K

k-Armed Bandit

Shie Mannor
Israel Institute of Technology, Haifa, Israel

Synonyms

Multi-armed bandit; Multi-armed bandit problem

Definition

In the classical *k*-armed bandit problem, there are *k* alternative arms, each with a stochastic reward whose probability distribution is initially unknown. A decision maker can try these arms in some order, which may depend on the rewards that have been observed so far. A common objective in this context is to find a policy for choosing the next arm to be tried, under which the sum of the expected rewards comes as close as possible to the ideal reward, that is, the expected reward that would be obtained if it were to try the "best" arm at all times. There are many variants of the *k*-armed bandit problem that are distinguished by the objective of the decision maker, the process governing the reward of each arm, and the information available to the decision maker at the end of every trial.

Motivation and Background

k-Armed bandit problems are a family of sequential decision problems that are among the most studied problems in statistics, control, decision theory, and machine learning. In spite of their simplicity, they encompass many of the basic problems of sequential decision making in uncertain environments such as the tradeoff between exploration and exploitation.

There are many variants of bandit problems including Bayesian, Markovian, adversarial, budgeted, and exploratory variants. Bandit formulations arise naturally in multiple fields and disciplines including communication networks, clinical trials, search theory, scheduling, supply chain automation, finance, control, information technology, etc. (Berry & Fristedt, 1985; Cesa-Bianchi & Lugosi, 2006; Gittins, 1989).

The term "multi-armed bandit" is borrowed from the slang term for a slot machine (the one-armed bandit), where a decision maker has to decide whether to insert a coin into the gambling machine and pull a lever possibly getting a significant reward, or to quit without spending any money.

Theory

We briefly review some of the most popular bandit variants.

The Stochastic *k*-Armed Bandit Problem

The classical stochastic bandit problem is described as follows. There are *k* arms (or machines or actions) and a single decision maker (or controller or agent). Each arm corresponds to a discrete time Markov process. At each timestep, the decision maker observes the current state of each arm's process and selects one of the arms. As a result, the decision maker obtains a reward from the process of the selected arm and the state of the corresponding process changes. Arms that are not selected are "frozen" and their processes remain in the same state. The objective of the decision maker is to maximize her (discounted) reward.

More formally, let the state of arm n's process at stage t be $x_n(t)$. Then, if the decision maker selects arm $m(t)$ at time t we have that:

$$x_n(t+1) = \begin{cases} x_n(t) & n \neq m(t) \\ f_n\left(x_n(t), \omega\right) & n = m(t) \end{cases},$$

where $f_n(x, \omega)$ is a function that describes the (possibly stochastic) transition probability of the n-th process and accepts the state of the n-th process and a random disturbance ω.

Claude Sammut & Geoffrey I. Webb (eds.), *Encyclopedia of Machine Learning*, DOI 10.1007/978-0-387-30164-8,

The reward the decision maker receives at time t is a function of the current state and a random element: $r(x_{m(t)}(t), \omega)$. The objective of the decision maker is to maximize her cumulative discounted reward. That is, she wishes to maximize

$$V = \mathbf{E}^{\pi}\left[\sum_{t=1}^{\infty} \gamma^t r\left(x_{m(t)}(t), \omega_t\right)\right],$$

where $\mathbf{E}^{\pi}$ is the expectation obtained when following policy π and γ is a discount factor $(0 < \gamma < 1)$. A policy is a decision rule for selected arms as a function of the state of the processes.

This problem can be solved using ▶dynamic programming, but the state space of the joint Markov decision process is exponential in the number of arms. Moreover, the dynamic programming solution does not reveal the important structural properties of the solution.

Gittins and Jones (1972) showed that there exists an optimal index policy. That is, there is a function that maps the state of each arm to real number (the "index") such that the optimal policy is to choose the arm with the highest index at any given time. Therefore, the stochastic bandit problem reduces to the problem of computing the index, which can be easily done in many important cases.

Regret Minimization for the Stochastic *k*-Armed Bandit Problem

A different flavor of the bandit problem focuses on the notion of regret, or learning loss. In this formulation, there are k arms as before and when selecting arm m a reward that is independent and identically distributed is given (the reward depends only on the identity of the arm and not on some internal state or the results of previous trials). The decision maker's objective is to obtain high expected reward. Of course, if the decision maker had known the statistical properties of each arm, she would have always chosen the arm with the highest expected reward. However, the decision maker does not know the statistical properties of the arms in advance, in this setting.

