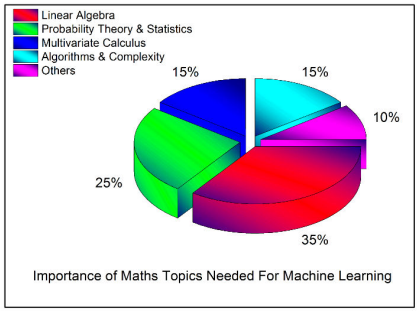
Optimization for Data Analytics

Abdullah Daşcı

8 Eylül 2019

The Mathematics of Machine Learning



From https://towardsdatascience.com/the-mathematics-of-machine-learning-894f046c568

Linear Algebra

- Matrix Operations
- Projections
- Eigenvalues & Eigenvectors
- Vector Spaces and Norms
- LU Decomposition
- QR Decomposition/Factorization
- Symmetric Matrices
- Orthogonalization & Orthonormalization
- Principal Component Analysis (PCA)
- Singular Value Decomposition (SVD)
- Eigendecomposition

Sabancı Üniversitesi VBYO 2019

Probability and Statistics

- Combinatorics
- Probability Rules & Axioms
- Bayes' Theorem
- Random Variables
- Variance and Expectation
- Conditional and Joint Distributions
- Standard Distributions (Bernoulli, Binomial, Uniform and Gaussian)
- Moment Generating Functions
- Maximum Likelihood Estimation (MLE)
- Prior and Posterior
- Maximum a Posteriori Estimation (MAP)
- Sampling Methods
- Bayesian Statistics

Multivariate Calculus

- Differential and Integral Calculus
- Partial Derivatives
- Vector-Valued Functions
- Directional Gradient
- Hessian
- Jacobian
- Lagrangian
- KKT Conditions

Sabancı Üniversitesi VBYO 2019

Algorithms and Complexity

- Knowledge of data structures (Binary Trees, Hashing, Heap, Stacking, etc),
- Integer Programming
- Nonlinear Programming
- Randomized & Sublinear Algorithm
- Graphs, Gradient/Stochastic Descents
- Primal-Dual methods
- Heuristics
- Metaheuristics
- Mathheuristics

Others

- Real and Complex Analysis
  - Sets and Sequences
  - Topology
  - Metric Spaces
  - Single-Valued and Continuous Functions
  - Limits
  - Cauchy Kernel
  - Fourier Transforms
- Information Theory
  - Entropy
  - Information Gain
- Function Spaces
- Manifolds

Sabancı Üniversitesi VBYO 2019

Concepts and Tools

Concepts

- Regression
  - Simple linear
  - Multiple
- Regularization
- Classification
  - Logistic Regression
  - Support Vector Machines
- Clustering

Tools

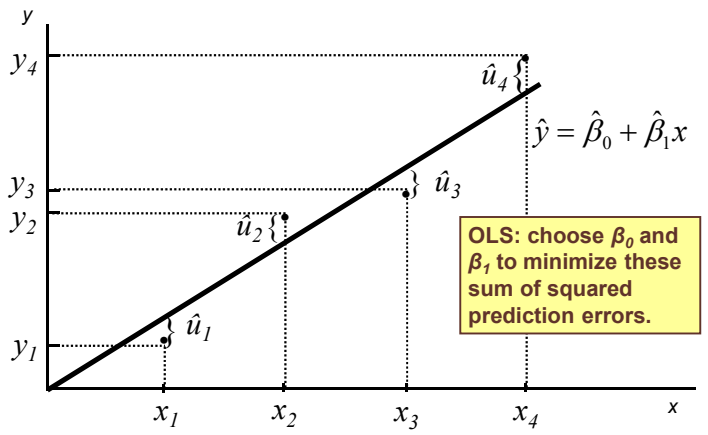
- Basic Linear Algebra
- Multivariate Calculus
- Probability/Statistics
- Unconstrained Optimization
- Constrained Optimization
- Metaheuristics
- Integer Programming

Sabancı Üniversitesi VBYO 2019

Linear Regression

Sabancı Üniversitesi VBYO 2019

Ordinary Least Squares (OLS) of Regression



10

Minimizing Residual Sum of Squares (RSS)

$$\min_{\hat{\beta}_0,\hat{\beta}_1} \sum_{i=1}^{n} (\hat{u}_i)^2 = \min_{\hat{\beta}_0,\hat{\beta}_1} \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i\right)^2$$

First order conditions (*recall the chain rule – Note the derivative with respect to beta's*):

Minimizing Residual Sum of Squares

$$\min_{\hat{\beta}_0,\hat{\beta}_1} \sum_{i=1}^{n} (\hat{u}_i)^2 = \min_{\hat{\beta}_0,\hat{\beta}_1} \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i\right)^2$$

First order conditions (*recall the chain rule – Note the derivative with respect to beta's*):

If $h(x) = g(f(x))$, then $h'(x) = g'(f(x))f'(x)$.

Minimizing Residual Sum of Squares

$$\min_{\hat{\beta}_0,\hat{\beta}_1} \sum_{i=1}^{n} (\hat{u}_i)^2 = \min_{\hat{\beta}_0,\hat{\beta}_1} \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i\right)^2$$

First order conditions (*recall the chain rule – Note the derivative with respect to beta's*):

$$-2\sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i\right) = 0$$

$$-2\sum_{i=1}^{n} x_i \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i\right) = 0$$

If h(x) = g(f(x)), then h' (x) = g' (f(x))f' (x).

Minimizing Residual Sum of Squares

$$-2\sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i\right) = 0 \qquad (1)$$

$$-2\sum_{i=1}^{n} x_i \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i\right) = 0 \qquad (2)$$

From Equation (1) we obtain:

$$\sum_{i=1}^{n} y_i = n\hat{\beta}_0 + \hat{\beta}_1 \sum_{i=1}^{n} x_i$$

If you divide both sides with *n*:

$$\bar{y} = \hat{\beta}_0 + \hat{\beta}_1 \bar{x} \Rightarrow \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

From Equation (2) we obtain:

$$\sum_{i=1}^{n} y_i x_i = \hat{\beta}_0 \sum_{i=1}^{n} x_i + \hat{\beta}_1 \sum_{i=1}^{n} x_i^2$$

Minimizing Residual Sum of Squares

From Equation (2) we obtain:

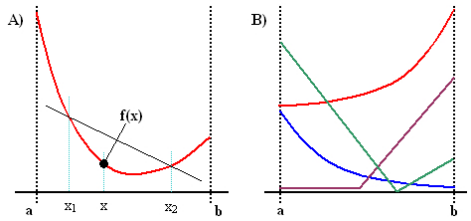
$$\sum_{i=1}^{n} y_i x_i = (\bar{y} - \hat{\beta}_1 \bar{x}) \sum_{i=1}^{n} x_i + \hat{\beta}_1 \sum_{i=1}^{n} x_i^2$$

If we solve for β1 we obtain the following equation:

$$\hat{\beta}_1 = \frac{\left(\sum_{i=1}^{n} y_i x_i - \bar{y} \sum_{i=1}^{n} x_i \right)}{\left(\sum_{i=1}^{n} x_i^2 - \bar{x} \sum_{i=1}^{n} x_i \right)}$$

Do they really minimize the squared error?

Convex functions

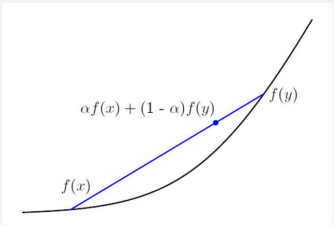


Definition #2: A function f(x) is convex if a line drawn between any two points on the function remains on or above the function in the interval between the two points.

