i(\mathbf{w}\cdot\mathbf{x_i} - b) \geq 1 - \xi_i \quad 1 \leq i \leq n.$$

This constraint in (2) along with the objective of minimizing $|\mathbf{w}|$ can be solved using Lagrange multipliers. The key advantage of a linear penalty function is that the slack variables vanish from the dual problem, with the constant *C* appearing only as an additional constraint on the Lagrange multipliers. Non-linear penalty functions have been used, particularly to reduce the effect of outliers on the classifier, but unless care is taken, the problem becomes non-convex, and thus it is considerably more difficult to find a global solution.

Non-linear classification

The original optimal hyperplane algorithm proposed by Vladimir Vapnik in 1963 was a linear classifier. However, in 1992, Bernhard Boser, Isabelle Guyon and Vapnik suggested a way to create non-linear classifiers by applying the kernel trick (originally proposed by Aizerman et al..[3]) to maximum-margin hyperplanes.[4] The resulting algorithm is formally similar, except that every dot product is replaced by a non-linear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in the transformed feature space. The transformation may be non-linear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space it may be non-linear in the original input space.

If the kernel used is a Gaussian radial basis function, the corresponding feature space is a Hilbert space of infinite dimension. Maximum margin classifiers are well regularized, so the infinite dimension does not spoil the results. Some common kernels include,

- Polynomial (homogeneous): $k(\mathbf{x},\mathbf{x}') = (\mathbf{x}\cdot\mathbf{x}')^d$
- Polynomial (inhomogeneous): $k(\mathbf{x},\mathbf{x}') = (\mathbf{x}\cdot\mathbf{x}' + 1)^d$
- Radial Basis Function: $k(\mathbf{x},\mathbf{x}') = \exp(-\gamma\|\mathbf{x}-\mathbf{x}'\|^2)$, for $\gamma > 0$
- Gaussian Radial basis function: $k(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$
- Sigmoid: $k(\mathbf{x},\mathbf{x}') = \tanh(\kappa\mathbf{x}\cdot\mathbf{x}' + c)$, for some (not every) $\kappa > 0$ and $c < 0$