More formally, if the reward when choosing arm m has expectation r_m, the regret is defined as:

$$r(t) = t \cdot \max_{1 \le m \le k} r_m - \mathbf{E}^{\pi}\left[\sum_{\tau=1}^{t} r(\tau)\right],$$

where $r(t)$ is sampled from the arm $m(t)$. This quantity represents the expected loss for not choosing the arm with the highest expected reward on every timestep.

This variant of the bandit problem highlights the tension between acquiring information (exploration) and using the available information (exploitation). The decision maker should carefully balance between the two since if she chooses to only try the arm with the highest estimated reward she might regret not exploring other arms whose reward is underestimated but is actually higher than the reward of the arm with highest estimated reward.

A basic question in this context is whether $R(t)$ can be made to grow sub-linearly. Robbins (1952) answered this question in the affirmative. It was later proved (Lai & Robbins, 1985) that it is possible in fact to obtain logarithmic regret (the growth of the regret is logarithmic in the number of timesteps). Matching lower bounds (and constants) were also derived.

The Non-stochastic *k*-Armed Bandit Problem

A third popular variant of the bandit problem is the non-stochastic one. In this problem, it is assumed that the sequence of rewards each arm produces is deterministic (possibly adversarial). The decision maker, as in the stochastic bandit problem, wants to minimize her regret, where the regret is measured with respect to the best fixed arm (this best arm might change with time, however). Letting the reward of arm m at time t be $r_m(t)$, we redefine the regret as:

$$r(t) = \max_{1 \le m \le k} \sum_{\tau=1}^{t} r_m(\tau) - \mathbf{E}^{\pi}\left[\sum_{\tau=1}^{t} r(\tau)\right],$$

where the expectation is now taken with respect to randomness in the arm selection. The basic question here is if the regret can be made to grow sub-linearly. The case where the reward of each arm is observed was addressed in the 1950s (see Cesa-Bianchi & Lugosi, 2006, for a discussion), where it was shown that there are algorithms that guarantee that the regret grows like $\sqrt{t}$. For the more difficult case, where only the reward of the selected arm is observed and that the rewards of the other arms may not be observed it was shown (Auer, Cesa-Bianchi, Freund, & Schapire, 2002) that the same conclusion still holds.

It should be noticed that the optimal policy of the decision maker in this adversarial setting is generally randomized. That is, the decision maker has to select an action at random by following some distribution. The reason is that if the action the decision maker takes is deterministic and can be predicted by Nature, then Nature can consistently "give" the decision maker a low reward for the selected arm while "giving" a high reward to all other arms, leading to a linear regret.

There are some interesting relationships between the non-stochastic bandit problem and prediction with expert advice, universal prediction, and learning in games (Cesa-Bianchi & Lugosi, 2006).

The Exploratory *k*-Armed Bandit Problem

This bandit variant emphasizes efficient exploration rather than on the exploration–exploitation tradeoff. As in the stochastic bandit problem, the decision maker is given access to k arms where each arm is associated with an independent and identically distributed random variable with unknown statistics. The decision maker's goal is to identify the "best" arm. That is, the decision maker wishes to find the arm with the highest expected reward as quickly as possible.

The exploratory bandit problem is a sequential hypothesis testing problem but with the added complication that the decision maker can choose where to sample next, making it among the simplest active learning problems. In the context of the probably approximate correct (PAC) setup, it was shown (Mannor & Tsitsiklis, 2004) that finding the ε-optimal arm (that is, an arm whose expected reward is lower than that of the best arm by at most ε) with probability of at least $1 - \delta$ requires

$$O\left(\frac{k}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)\right)$$

samples on expectation. Moreover, this bound can be obtained (up to multiplicative constants) via an algorithm known as median elimination.

Bandit analyses such as these have played a key role in understanding the efficiency of ►reinforcement-learning algorithm as well.

Cross References

►Active Learning
►Associative Bandit Problems
►Dynamic Programming
►Machine Learning in Games
►Markov Processes
►PAC Learning
►Reinforcement Learning

Recommended Reading

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2002). The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing, 32*(1), 48–77.

Berry, D., & Fristedt, B. (1985). *Bandit problems: Sequential allocation of experiments*. London/New York: Chapman and Hall.

Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. New York: Cambridge University Press.

Gittins, J. C. (1989). *Multi-armed bandit allocation indices*. New York: Wiley.

Gittins, J., & Jones, D. (1972). A dynamic allocation index for sequential design of experiments. In *Progress in statistics, European Meeting of Statisticians* (Vol. 1, pp. 241–266).

Lai, T. L., & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics, 6*, 4–22.

Mannor, S., & Tsitsiklis, J. N. (2004). The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research, 5*, 623–648.