Convex functions in multiple dimensions

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if for $x, y \in \text{dom} f$ and any $a \in [0, 1]$,

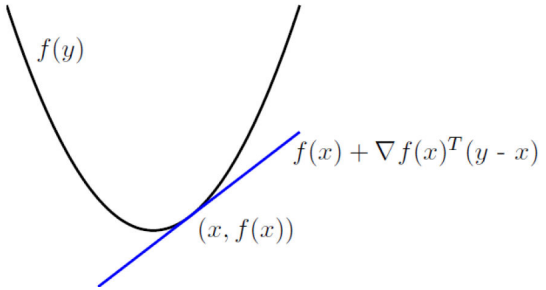
$$f(ax + (1 - a)y) \leq af(x) + (1 - a)f(y)$$



Convex functions: Alternative definition 1

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable. Then f is convex if and only if for all $x, y \in \text{dom } f$

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

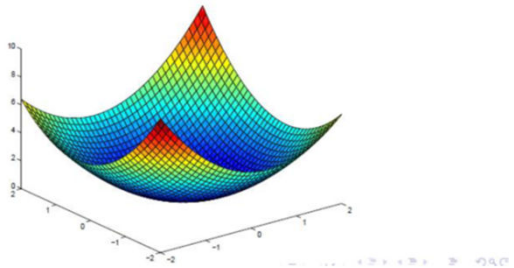


Convex functions: Alternative definition 2 (the Hessian is positive definite)

Theorem

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable. Then f is convex if and only if for all $x \in \operatorname{dom} f$,

$$\nabla^2 f(x) \succeq 0.$$



Gradient and the Hessian

Gradients

The single most important concept from calculus in the context of machine learning is the **gradient**. Gradients generalize derivatives to scalar functions of several variables. The gradient of $f : \mathbb{R}^d \to \mathbb{R}$, denoted ∇f, is given by

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \quad \text{i.e.} \quad [\nabla f]_i = \frac{\partial f}{\partial x_i}$$

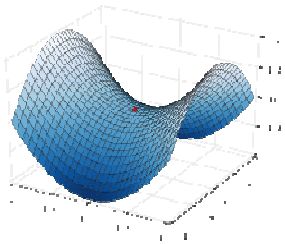
The Hessian

The **Hessian** matrix of $f : \mathbb{R}^d \to \mathbb{R}$ is a matrix of second-order partial derivatives:

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix} \quad \text{i.e.} \quad [\nabla^2 f]_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

Why the Hessian: Consider $f(x, y) = x^2 + 4xy + y^2$

Is a tilted (and scaled) version of $f(x, y) = x^2 - y^2$



OLS: Second Order Conditions

Second order partial derivatives:

$$\partial^2 L / \partial \beta_0^{\,2} = \sum_{i=1}^{n} 2 = 2n > 0.$$

$$\partial^2 L / \partial \beta_1^{\,2} = \sum_{i=1}^{n} 2x_i^2 = 2\sum_{i=1}^{n} x_i^2 > 0$$

$$\partial^2 L / \partial \beta_0 \partial \beta_1 = \sum_{i=1}^{n} 2x_i = 2\sum_{i=1}^{n} x_i.$$

The Hessian

$$\begin{bmatrix} 2n & 2\sum_{i=1}^{n} x_i \\ 2\sum_{i=1}^{n} x_i & 2\sum_{i=1}^{n} x_i^2 \end{bmatrix}.$$

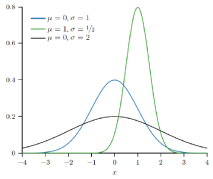
The determinant

$$4n\sum_{i=1}^{n} x_i^2 - 4(\sum_{i=1}^{n} x_i)^2 > 0$$ "Cauchy's Inequality"

Gaussian (Normal) distribution

- Distribution over real-valued scalar r.v. x
- Defined by a scalar **mean** μ and a scalar **variance** σ^2
- Distribution defined as

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



- Mean: $\mathbb{E}[x] = \mu$
- Variance: $\text{var}[x] = \sigma^2$

Linear Regression Assumptions

1. The distribution of X is arbitrary (and perhaps X is even non-random).
2. If $X = x$, then $Y = \beta_0 + \beta_1 x + \epsilon$, for some constants ("coefficients", "parameters") β_0 and β_1, and some random noise variable ϵ.
3. $\epsilon \sim N(0, \sigma^2)$, and is independent of X.
4. ϵ is independent across observations.

Because of these stronger assumptions, the model tells us the conditional pdf of Y for each x, $p(y|X = x; \beta_0, \beta_1, \sigma^2)$. (This notation separates the random variables from the parameters.) Given any data set $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$, we can now write down the probability density, under the model, of seeing that data:

$$\prod_{i=1}^{n} p(y_i|x_i; \beta_0, \beta_1, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - (\beta_0 + \beta_1 x_i))^2}{2\sigma^2}}$$

24

Maximum Likelihood Estimator

In multiplying together the probabilities like this, we are using the independence of the Y_i.

When we see the data, we do not *known* the true parameters, but any guess at them, say (b_0, b_1, s^2), gives us a probability density:

$$\prod_{i=1}^{n} p(y_i|x_i; b_0, b_1, s^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi s^2}} e^{-\frac{(y_i-(b_0+b_1x_i))^2}{2s^2}}$$

This is the **likelihood**, a function of the parameter values. It's just as informative, and much more convenient, to work with the **log-likelihood**,

$$\begin{aligned} L(b_0, b_1, s^2) &= \log \prod_{i=1}^{n} p(y_i|x_i; b_0, b_1, s^2) && (1) \\ &= \sum_{i=1}^{n} \log p(y_i|x_i; b_0, b_1, s^2) && (2) \\ &= -\frac{n}{2}\log 2\pi - n\log s - \frac{1}{2s^2}\sum_{i=1}^{n}(y_i - (b_0 + b_1x_i))^2 && (3) \end{aligned}$$

Maximum Likelihood Estimator

In the **method of maximum likelihood**, we p[ick the parameter values which maximize the likelihood, or, equivalently, maximize the log-likelihood.

After some calculus

$$\begin{aligned} \hat{\beta}_1 &= \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2} = \frac{c_{XY}}{s_X^2} \\ \hat{\beta}_0 &= \overline{y} - \hat{beta}_1\overline{x} \\ \hat{\sigma}^2 &= \frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1x_i))^2 \end{aligned}$$

OLS in Multiple Linear Regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

- We interpret β_j as the *average* effect on Y of a one unit increase in X_j, *holding all other predictors fixed*. In the advertising example, the model becomes

OLS in Multiple Linear Regression

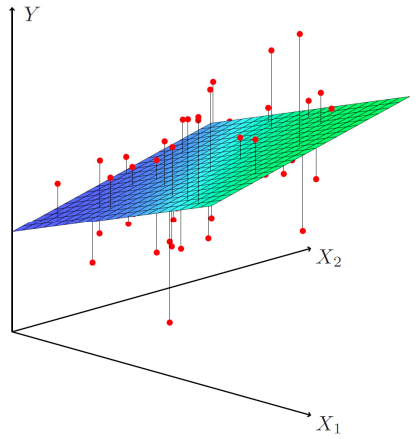
- Given estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots \hat{\beta}_p$, we can make predictions using the formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p.$$

- We estimate $\beta_0, \beta_1, \ldots, \beta_p$ as the values that minimize the sum of squared residuals

$$\begin{aligned} \text{RSS} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2. \end{aligned}$$

This is done using standard statistical software. The values $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ that minimize RSS are the multiple least squares regression coefficient estimates.



Sabancı Üniversitesi VBYO 2019

OLS in Multiple Linear Regression

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_{p-1} x_{i,p-1} + \varepsilon_i$$

- Let X be an $n \times k$ matrix where we have observations on k independent variables for n observations. Since our model will usually contain a constant term, one of the columns in the X matrix will contain only ones. This column should be treated exactly the same as any other column in the X matrix.
- Let y be an $n \times 1$ vector of observations on the dependent variable.
- Let ϵ be an $n \times 1$ vector of disturbances or errors.
- Let β be an $k \times 1$ vector of unknown population parameters that we want to estimate.

Our statistical model will essentially look something like the following:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ \vdots \\ Y_n \end{bmatrix}_{n\times 1} = \begin{bmatrix} 1 & X_{11} & X_{21} & \dots & X_{k1} \\ 1 & X_{12} & X_{22} & \dots & X_{k2} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & X_{1n} & X_{2n} & \dots & X_{kn} \end{bmatrix}_{n\times k} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \vdots \\ \beta_n \end{bmatrix}_{k\times 1} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \vdots \\ \epsilon_n \end{bmatrix}_{n\times 1}$$

This can be rewritten more simply as:

$$y = X\beta + \epsilon \qquad (1)$$

OLS in Multiple Linear Regression

The vector of residuals e is given by:

$$e = y - X\hat{\beta} \tag{2}$$

The sum of squared residuals (RSS) is $e'e$.[2]

$$\begin{bmatrix} e_1 & e_2 & \dots & \dots & e_n \end{bmatrix}_{1\times n} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ \vdots \\ e_n \end{bmatrix}_{n\times 1} = \begin{bmatrix} e_1 \times e_1 + e_2 \times e_2 + \ldots + e_n \times e_n \end{bmatrix}_{1\times 1} \tag{3}$$

It should be obvious that we can write the sum of squared residuals as:

$$\begin{aligned} e'e &= (y - X\hat{\beta})'(y - X\hat{\beta}) \\ &= y'y - \hat{\beta}'X'y - y'X\hat{\beta} + \hat{\beta}'X'X\hat{\beta} \\ &= y'y - 2\hat{\beta}'X'y + \hat{\beta}'X'X\hat{\beta} \end{aligned} \tag{4}$$

where this development uses the fact that the transpose of a scalar is the scalar i.e. $y'X\hat{\beta} = (y'X\hat{\beta})' = \hat{\beta}'X'y$.

OLS in Multiple Linear Regression

To find the $\hat{\beta}$ that minimizes the sum of squared residuals, we need to take the derivative of Eq. 4 with respect to $\hat{\beta}$. This gives us the following equation:

$$\frac{\partial e'e}{\partial \hat{\beta}} = -2X'y + 2X'X\hat{\beta} = 0 \tag{5}$$

To check this is a minimum, we would take the derivative of this with respect to $\hat{\beta}$ again – this gives us $2X'X$. It is easy to see that, so long as X has full rank, this is a positive definite matrix (analogous to a positive real number) and hence a minimum.

From Eq. 5 we get what are called the 'normal equations'.

$$(X'X)\hat{\beta} = X'y \tag{10}$$

$$\hat{\beta} = (X'X)^{-1}X'y$$

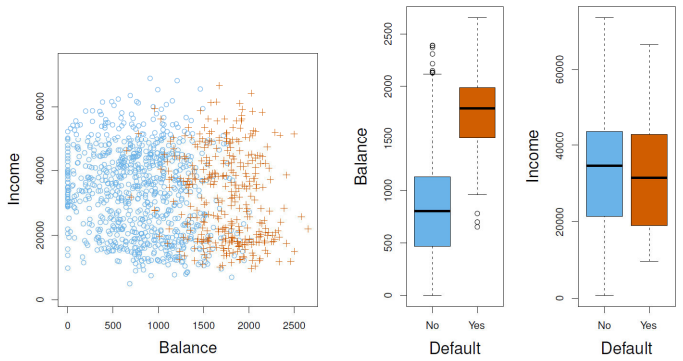
Classification
Logistic Regression

Classification

- Qualitative variables take values in an unordered set $\mathcal{C}$, such as:
  `eye color` $\in \{$`brown`, `blue`, `green`$\}$
  `email` $\in \{$`spam`, `ham`$\}$.
- Given a feature vector X and a qualitative response Y taking values in the set $\mathcal{C}$, the classification task is to build a function $C(X)$ that takes as input the feature vector X and predicts its value for Y; i.e. $C(X) \in \mathcal{C}$.
- Often we are more interested in estimating the *probabilities* that X belongs to each category in $\mathcal{C}$.

  For example, it is more valuable to have an estimate of the probability that an insurance claim is fraudulent, than a classification fraudulent or not.

Example: Credit Card Defualt

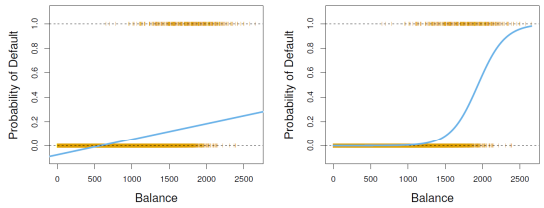


Can we use Linear Regression?

Suppose for the `Default` classification task that we code

$$Y = \begin{cases} 0 & \text{if } \texttt{No} \\ 1 & \text{if } \texttt{Yes}. \end{cases}$$

Can we simply perform a linear regression of Y on X and classify as `Yes` if $\hat{Y} > 0.5$?



The orange marks indicate the response Y, either 0 or 1. Linear regression does not estimate $\Pr(Y = 1|X)$ well. Logistic regression seems well suited to the task.

Logistic Regression

Let's write $p(X) = \Pr(Y = 1|X)$ for short and consider using `balance` to predict `default`. Logistic regression uses the form

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}.$$

($e \approx 2.71828$ is a mathematical constant [Euler's number.])
It is easy to see that no matter what values β_0, β_1 or X take, $p(X)$ will have values between 0 and 1.

A bit of rearrangement gives

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X.$$

This monotone transformation is called the *log odds* or *logit* transformation of $p(X)$.

> Logistic regression ensures that our estimate for $p(X)$ lies between 0 and 1.

Logistic regression with several variables

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

Logistic regression with more than two classes

So far we have discussed logistic regression with two classes.
It is easily generalized to more than two classes. One version

$$\Pr(Y = k|X) = \frac{e^{\beta_{0k}+\beta_{1k} X_1+\ldots+\beta_{pk} X_p}}{\sum_{\ell=1}^{K} e^{\beta_{0\ell}+\beta_{1\ell} X_1+\ldots+\beta_{p\ell} X_p}}$$

Likelihood function for Logistic regression

Because logistic regression predicts probabilities, rather than just classes, we can fit it using likelihood. For each training data-point, we have a vector of features, x_i, and an observed class, y_i. The probability of that class was either p, if $y_i = 1$, or $1 - p$, if $y_i = 0$. The likelihood is then

$$L(\beta_0, \beta) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i)^{1-y_i}$$

Sabancı Üniversitesi VBYO 2019

Likelihood function for Logistic regression

The log-likelihood turns products into sums:

$$\begin{aligned}
\ell(\beta_0, \beta) &= \sum_{i=1}^{n} y_i \log p(x_i) + (1 - y_i) \log 1 - p(x_i) \\
&= \sum_{i=1}^{n} \log 1 - p(x_i) + \sum_{i=1}^{n} y_i \log \frac{p(x_i)}{1 - p(x_i)} \\
&= \sum_{i=1}^{n} \log 1 - p(x_i) + \sum_{i=1}^{n} y_i(\beta_0 + x_i \cdot \beta) \\
&= \sum_{i=1}^{n} -\log 1 + e^{\beta_0 + x_i \cdot \beta} + \sum_{i=1}^{n} y_i(\beta_0 + x_i \cdot \beta)
\end{aligned}$$

Likelihood function for Logistic regression

Typically, to find the maximum likelihood estimates we'd differentiate the log likelihood with respect to the parameters, set the derivatives equal to zero, and solve. To start that, take the derivative with respect to one component of β, say β_j.