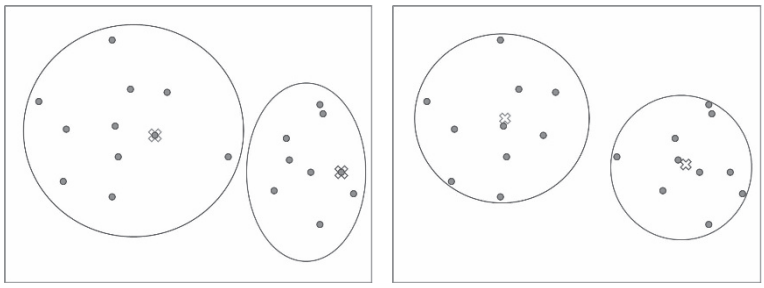
Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society, 55*, 527–535.

K-Means Clustering

Xin Jin, Jiawei Han
University of Illinois at Urbana-Champaign
Urbana, IL, USA

K-means (Lloyd, 1957; MacQueen, 1967) is one of the most popular clustering methods. Algorithm 1 shows the procedure of *K*-means clustering. The basic idea is: Given an initial but not optimal clustering, relocate each point to its new nearest center, update the clustering centers by calculating the mean of the member points, and repeat the relocating-and-updating process until convergence criteria (such as predefined number of iterations, difference on the value of the distortion function) are satisfied.

The task of initialization is to form the initial K clusters. Many initializing techniques have been proposed, from simple methods, such as choosing the first K data points, Forgy initialization (randomly choosing K data points in the dataset) and Random partitions



***K*-Means Clustering. Figure 1.** ***K*-Means clustering example ($K = 2$). The center of each cluster is marked by "*x*"**

(dividing the data points randomly into K subsets), to more sophisticated methods, such as density-based initialization, Intelligent initialization, Furthest First initialization (FF for short, it works by picking the first center point randomly, then adding more center points which are furthest from existing ones), and subset furthest-first (SFF)initialization. For more details, refer to paper Steinley and Brusco (2007) which provides a survey and comparison of over 12 initialization methods.

Figure 1 shows an example of K-means clustering on a set of points, with $K = 2$. The clusters are initialized by randomly selecting two points as centers.

Complexity analysis. Let N be the number of points, D the number of dimensions, and K the number of centers. Suppose the algorithm runs I iterations to converge. The space complexity of K-means clustering algorithm is $O(N(D+K))$. Based on the number of distance calculations, the time complexity of K-means is $O(NKI)$.

Algorithm 1 K-means clustering algorithm

Require: K, number of clusters; D, a data set of N points
Ensure: A set of K clusters

1. Initialization.
2. **repeat**
3. **for** each point p in D **do**
4. find the nearest center and assign p to the corresponding cluster.
5. **end for**
6. update clusters by calculating new centers using mean of the members.
7. **until** stop-iteration criteria satisfied
8. **return** clustering result.

Recommended Reading

Lloyd, S. P. (1957). Least squares quantization in PCM. Technical Report RR-5497, Bell Lab, September 1957.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In L. M. Le Cam & J. Neyman (Eds.), *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297). California: University of California Press.

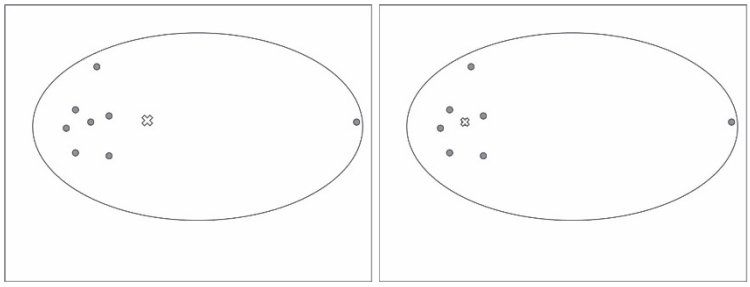
Steinley, D., & Brusco, M. J. (2007). Initializing k-means batch clustering: A critical evaluation of several techniques. *Journal of Classification, 24*(1), 99–121.

K-Medoids Clustering

Xin Jin, Jiawei Han
University of Illinois at Urbana-Champaign
Urbana, IL, USA

The K-means clustering algorithm is sensitive to outliers, because a mean is easily influenced by extreme values. K-medoids clustering is a variant of K-means that is more robust to noises and outliers. Instead of using the mean point as the center of a cluster, K-medoids uses an actual point in the cluster to represent it. Medoid is the most centrally located object of the cluster, with minimum sum of distances to other points. Figure 1 shows the difference between mean and medoid in a 2-D example. The group of points in the right form a cluster, while the rightmost point is an outlier. Mean is greatly influenced by the outlier and thus cannot represent the correct cluster center, while medoid is robust to the outlier and correctly represents the cluster center.