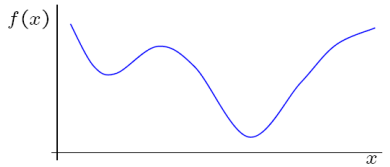
$$\begin{aligned}\frac{\partial \ell}{\partial \beta_j} &= -\sum_{i=1}^{n} \frac{1}{1+e^{\beta_0+x_i\cdot\beta}} e^{\beta_0+x_i\cdot\beta} x_{ij} + \sum_{i=1}^{n} y_i x_{ij} \\ &= \sum_{i=1}^{n} \left(y_i - p(x_i;\beta_0,\beta)\right) x_{ij}\end{aligned}$$

We are not going to be able to set this to zero and solve exactly. (That's a transcendental equation, and there is no closed-form solution.) We can however approximately solve it numerically.

Unconstrained univariate optimization

Assume we can start close to the global minimum

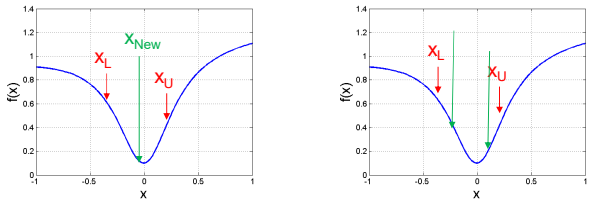
$$\min_x f(x)$$



How to determine the minimum?

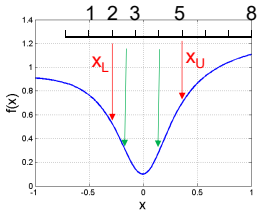
- Search methods (Dichotomous, Fibonacci, Golden-Section)
- Approximation methods
  - Polynomial interpolation
- Derivative based methods
  1. Gradient
  2. Newton method
  3. Stochastic gradient

Search methods



Dichotomous

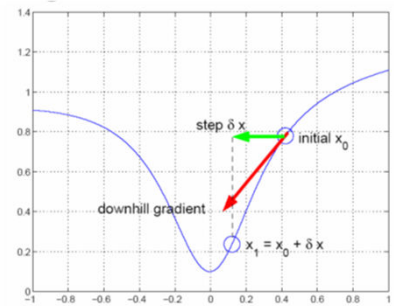
Golden section



Fibonacci: 1 1 2 3 5 8 …

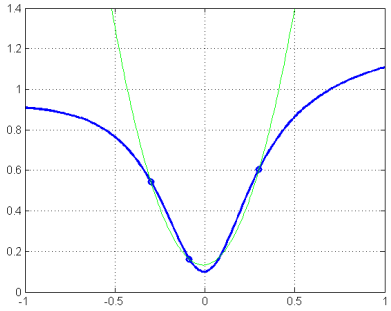
Gradient descent

Given a starting location, x_0, examine df/dx and move in the *downhill* direction to generate a new estimate, $x_1 = x_0 + \delta x$



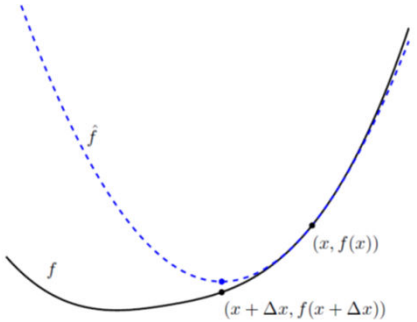
How to determine the step size δx?

Polynomial interpolation



- Quadratic interpolation using 3 points, 2 iterations
- Other methods to interpolate?
  - 2 points and one gradient
  - Cubic interpolation

Single step in Newton's method



$\hat{f}$ is 2nd-order approximation, f is true function.

Newton method

Fit a quadratic approximation to $f(x)$ using both gradient and curvature information at x.

- Expand $f(x)$ locally using a Taylor series.

$$f(x+\delta x) = f(x) + f'(x)\delta x + \frac{1}{2}f''(x)\delta x^2 + o(\delta x^2)$$

- Find the δx which minimizes this local quadratic approximation.

$$\delta x = -\frac{f'(x)}{f''(x)}$$

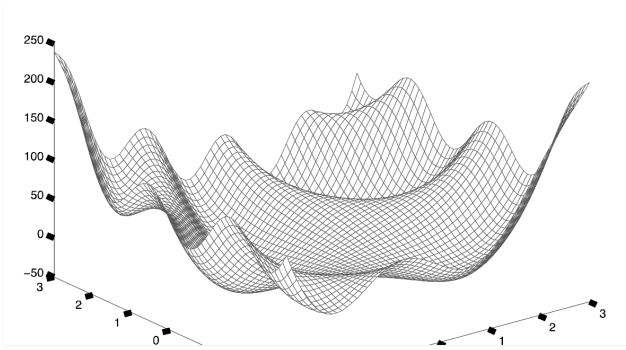
- Update x.

$$x_{n+1} = x_n - \delta x = x_n - \frac{f'(x)}{f''(x)}$$

Stochastic gradient

- Any iteration of gradient descent method requires that we sum over the entire dataset to compute gradient.
- SGD idea: at each iteraton, sub-sample a small amount of data (even just 1 point can work) and use that to estimate the gradient. (typically use around 100 samples)
- Each update is noisy, but very fast!
- This is the basis of optimizing ML algorithms with huge datasets (e.g., deep learning).

Why Optimization is difficult in general?



Lots of metaheuristics

- Genetic Algorithms
- Simulated Annealing
- Tabu Search
- Scatter Search
- Ant Colony Optimization
- Particle Swarm Optimization
- Iterated Local Search
- Variable Neighborhood Search
- …

Regularization (Shrinkage)

Ridge regression

- Recall that the least squares fitting procedure estimates $\beta_0, \beta_1, \ldots, \beta_p$ using the values that minimize

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

- In contrast, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2,$$

where $\lambda \geq 0$ is a *tuning parameter*, to be determined separately.

52

Lasso regression

- The *Lasso* is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso coefficients, $\hat{\beta}_\lambda^L$, minimize the quantity

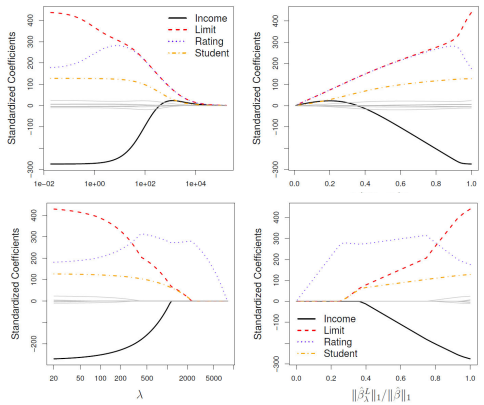
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \;= \text{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|.$$

- In statistical parlance, the lasso uses an ℓ_1 (pronounced "ell 1") penalty instead of an ℓ_2 penalty. The ℓ_1 norm of a coefficient vector β is given by $\|\beta\|_1 = \sum|\beta_j|$.

The Lasso: continued

- As with ridge regression, the lasso shrinks the coefficient estimates towards zero.
- However, in the case of the lasso, the ℓ_1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large.
- Hence, much like best subset selection, the lasso performs *variable selection*.
- We say that the lasso yields *sparse* models — that is, models that involve only a subset of the variables.

Ridge vs. Lasso



Ridge

Lasso

Lasso and Ridge as constrained optimization

One can show that the lasso and ridge regression coefficient estimates solve the problems

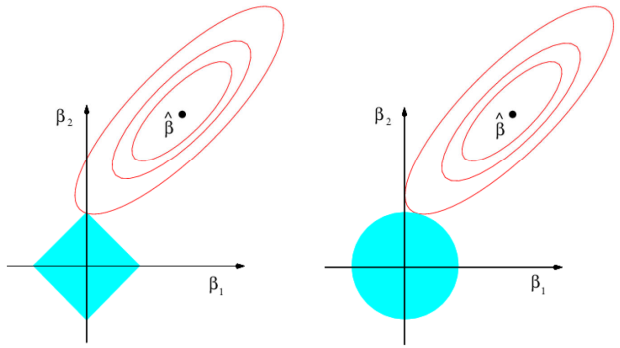
$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s$$

and

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s,$$

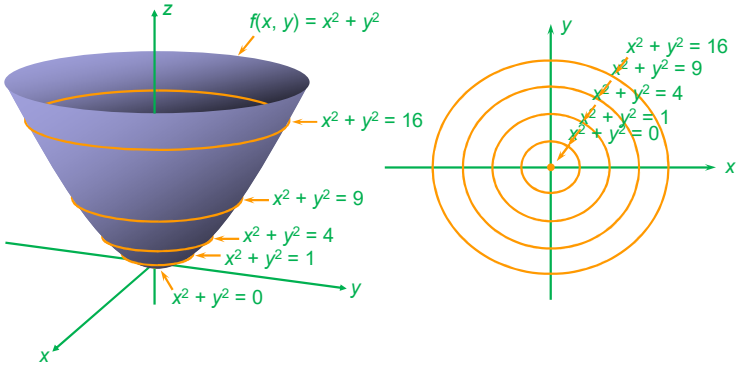
respectively.

Lasso and Ridge: A Picture



Level curves

- Level curves of the function $f(x, y) = x^2 + y^2$.
- A level curve is the graph of the equation $x^2 + y^2 = c$, which describes a circle
- Taking different values of c we obtain:



Constrained Optimization

General Optimization Problem: Standard Form

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leqslant 0, \;\; i = 1, \ldots, m \\ & h_i(x) = 0, \;\; i = 1, \ldots p, \end{array}$$

where $x \in \mathbf{R}^n$ are the **optimization variables** and f_0 is the **objective function.**

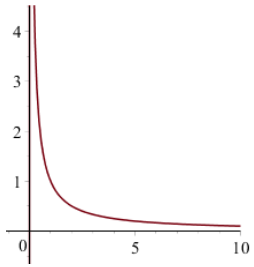
Assume **domain** $\mathcal{D} = \bigcap_{i=0}^{m} \mathbf{dom}\, f_i \cap \bigcap_{i=1}^{p} \mathbf{dom}\, h_i$ is nonempty.

- The set of points satisfying the constraints is called the **feasible set**.
- A point x in the feasible set is called a **feasible point.**
- If x is feasible and $f_i(x) = 0$,
  - then we say the inequality constraint $f_i(x) \leqslant 0$ is **active** at x.
- The **optimal value** p^* of the problem is defined as

$$p^* = \inf\{f_0(x) \mid x \text{ satisfies all constraints}\}.$$

- x^* is an **optimal point** (or a solution to the problem) if x^* is feasible and $f(x^*) = p^*$.

Why “inf” and “sup” instead of “min” and “max”



“Min” and “max” are not well-defined

Sabancı Üniversitesi VBYO 2019

Reduced Form: Drop equality constraints

- Note that

$$h(x) = 0 \iff (h(x) \geqslant 0 \text{ AND } h(x) \leqslant 0)$$

- Consider an equality-constrained problem:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & h(x) = 0 \end{array}$$

- Can be rewritten as

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & h(x) \leqslant 0 \\ & -h(x) \leqslant 0. \end{array}$$

The Lagrangian

The general [inequality-constrained] optimization problem is:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leqslant 0, \quad i = 1, \ldots, m \end{array}$$

Definition

The **Lagrangian** for this optimization problem is

$$L(x, \lambda) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x).$$

- λ_i's are called **Lagrange multipliers** (also called the **dual variables**).
- Supremum over Lagrangian gives back encoding of objective and constraints:

$$\begin{aligned} \sup_{\lambda \succeq 0} L(x, \lambda) &= \sup_{\lambda \succeq 0} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) \right) \\ &= \begin{cases} f_0(x) & \text{when } f_i(x) \leqslant 0 \text{ all } i \\ \infty & \text{otherwise.} \end{cases} \end{aligned}$$

- Equivalent **primal form** of optimization problem:

$$p^* = \inf_x \sup_{\lambda \succeq 0} L(x, \lambda)$$

The Primal and the Dual

- Original optimization problem in **primal form:**

$$p^* = \inf_x \sup_{\lambda \succeq 0} L(x, \lambda)$$

- Get the **Lagrangian dual problem** by "swapping the inf and the sup":

$$d^* = \sup_{\lambda \succeq 0} \inf_x L(x, \lambda)$$

- We will show **weak duality**: $p^* \geqslant d^*$ for any optimization problem.

Weak Max-Min Inequality

Theorem

*For **any** $f : W \times Z \to \mathbf{R}$, we have*

$$\sup_{z \in Z} \inf_{w \in W} f(w, z) \leqslant \inf_{w \in W} \sup_{z \in Z} f(w, z).$$

Proof: For any $w_0 \in W$ and $z_0 \in Z$, we clearly have

$$\inf_{w \in W} f(w, z_0) \leqslant f(w_0, z_0) \leqslant \sup_{z \in Z} f(w_0, z).$$

Since $\inf_{w \in W} f(w, z_0) \leqslant \sup_{z \in Z} f(w_0, z)$ for all w_0 and z_0, we must also have

$$\sup_{z_0 \in Z} \inf_{w \in W} f(w, z_0) \leqslant \inf_{w_0 \in W} \sup_{z \in Z} f(w_0, z).$$

Weak Duality

- For any optimization problem (**not just convex**), weak max-min inequality implies **weak duality**:

$$\begin{aligned} p^* &= \inf_x \sup_{\lambda \succeq 0} \left[f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) \right] \\ &\geqslant \sup_{\lambda \succeq 0, \nu} \inf_x \left[f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) \right] = d^* \end{aligned}$$

- The difference $p^* - d^*$ is called the **duality gap**.
- For *convex* problems, we often have **strong duality**: $p^* = d^*$.

The Lagrange Dual Function

- The **Lagrangian dual problem**:

$$d^* = \sup_{\lambda \succeq 0} \inf_x L(x, \lambda)$$

Definition

The **Lagrange dual function** (or just **dual function**) is

$$g(\lambda) = \inf_x L(x, \lambda) = \inf_x \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) \right).$$

- The dual function may take on the value $-\infty$ (e.g. $f_0(x) = x$).
- The dual function is always **concave**
  - since pointwise min of affine functions

The Lagrange Dual Problem: Search for Best Lower Bound

- In terms of Lagrange dual function, we can write weak duality as

$$p^* \geqslant \sup_{\lambda \geqslant 0} g(\lambda) = d^*$$

- So for any λ with $\lambda \geqslant 0$, **Lagrange dual function gives a lower bound on optimal solution**:

$$p^* \geqslant g(\lambda) \text{ for all } \lambda \geqslant 0$$

- The **Lagrange dual problem** is a search for best lower bound on p^*:

$$\begin{array}{ll} \text{maximize} & g(\lambda) \\ \text{subject to} & \lambda \succeq 0. \end{array}$$

  - λ **dual feasible** if $\lambda \succeq 0$ and $g(\lambda) > -\infty$.
  - λ^* **dual optimal** or **optimal Lagrange multipliers** if they are optimal for the Lagrange dual problem.
- Lagrange dual problem often easier to solve (simpler constraints).
- d^* can be used as stopping criterion for primal optimization.
- Dual can reveal hidden structure in the solution.

Complementary Slackness

- Consider a general optimization problem (i.e. not necessarily convex).
- If we have **strong duality**, we get an interesting relationship between
  - the optimal Lagrange multiplier λ_i^* and
  - the ith constraint at the optimum: $f_i(x^*)$
- Relationship is called "**complementary slackness**":

$$\lambda_i^* f_i(x^*) = 0$$

- Always have Lagrange multiplier is zero **or** constraint is active at optimum **or** both.