Partitioning around medoids (PAM) (Kaufman & Rousseeuw, 2005) is a representative K-medoids clustering method. The basic idea is as follows: Select K representative points to form initial clusters, and then



(a) Mean (b) Medoid

***K*-Medoids Clustering. Figure 1. Mean vs. medoid in 2-D space. In both figures (a) and (b), the group of points in the right form a cluster and the rightmost point is an outlier. The red point represents the center found by mean or medoid**

repeatedly moves to better cluster representatives. All possible combinations of representative and nonrepresentative points are analyzed, and the quality of the resulting clustering is calculated for each pair. An original representative point is replaced with the new point which causes the greatest reduction in distortion function. At each iteration, the set of best points for each cluster form the new respective medoids.

The time complexity of the PAM algorithm is $O(K(N-K)^2 I)$. PAM is not scalable for large dataset, and some algorithms have been proposed to improve the efficiency, such as Clustering LARge Applications (CLARA) (Kaufman & Rousseeuw, 2005) and Clustering Large Applications based upon RANdomized Search (CLARANS) (Ng & Han, 2002).

Recommended Reading

Kaufman, L., & Rousseeuw, P. J. (2005). *Finding groups in data: An introduction to cluster analysis (Wiley series in probability and statistics)*. New York: Wiley-Interscience.

Ng, R. T., & Han, J. (2002). Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering, 14*(5), 1003–1016.

K-Way Spectral Clustering

Xin Jin, Jiawei Han
University of Illinois at Urbana-Champaign
Urbana, IL, USA

In spectral clustering (Luxburg, 2007), the dataset is represented as a similarity graph $G=(V,E)$. The vertices represent the data points. Two vertices are connected if the similarity between the corresponding data points is larger than a certain threshold, and the edge is weighted by the similarity value. Clustering is achieved by choosing a suitable partition of the graph that each group corresponds to one cluster.

A good partition (i.e., a good clustering) is that the edges between different groups have overall low weights and the edges within a group have high weights, which indicates that the points in different clusters are dissimilar from each other and the points within the same cluster are similar to each other. One basic spectral clustering algorithm finds a good partition in the following way:

Given a set of data points P and the similarity matrix S, where S_{ij} measures the similarity between points $i,j \in P$, form a graph. Build a Laplacian matrix L of the graph,

$$L = I - D^{-1/2} S D^{-1/2}, \tag{1}$$

where D is the diagonal matrix

$$D_{ii} = \sum_j S_{ij}. \tag{2}$$

Find the eigenvalues and eigenvectors of the matrix L, map the vertices to corresponding components and form clusters based on the embedding space.

The methods to find K clusters include recursive bipartitioning and clustering multiple eigenvectors. The former technique is inefficient and unstable. The latter approach is more preferable because it is able to prevent instability due to information loss.

Recommended Reading

Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing, 17*(4), 395–416.

Kernel Density Estimation

►Density Estimation

Kernel Matrix

Synonyms

Gram matrix

Definition

Given a kernel function $k : X \times X \to$ and patterns $x_1, \ldots, x_m \in X$, the $m \times m$ matrix K with elements $K_{ij} := k(x_i, x_j)$ is called the kernel matrix of k with respect to $x_1, \ldots, x_m$.

Kernel Methods

Xinhua Zhang
Australian National University, Canberra, Australia
NICTA London Circuit, Canberra, Australia

Definition

Kernel methods refer to a class of techniques that employ positive definite kernels. At an algorithmic level, its basic idea is quite intuitive: implicitly map objects to high-dimensional feature spaces, and then directly specify the inner product there. As a more principled interpretation, it formulates learning and estimation problems in a reproducing kernel Hilbert space, which is advantageous in a number of ways:

- It induces a rich feature space and admits a large class of (nonlinear) functions.
- It can be flexibly applied to a wide range of domains including both Euclidean and non-Euclidean spaces.
- Searching in this infinite-dimensional space of functions can be performed efficiently, and one only needs to consider the finite subspace expanded by the data.
- Working in the linear spaces of function lends significant convenience to the construction and analysis of learning algorithms.

Motivation and Background

Over the past decade, kernel methods have gained much popularity in machine learning. Linear estimators have been popular due to their convenience in analysis and computation. However, nonlinear dependencies exist intrinsically in many real applications, and are indispensable for effective modeling. Kernel methods can sometimes offer the best of both aspects. The reproducing kernel Hilbert space provides a convenient way to model nonlinearity, while the estimation is kept linear. Kernels also offer significant flexibility in analyzing generic non-Euclidean objects such as graphs, sets, and dynamic systems. Moreover, kernels induce a rich function space where functional optimization can be performed efficiently. Furthermore, kernels have also been used to define statistical models via exponential families or Gaussian processes, and can be factorized by graphical models. Indeed, kernel methods have been widely used in almost all tasks in machine learning.

The reproducing kernel was first studied by Aronszajn (1950). Poggio and Girosi (1990) and Wahba (1990) used kernels for data analysis and Boser, Guyon, and Vapnik (1992) incorporated kernel function into the maximum margin models. Schölkopf, Smola, and Müller (1998) first used kernels for principal component analysis.

Theory

Positive semi-definite kernels are the most commonly used type of kernels, and its motivation is as follows. Given two objects x_1, x_2 from a space $\mathcal{X}$, which is not necessarily Euclidean, we map them to a high-dimensional feature space via $\phi(x_1)$ and $\phi(x_2)$, and then compute the inner products there by $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$. In many algorithms, the set $\{x_i\}$ influences learning only via inner products between x_i and x_j, hence it is sufficient to specify $k(x_1, x_2)$ directly without explicitly defining ϕ. This leads to considerable savings in computation, when ϕ ranges in high-dimensional spaces or even infinite-dimensional spaces. Clearly, the function k must satisfy some conditions. For example, as a necessary condition, for any finite number of examples $x_1, \ldots, x_n$ from $\mathcal{X}$, the matrix

$$K := (k(x_i, x_j))_{i,j} = (\phi(x_1), \ldots, \phi(x_n))^\top (\phi(x_1), \ldots, \phi(x_n))$$

must be positive semi-definite. Surprisingly, this turns out to be a sufficient condition as well, and hence we define the positive semi-definite kernels.

Definition 1 (Positive semi-definite kernels) *Let $\mathcal{X}$ be a nonempty set. A function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is called a positive semi-definite kernel if for any $n \in \mathbb{N}$ and $x_1, \ldots, x_n \in \mathcal{X}$, the Gram matrix $K := (k(x_i, x_j))_{i,j}$ is symmetric and positive semi-definite (psd).*

Reproducing Kernel Hilbert Space

Given a psd kernel k, we are able to construct a map ϕ from $\mathcal{X}$ to an inner product space $\mathcal{H}$, such that $\langle \phi(x_1), \phi(x_2) \rangle = k(x_1, x_2)$. The image of x under ϕ is just a function $\phi(x) := k(x, \cdot)$, where $k(x, \cdot)$ is a function of $\cdot$, assigning the value $k(x, x')$ for any $x' \in \mathcal{X}$. To define inner products between functions, we need to construct an inner product space $\mathcal{H}$ that contains $\{k(x, \cdot) : x \in \mathcal{X}\}$. First, $\mathcal{H}$ must contain the linear combinations $\{\sum_{i=1}^{n} \alpha_i k(x_i, \cdot) : n \in \mathbb{N}, x_i \in \mathcal{X}, \alpha_i \in \mathbb{R}\}$. Then, we endow it with an inner product as follows. For any $f, g \in \mathcal{H}$ and $f = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot), g = \sum_{j=1}^{m} \beta_j k\left(x'_j, \cdot\right)$, define

$$\langle f, g \rangle := \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k\left(x_i, x'_j\right),$$

and it is easy to show that this is well defined (independent of the expansion of f and g). Using the induced norm, we can complete the space and thus get a Hilbert space $\mathcal{H}$, which is called reproducing kernel Hilbert space (RKHS). The term "reproducing" is because for any function $f \in \mathcal{H}$, $\langle f, k(x, \cdot) \rangle = f(x)$.

Properties of psd Kernels

Let $\mathcal{X}$ be a nonempty set and $k_1, k_2, \ldots$ be arbitrary psd kernels on $\mathcal{X} \times \mathcal{X}$. Then

- The set of psd kernels is a closed convex cone, that is, (a) if $\alpha_1, \alpha_2 \geq 0$, then $\alpha_1 k_1 + \alpha_2 k_2$ is psd; (b) if $k(x, x') := \lim_{n \to \infty} k_n(x, x')$ exists for all x, x', then k is psd.
- The pointwise product $k_1 k_2$ is psd.
- Assume for $i = 1, 2$, k_i is a psd kernel on $\mathcal{X}_i \times \mathcal{X}_i$, where $\mathcal{X}_i$ is a nonempty set. Then the tensor product $k_1 \otimes k_2$ and the direct sum $k_1 \oplus k_2$ are psd kernels on $(\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2)$.

Example Kernels

One of the key advantage of kernels lies in its applicability to a wide range of objects.