Complementary Slackness "Sandwich Proof"

- Assume strong duality: $p^* = d^*$ in a general optimization problem
- Let x^* be primal optimal and λ^* be dual optimal. Then:

$$\begin{aligned} f_0(x^*) &= g(\lambda^*) = \inf_x L(x, \lambda^*) \quad \text{(strong duality and definition)} \\ &\leqslant L(x^*, \lambda^*) \\ &= f_0(x^*) + \sum_{i=1}^{m} \underbrace{\lambda_i^* f_i(x^*)}_{\leqslant 0} \\ &\leqslant f_0(x^*). \end{aligned}$$

Each term in sum $\sum_{i=1} \lambda_i^* f_i(x^*)$ must actually be 0. That is

$$\boxed{\lambda_i^* f_i(x^*) = 0, \quad i = 1, \ldots, m}.$$

This condition is known as **complementary slackness**.

When there is equality constraint

Consider the following constrained problem:

$$\begin{aligned} \textbf{minimize} \quad & f(x) \\ \textbf{subject to} \quad & g_j(x) \leq 0, \quad j = 1, \ldots, m \\ & h_\ell(x) = 0, \quad \ell = 1, \ldots, r \end{aligned} \tag{4}$$

The functions f and all g_j and h_ℓ are continuously differentiable over $\mathbb{R}^n$. The optimality conditions are specified through the use of Lagrangian Function:

- Introduce a multiplier (shadow price) per constraint:
  $\mu_j \geq 0$ for each constraint $g_j(x) \leq 0$ and
  λ_ℓ for each constraint $h_\ell(x) = 0$
- Lagrangian Function is given by

$$L(x, \mu, \lambda) = f(x) + \sum_{j=1}^{m} \mu_j g_j(x) + \sum_{\ell=1}^{r} \lambda_\ell h_\ell(x)$$

where $\mu = (\mu_1, \ldots, \mu_m)$ and $\lambda = (\lambda_1, \ldots, \lambda_r)$

Necessary Optimality Conditions

Assuming some regularity conditions for problem (4), if x^* is an optimal solution of the problem, then there exist Lagrange multipliers (optimal shadow prices) $\mu^* = (\mu_1^*, \ldots, \mu_m^*) \geq 0$ and $\lambda^* = (\lambda_1^*, \ldots, \lambda_r^*)$ such that

$$\nabla f(x^*) + \textstyle\sum_{j=1}^{m} \mu_j^* \nabla g_j(x^*) + \sum_{\ell=1}^{r} \lambda_\ell^* \nabla h_\ell(x^*) = 0$$

$$g_j(x^*) \leq 0 \quad \text{for all } j = 1, \ldots, m$$

$$h_\ell(x^*) = 0 \text{ for all } \ell = 1, \ldots, r$$

$$\mu_j^* \geq 0 \quad \text{for all } j = 1, \ldots, m$$

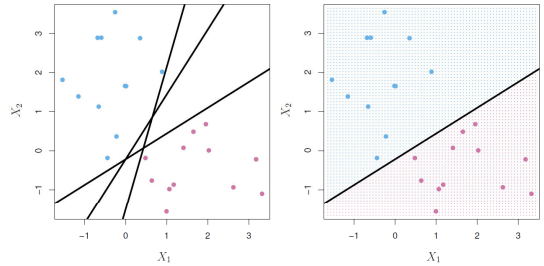
$$\mu_j^* g_j(x^*) = 0 \quad \text{for all } j = 1, \ldots, m$$

μ^* and λ^* is the optimal shadow price associated with the solution x^*

This is the KKT condition

Support Vector Machines

Separating Hyperplanes

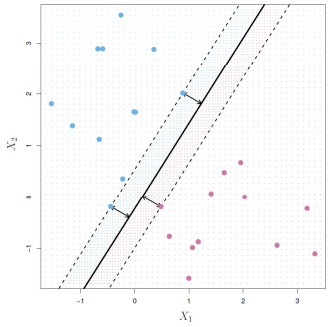


- If $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$, then $f(X) > 0$ for points on one side of the hyperplane, and $f(X) < 0$ for points on the other.
- If we code the colored points as $Y_i = +1$ for blue, say, and $Y_i = -1$ for mauve, then if $Y_i \cdot f(X_i) > 0$ for all i, $f(X) = 0$ defines a *separating hyperplane*.

74

Maximal Margin Classifier

Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.



Constrained optimization problem

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$$

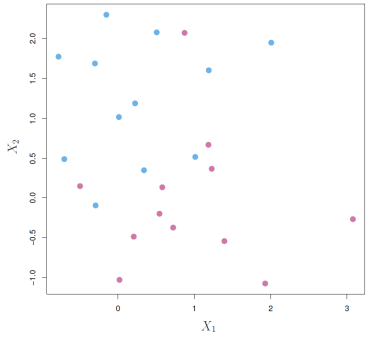
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M$$

$$\text{for all } \; i = 1, \ldots, N.$$

75

Non-separable Data

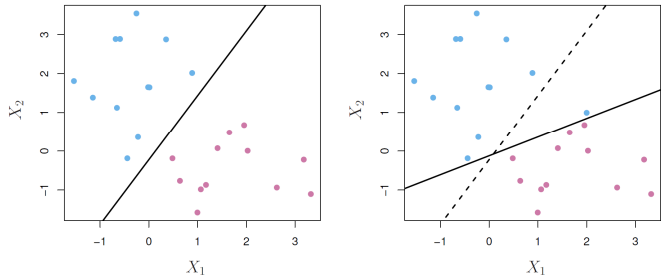


The data on the left are not separable by a linear boundary.

This is often the case, unless $N < p$.

76

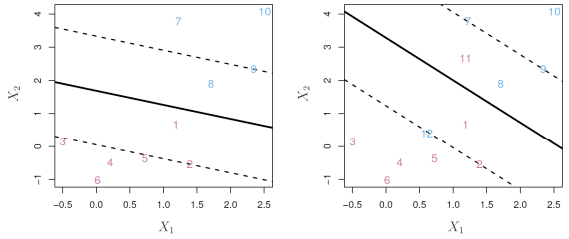
Noisy Data



Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier.

The *support vector classifier* maximizes a *soft* margin.

Support Vector Classifier (soft margin)

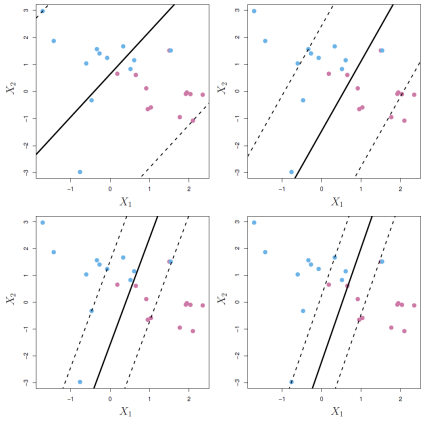


$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \; M \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

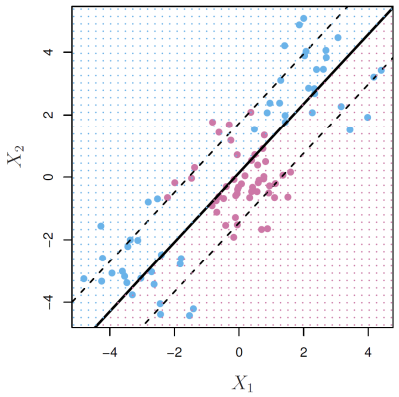
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \; \sum_{i=1}^{n} \epsilon_i \leq C,$$

C is a regularization parameter



Linear boundary can fail



Sometime a linear boundary simply won't work, no matter what value of C.

The example on the left is such a case.

What to do?