Euclidean spaces: In $\mathbb{R}^n$, popular kernels include linear kernel $k(x_1, x_2) = \langle x_1, x_2 \rangle$, polynomial kernels $k(x_1, x_2) = (\langle x_1, x_2 \rangle + c)^d$ where $d \in \mathbb{N}$ and $c \geq 0$, Gaussian RBF kernels $k(x_1, x_2) = \exp(-\gamma \|x_1 - x_2\|^2)$ where $\gamma > 0$, and Laplacian RBF kernels $k(x_1, x_2) = \exp(-\gamma \|x_1 - x_2\|)$. Another useful type of kernels on Euclidean spaces is the spline kernels.

Convolution kernels: Haussler (1999) investigated how to define a kernel between composite objects by building on the similarity measures that assess their respective parts. It needs to enumerate all possible ways to decompose the objects, hence efficient algorithms like dynamic programming are needed.

Graph kernels: Graph kernels are available in two categories: between graphs and on a graph. The first type is similar to convolution kernels, which measures the similarity between two graphs. The second type defines a metric between the vertices, and is generally based on the graph Laplacian. By applying various transform functions to the eigenvalue of the graph Laplacian, various smoothing and regularization effects can be achieved.

Fisher kernels: Kernels can also be defined between probability densities $p(x|\theta)$. Let $U_\theta(x) = -\partial_\theta \log p(x|\theta)$ and $I = \mathbb{E}_x \left[U_\theta(x) U_\theta^\top(x)\right]$ be the Fisher score and Fisher information matrix respectively. Then the normalized and unnormalized Fisher kernels are defined by

$$k(x, x') = U_\theta^\top(x) I^{-1} U_\theta(x') \quad \text{and}$$
$$k(x, x') = U_\theta^\top(x) U_\theta(x'),$$

respectively. In theory, estimation using normalized Fisher kernels corresponds to regularization on the $L_2(p(\cdot|\theta))$ norm. And in the context of exponential families, the unnormalized Fisher kernels are identical to the inner product of sufficient statistics.

Kernel Function Classes

Many machine learning algorithms can be posed as functional minimization problems, and the RKHS is chosen as the candidate function set. The main advantage of optimizing over an RKHS originates from the representor theorem.

Theorem 2 (Representor theorem) *Denote by* $\Omega : [0,\infty) \mapsto \mathbb{R}$ *a strictly monotonic increasing function, by* $\mathcal{X}$ *a set, and by* $c : (\mathcal{X} \times \mathbb{R}^2)^n \mapsto \mathbb{R} \cup \{\infty\}$ *an arbitrary loss function. Then each minimizer* $f \in \mathcal{H}$ *of the regularized risk functional*

$$c((x_1,y_1,f(x_1)),\ldots,(x_n,y_n,f(x_n))) + \Omega\left(\|f\|_{\mathcal{H}}^2\right) \quad (1)$$

admits a representation of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x).$$

The representer theorem is important in that although the optimization problem is in an infinite-dimensional space $\mathcal{H}$, the solution is guaranteed to lie in the span of n particular kernels centered on the training points.

The objective (1) is composed of two parts: the first part measures the loss on the training set $\{x_i, y_i\}_{i=1}^n$, which depends on f only via its value at x_i. The second part is the regularizer, which encourages small RKHS norm of f. Intuitively, this regularizer penalizes the complexity of f and prefers smooth f. When the kernel k is translation invariant, that is, $k(x_1, x_2) = h(x_1 - x_2)$, Smola, Schölkopf, and Müller (1998) showed that $\|f\|^2$ is related to the Fourier transform of h, with more penalty imposed on the high frequency components of f.

Applications

Kernels have been applied to almost all branches of machine learning.

Supervised Learning

One of the most well-known applications of kernel method is the SVM for binary classification. Its primal form can be written as

$$\begin{aligned} &\underset{w,b,\xi}{\text{minimize}}\ \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n}\xi_i, \\ &\text{s.t. } y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i, \text{ and } \xi_i \geq 0,\ \forall i. \end{aligned}$$

Its dual form can be written as

$$\begin{aligned} &\underset{\alpha_i}{\text{minimize}}\ \frac{1}{2\lambda}\sum_{i,j} y_i y_j \langle x_i, x_j\rangle \alpha_i \alpha_j - \sum_i \alpha_i, \\ &\text{s.t. } \sum_i y_i \alpha_i = 0,\ \alpha_i \in [0, n^{-1}],\ \forall i. \end{aligned}$$

Clearly, this can be extended to feature maps and kernels by setting $k(x_i, x_j) = \langle x_i, x_j\rangle$. The same trick can be applied to other algorithms like ν-SVM, regression, density estimation, etc. For multi-class classification and structured output classification where the possible label set $\mathcal{Y}$ can be large, kernel maximum margin machines can be formulated by introducing a joint kernel on pairs of (x_i, y) $(y \in \mathcal{Y})$, that is, the feature map takes the tuple (x_i, y). Letting $\Delta(y_i, y)$ be the discrepancy between the true label y_i and the candidate label y, the primal form is

$$\begin{aligned} &\underset{w,\xi_i}{\text{minimize}}\ \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n}\xi_i, \\ &\text{s.t. } \langle w, \phi(x_i, y_i) - \phi(x_i, y)\rangle \geq \Delta(y_i, y) - \xi_i,\ \forall\, i, y, \end{aligned}$$

and the dual form is

$$\begin{aligned} \underset{\alpha_{i,y}}{\text{minimize}}\ &\frac{1}{2\lambda}\sum_{(i,y),(i',y')} \alpha_{i,y}\alpha_{i',y'}\langle \phi(x_i, y_i) \\ &\quad - \phi(x_i, y), \phi(x_{i'}, y_{i'}) - \phi(x_{i'}, y')\rangle \\ &\quad - \sum_{i,y} \Delta(y_i, y)\alpha_{i,y} \\ \text{s.t.}\quad &\alpha_{i,y} \geq 0,\ \forall\, i, y; \quad \sum_y \alpha_{i,y} = \frac{1}{n},\ \forall i. \end{aligned}$$

Again all the inner products $\langle \phi(x_i, y), \phi(x_{i'}, y')\rangle$ can be replaced by the joint kernel $k((x_i, y), (x_{i'}, y'))$. Further factorization using graphical models are possible (see Taskar, Guestrin, & Koller, 2004). Notice when $\mathcal{Y} = \{1, -1\}$, setting $\phi(x_i, y) = y\phi(x_i)$ recovers the binary SVM formulation. Effective methods to optimize the dual objective include sequential minimal optimization, exponentiated gradient (Collins, Globerson, Koo, Carreras, & Bartlett, 2008), mirror descent, cutting plane, or bundle methods (Smola, Vishwanathan, & Le, 2007).

Unsupervised Learning

Data analysis can benefit from modeling the distribution of data in feature space. There we can still use the rather simple linear methods, which gives rise to nonlinear methods on the original data space. For example, the principal components analysis (PCA) can be extended to Hilbert spaces (Schölkopf et al., 1998),

which allows for image denoising, clustering, and nonlinear dimensionality reduction.

Given a set of data points $\{x_i\}_{i=1}^n$, PCA tries to find a direction d such that the projection of $\{x_i\}$ to d has the maximal variance. Mathematically, one solves:

$$\max_{d:\|d\|=1} \mathrm{Var}\{\langle x_i, d\rangle\} \iff \max_{d:\|d\|=1} d^\top \left(\frac{1}{n}\sum_i x_i x_i^\top - \frac{1}{n^2}\sum_{ij} x_i x_j^\top\right) d,$$

which can be solved by finding the maximum eigenvalue of the variance of $\{x_i\}$. Along the same line, we can map the examples to the RKHS and find the maximum variance projection direction again. Here we first center the data, that is, let the feature map be $\tilde{\phi}(x_i) = \phi(x_i) - \frac{1}{n}\sum_j \phi(x_j)$, and define a kernel $\tilde{k}$ based on the centered feature. So we have $\sum_{j=1}^n \tilde{K}_{ij} = 0$ for all i. Now the objective can be written as

$$\max_{f:\|f\|_{\mathcal{H}}=1} \mathrm{Var}\left\{\left\langle \tilde{\phi}(x_i), f\right\rangle_{\mathcal{H}}\right\} \iff \max_{f:\|f\|=1} \mathrm{Var}\{f(x_i)\} \iff \max_{f:\|f\|\le 1} \mathrm{Var}\{f(x_i)\}. \quad (2)$$