Feature Expansion

- Enlarge the space of features by including transformations; e.g. X_1^2, X_1^3, X_1X_2, $X_1X_2^2$,.... Hence go from a p-dimensional space to a $M > p$ dimensional space.
- Fit a support-vector classifier in the enlarged space.
- This results in non-linear decision boundaries in the original space.

Example: Suppose we use $(X_1,\ X_2,\ X_1^2,\ X_2^2,\ X_1X_2)$ instead of just (X_1, X_2). Then the decision boundary would be of the form

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 = 0$$

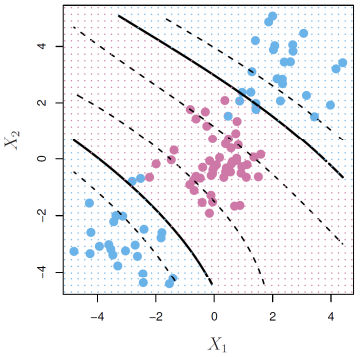
This leads to nonlinear decision boundaries in the original space (quadratic conic sections).

Cubic Polynomials

Here we use a basis expansion of cubic polynomials

From 2 variables to 9

The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space



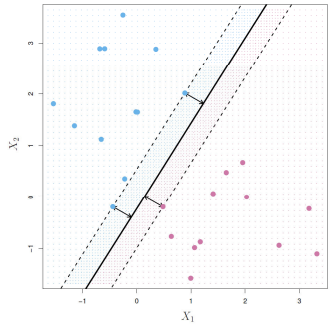
$$\beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_1^2+\beta_4X_2^2+\beta_5X_1X_2+\beta_6X_1^3+\beta_7X_2^3+\beta_8X_1X_2^2+\beta_9X_1^2X_2 = 0$$

Nonlinearities and Kernels

- Polynomials (especially high-dimensional ones) get wild rather fast.
- There is a more elegant and controlled way to introduce nonlinearities in support-vector classifiers — through the use of *kernels*.

SVM Again: Maximal Margin Classifier

Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.



Constrained optimization problem

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M$$

$$\text{for all } \; i = 1, \ldots, N.$$

Maximal Margin Classifier: Matrix Notation

$$\max_{\beta,\beta_0,||\beta||=1} M \quad \text{subject to } y_i(x_i^T\beta+\beta_0)\geq M,\ i=1,\ldots,N. \tag{4.45}$$

We can get rid of the $||\beta|| = 1$ constraint by replacing the conditions with

$$\frac{1}{||\beta||}y_i(x_i^T\beta+\beta_0)\geq M, \tag{4.46}$$

(which redefines β_0) or equivalently

$$y_i(x_i^T\beta+\beta_0)\geq M||\beta||. \tag{4.47}$$

Since for any β and β_0 satisfying these inequalities, any positively scaled multiple satisfies them too, we can arbitrarily set $||\beta|| = 1/M$. Thus (4.45) is equivalent to

$$\min_{\beta,\beta_0}\frac{1}{2}||\beta||^2 \quad \text{subject to } y_i(x_i^T\beta+\beta_0)\geq 1,\ i=1,\ldots,N. \tag{4.48}$$

SVM Again: Maximal Margin Classifier

The Lagrange (primal) function, to be minimized w.r.t. β and β_0, is

$$L_P = \frac{1}{2}||\beta||^2-\sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta+\beta_0)-1]. \tag{4.49}$$

Setting the derivatives to zero, we obtain:

$$\beta = \sum_{i=1}^{N}\alpha_i y_i x_i, \tag{4.50}$$

$$0 = \sum_{i=1}^{N}\alpha_i y_i, \tag{4.51}$$

and substituting these in (4.49) we obtain the so-called Wolfe dual

$$L_D = \sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N}\alpha_i\alpha_k y_i y_k x_i^T x_k$$
$$\text{subject to } \alpha_i\geq 0 \text{ and } \sum_{i=1}^{N}\alpha_i y_i = 0. \tag{4.52}$$

SVM: Kernels

We can represent the optimization problem (12.9) and its solution in a special way that only involves the input features via inner products. We do this directly for the transformed feature vectors $h(x_i)$. We then see that for particular choices of h, these inner products can be computed very cheaply.

The Lagrange dual function (12.13) has the form

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle. \tag{12.19}$$

From (12.10) we see that the solution function $f(x)$ can be written

$$\begin{aligned} f(x) &= h(x)^T \beta + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0. \end{aligned} \tag{12.20}$$

SVM: Kernels

So both (12.19) and (12.20) involve $h(x)$ only through inner products. In fact, we need not specify the transformation $h(x)$ at all, but require only knowledge of the kernel function

$$K(x, x') = \langle h(x), h(x') \rangle \tag{12.21}$$

Three popular choices for K in the SVM literature are

$$\begin{aligned} &d\text{th-Degree polynomial: } K(x, x') = (1 + \langle x, x' \rangle)^d, \\ &\text{Radial basis: } K(x, x') = \exp(-\gamma \|x - x'\|^2), \\ &\text{Neural network: } K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2). \end{aligned} \tag{12.22}$$

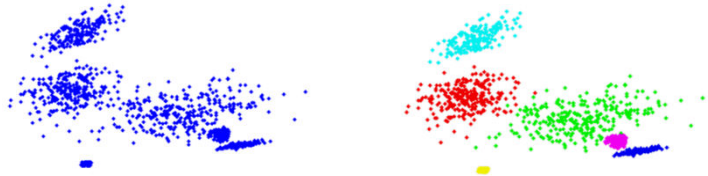
Sabancı Üniversitesi VBYO 2019

Clustering

Sabancı Üniversitesi VBYO 2019

What is Cluster Analysis?

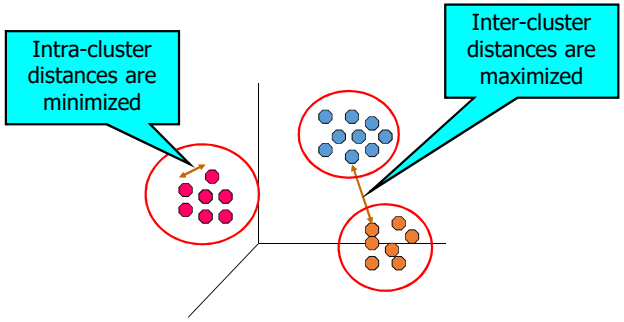
Cluster analysis or **clustering** is the assignment of a set of observations into subsets (called *clusters*) so that observations in the same cluster are similar in some sense.



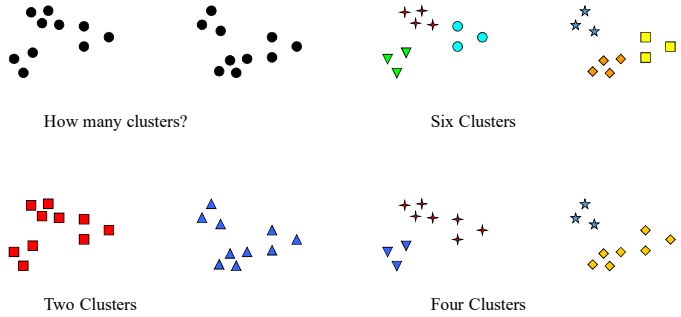
90

What is Cluster Analysis?