Treat the constraint $\|f\| \le 1$ as an indicator function $\Omega(\|f\|^2)$ where $\Omega(x) = 0$ if $x \le 1$ and ∞ otherwise. Then the representer theorem can be invoked to guarantee that the optimal solution is $f = \sum_i \alpha_i \tilde{k}(x_i, \cdot)$ for some $\alpha_i \in \mathbb{R}$. Plugging it into (2), the problem becomes $\max_{\alpha:\alpha^\top \tilde{K}\alpha=1} \alpha^\top \tilde{K}^2 \alpha$. To get necessary conditions for optimality, we write out the Lagrangian $L = \alpha \tilde{K}^2 \alpha - \lambda(\alpha \tilde{K} \alpha - 1)$. Setting to 0 the derivative over α, we get

$$\tilde{K}^2 \alpha = \lambda \tilde{K} \alpha. \quad (3)$$

Therefore $\alpha^\top \tilde{K}^2 \alpha = \lambda$. Although (3) does not guarantee that α is an eigenvector of $\tilde{K}$, one can show that for each λ satisfying (3) there exists an eigenvector α of $\tilde{K}$ such that $\tilde{K}\alpha = \lambda\alpha$. Hence, it is sufficient to study the eigensystem of $\tilde{K}$ just like in the vanilla PCA. Once the optimal α_i^* is obtained, any data point x can be projected to $\sum_i \alpha_i^* \tilde{k}(x_i, x)$.

More applications of kernels in unsupervised learning can be found in canonical correlation analysis, independent component analysis (Bach & Jordan, 2002), kernelized independence criteria via Hilbert space embeddings of distributions (Smola, Gretton, Song, & Schölkopf, 2007), etc.

Cross References

►Principal Component Analysis
►Support Vector Machine

Further Reading

A survey paper on kernel methods up to year 2007 is Hofmann, Schölkopf, and Smola (2008). For an introduction to SVMs and kernel methods, read Cristianini and Shawe-Taylor (2000). More comprehensive treatment can be found in Schölkopf and Smola (2002), Shawe-Taylor and Cristianini (2004), and Steinwart and Christmann (2008). As far as applications are concerned, see Lampert (2009) for computer vision and Schölkopf, Tsuda, and Vert (2004) for bioinformatics. Finally, Vapnik (1998) provides the details on statistical learning theory.

Recommended Reading

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, *68*, 337–404.

Bach, F. R., & Jordan, M. I. (2002). Kernel independent component analysis. *Journal of Machine Learning Research*, *3*, 1–48.

Boser, B., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In D. Haussler, (Ed.), *Proceedings of the annual conference computational learning theory*, (pp. 144–152). Pittsburgh: ACM Press.

Collins, M., Globerson, A., Koo, T., Carreras, X., & Bartlett, P. (2008). Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks. *Journal of Machine Learning Research*, *9*, 1775–1822.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge: Cambridge University Press.

Haussler, D. (1999). *Convolution kernels on discrete structures* (Tech. Rep. UCS-CRL-99-10). University of California, Santa Cruz.

Hofmann, T., Schölkopf, B., & Smola, A. J. (2008). Kernel methods in machine learning. *Annals of Statistics*, *36*(3), 1171–1220.

Lampert, C. H. (2009). Kernel methods in computer vision. *Foundations and Trends in Computer Graphics and Vision*, *4*(3), 193–285.

Poggio, T., & Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, *78*(9), 1481–1497.

Schölkopf, B., & Smola, A. (2002). *Learning with Kernels*. Cambridge: MIT Press.

Schölkopf, B., Smola, A. J., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*, 1299–1319.

Schölkopf, B., Tsuda, K., & Vert, J.-P. (2004). *Kernel methods in computational biology*. Cambridge: MIT Press.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge: Cambridge University Press.

Smola, A. J., Gretton, A., Song, L., & Schölkopf, B. (2007). A Hilbert space embedding for distributions. In *International conference on algorithmic learning theory. LNAI* (Vol. 4754, pp. 13–31). Springer, Berlin, Germany.

Smola, A. J., Schölkopf, B., & Müller, K.-R. (1998). The connection between regularization operators and support vector kernels. *Neural Networks*, *11*(5), 637–649.

Smola, A., Vishwanathan, S. V. N., & Le, Q. (2007). Bundle methods for machine learning. In D. Koller, & Y. Singer, (Eds.), *Advances in neural information processing systems* (Vol. 20). Cambridge: MIT Press.

Steinwart, I., & Christmann, A. (2008). *Support vector machines. Information Science and Statistics*. Springer, New York.

Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin Markov networks. In S. Thrun, L. Saul, & B. Schölkopf, (Eds.), *Advances in neural information processing systems* (Vol. 16, pp. 25–32). Cambridge: MIT Press.

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley.

Wahba, G. (1990). *Spline models for observational data. CBMS-NSF regional conference series in applied mathematics* (Vol. 59). Philadelphia: SIAM.

Kernel Shaping

►Local Distance Metric Adaptation
►Locally Weighted Regression for Control

Kernel-Based Reinforcement Learning

►Instance-Based Reinforcement Learning

Kernels

►Gaussian Process

Kind

►Class

Knowledge Discovery

►Text Mining for Semantic Web

Kohonen Maps

►Self-Organizing Maps