- ***More formal perspective:*** Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



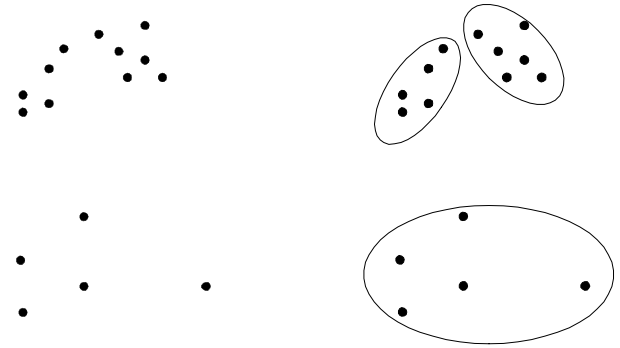
Notion of a Cluster can be Ambiguous



Types of Clusterings

- A clustering is a set of clusters
- Important distinction between hierarchical and partitional sets of clusters
- Partitional Clustering
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree

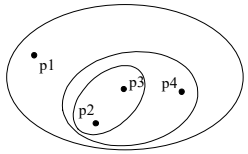
Partitional Clustering



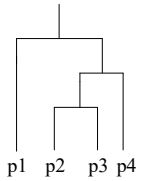
Original Points

A Partitional Clustering

Hierarchical Clustering



Hierarchical Clustering



Dendrogram

Other Distinctions Between Sets of Clusters

- Exclusive versus non-exclusive
  - In non-exclusive clusterings, points may belong to multiple clusters.
  - Can represent multiple classes or 'border' points
- Fuzzy versus non-fuzzy
  - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
  - Weights "must" sum to 1
  - Probabilistic clustering has similar characteristics
- Partial versus complete
  - In some cases, we only want to cluster some of the data
- Heterogeneous versus homogeneous
  - Cluster of widely different sizes, shapes, and densities

Graph Theory Representation of Clustering

- Map the clustering problem to a graph and solve a related problem in that domain
  - **Proximity matrix** defines a weighted graph, where the **nodes** are the points being clustered, and the weighted **edges** represent the proximities between points
  - Clustering is equivalent to breaking the graph into **connected components**, one for each cluster.
  - Want to **minimize** the edge weight (i.e., *max distance*) **between** clusters and **maximize** the edge weight (i.e., *min distance*) **within** clusters

Sabancı Üniversitesi VBYO 2019

K-means

Data set: $X = \{\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n\}$
Clusters: $C_1, C_2, \cdots C_k$
Codebook : $V = \{\bar{v}_1, \bar{v}_2, \cdots, \bar{v}_k\}$
Partition matrix: $\Gamma = \{\gamma_{ij}\}$

$$\gamma_{ij} = \begin{cases} 1 & \text{if } \bar{x}_j \in C_i \\ 0 & \text{otherwise} \end{cases}$$

- Minimizes functional:

$$E(\Gamma, V) = \sum_{i=1}^{k} \sum_{j=1}^{n} \gamma_{ij} \left\| \bar{x}_j - \bar{v}_i \right\|^2$$

- Iterative algorithm:
  - Initialize the codebook *V* with vectors randomly picked from *X*
  - Assign each pattern to the nearest cluster
  - Recalculate partition matrix
  - Repeat the above two steps until convergence

Sabancı Üniversitesi VBYO 2019

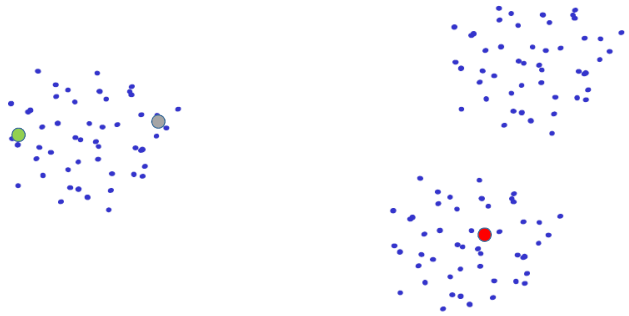
K-means

- Disadvantages
  - Dependent on initialization

Sabancı Üniversitesi VBYO 2019

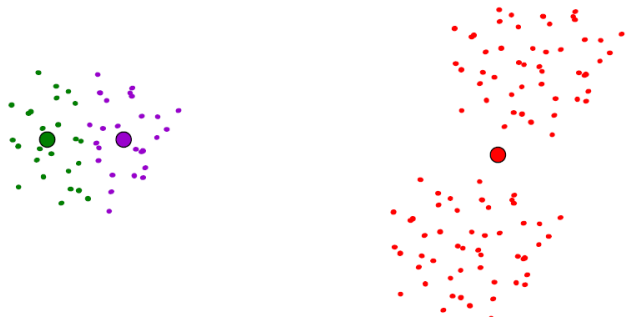
K-means

- Disadvantages
  - Dependent on initialization



K-means

- Disadvantages
  - Dependent on initialization



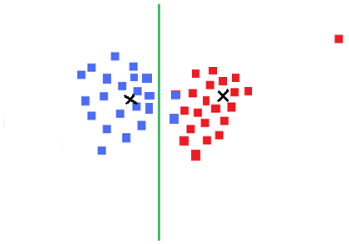
Sabancı Üniversitesi VBYO 2019

K-means

- Disadvantages
  - Dependent on initialization
    - Select random seeds with at least D_{min}
    - Or, run the algorithm many times

K-means

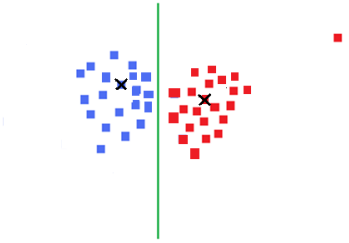
- Disadvantages
  - Dependent on initialization
  - Sensitive to outliers



Sabancı Üniversitesi VBYO 2019

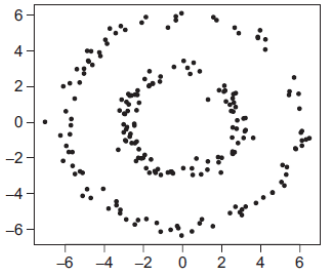
K-means

- Disadvantages
  - Dependent on initialization
  - Sensitive to outliers
    - Use K-medoids



K-means

- Disadvantages
  - Dependent on initialization
  - Sensitive to outliers (K-medoids)
  - Can deal only with clusters with spherical symmetrical point distribution
    - Kernel trick



Sabancı Üniversitesi VBYO 2019

K-means

- Disadvantages
  - Dependent on initialization
  - Sensitive to outliers (K-medoids)
  - Can deal only with clusters with spherical symmetrical point distribution
  - Deciding *K*

Deciding K

- Try a couple of K

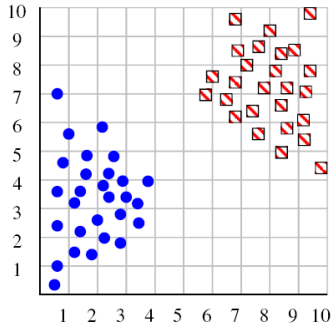


Image: Henry Lin

Deciding K

- When k = 1, the objective function is 873.0

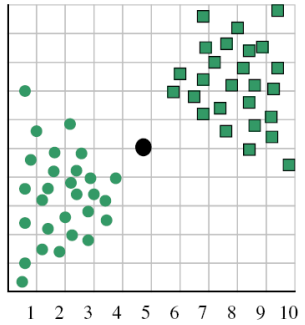


Image: Henry Lin

Deciding K

- When k = 2, the objective function is 173.1

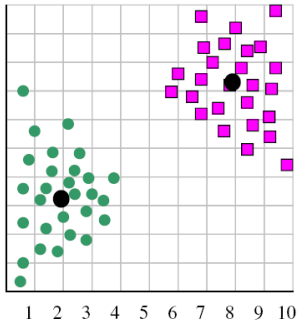


Image: Henry Lin

Deciding K

- When k = 3, the objective function is 133.6

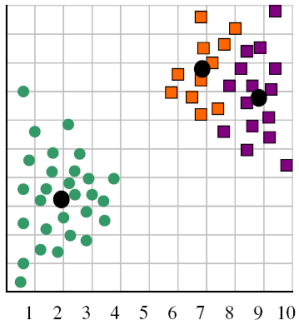


Image: Henry Lin

Deciding K

- We can plot objective function values for k=1 to 6
- The abrupt change at k=2 is highly suggestive of two clusters
- "knee finding" or "elbow finding"
- Note that the results are not always as clear cut as in this toy example