General Remarks

l_2-euclidean distance	$\sqrt{\sum_{i=1}^{D}(x_i-y_i)^2}$
l_1-manhatten distance	$\sum_{i=1}^{D}\lvert x_i-y_i\rvert$
l_p-distance	$(\sum_{i=1}^{D}\lvert x_i-y_i\rvert^p)^{1/p}$
l_∞-distance	$\max_i \lvert x_i-y_i\rvert$
Mahalanobis norm	$\lvert\lvert w\rvert\rvert_G^2 = \lvert\lvert Gw\rvert\rvert_2^2$
Cosine-Distance	$\arccos \frac{x^Ty}{\lvert\lvert x\rvert\rvert_2 \lvert\lvert y\rvert\rvert_2}$
Jaccard-Distance	$1-\mathrm{Sim}(A,B) = 1-\frac{\lvert A\cap B\rvert}{\lvert A\cup B\rvert}$

Function Properties

Concave function A function f is called concave if $f(a+s)-f(a) \geq f(b+s)-f(b)\ \forall a \leq b, s>0$

Convex functions A function $f: S \to \mathbb{R}$, $S \subseteq \mathbb{R}^d$, is called convex if $\forall x, x' \in S, \lambda \in [0,1]$ it holds that

$$\lambda f(x) + (1-\lambda)f(x') \geq f(\lambda x + (1-\lambda)x')$$

Subgradients Given a convex not necessarily differentiable function f, a subgradient $g_x \in \nabla f(x)$ is the slope of a linear lower bound of f, tight at x, that is

$$\forall x' \in S \colon f(x') \geq f(x) + g_x^T(x'-x)$$

Locality Sensitive Hashing

Near-duplicate detection $\{x, x' \in X, x \neq x' \text{s.t.} d(x,x') \leq \epsilon\}$

(r,ϵ)-neighbour search Find all points with distance $\leq r$ and no points with distance $>(1+\epsilon)r$ from query q. Pick $(r,(1+\epsilon)\cdot r, p, q)$-sensitive family and boost.

Min-hashing $h(C) = h_\pi(C) = \min_{i:C(i)=1}\pi(i)$
$\pi(i) = h_{a,b}(i) = (a \cdot i + b \mod p) \mod N)$,
p prime (fixed) $>N$, N number of documents

$(d1,d2,p1,p2)$-sensitivity Assume $d_1 < d_2$, $p_1 > p_2$. Then

$$\forall x,y \in S: d(x,y) \leq d_1 \Rightarrow Pr[h(x)=h(y)] \geq p_1$$

$$\forall x,y \in S: d(x,y) \geq d_2 \Rightarrow Pr[h(x)=h(y)] \leq p_2$$

r-way AND (d_1, d_2, p_1^r, p_2^r)—more false negatives with bigger r

b-way OR $(d_1, d_2, 1-(1-p_1)^b, (1-p_2)^b)$—more false positives with bigger b

AND-OR cascade $(d_1, d_2, 1-(1-p_1^r)^b, 1-(1-p_2^r)^b)$

OR-AND cascade $(d_1, d_2, (1-(1-p_1)^b)^r, (1-(1-p_2)^b)^r)$

Hash Functions

Euclidean distance $h_{w,b}(x) = \lfloor(\frac{w^Tx-b}{a})\rfloor$ where $w \leftarrow \frac{w}{\lvert\lvert w\rvert\rvert_2}$, $w \sim \mathcal{N}(0,I)$, $w_i \sim \mathcal{N}(0,1)$, $b \sim Unif([0,a])$, yields $(a/2, 2a, 1/2, 1/3)$-sensitive

Cosine distance $\mathcal{H} = \{h(v) = \mathrm{sign}(w^Tv) \text{ for some } w \in \mathbb{R}^n \text{ s.t. } \lvert\lvert w\rvert\rvert_2 = 1\}$

Support Vector Machines

Linear Classifier $y = \mathrm{sign}\left(w^Tx+b\right)$. Train classifier $\sim$ find w. Want: $y_iw^Tx_i > 0\ \forall i$ for linearly separable data.

SVM SVM = Max margin linear classifier (with optional slack ξ)

$$\min_{w,\xi} \frac{1}{2}w^Tw + C\sum_{i=1}^{n}\xi_i \text{ s. t. } y_iw^Tx_i > 1-\xi_i\ \forall i$$

Support vectors (SV) are all data points on the margin and data points with non-zero slack

Equivalent primal SVM formulations

Regularized hinge loss formulation

$$\min_w w^Tw + C\sum_i \max(0, 1-y_iw^Tx_i)$$

Norm-constrained hinge loss minimization
$\min_w \sum_i \max(0,1-y_iw^Tx_i)$ s.t. $\lvert\lvert w\rvert\rvert_2 \leq \frac{1}{\sqrt{\lambda}}$

Strongly convex formulation
$\arg\ \min_w \frac{1}{T}\sum_{t=1}^{T}\left(\frac{\lambda}{2}\lvert\lvert w\rvert\rvert_2^2 + \max(0,1-y_tw^Tx_t)\right)$
s.t. $\lvert\lvert w\rvert\rvert_2 \leq \frac{1}{\sqrt{\lambda}}$

Small C, Big λ: Greater margin, more misclassification

Kernels

Dual SVM Formulation
$\max_{\alpha_{1:n}} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_jy_iy_jx_i^Tx_j$
s.t. $0 \leq \alpha_i \leq C$
$\Rightarrow$ optimal w: $w^* = \sum_i \alpha_i^*y_ix_i = \sum_{i\in\mathrm{SV}}\alpha_i^*y_ix_i$

Kernel trick Substitute inner product $x_i^Tx_j$ in dual formulation and in classification function with $k(x_i,x_j) = \phi(x_i)^T\phi(x_j)$, where $\phi(\cdot)$ is projection in higher dimensional data space (feature space). Classify new point x with

$$y = \mathrm{sign}(\sum_{i=1}^{n}\alpha_iy_ik(x_i,x))$$

Valid kernel functions A kernel is function $k: X \times X \to \mathbb{R}$ satisfying:

1. Symmetry: For any $x, x' \in X$,
$$k(x,x') = k(x',x)$$
2. Positive semi-definitenss: For any n, any set $S = \{x_1,..,x_n\} \subseteq X$, the kernel matrix $K[i,j] = k(x_i,x_j)$ must be positive semi-definite, i.e. all eigenvalues must be ≥ 0 ($x^TKx \geq 0\ \forall x$).

Random Features (Inverse Kernel Trick)

Shift-invariant kernel A kernel $k(x,y), x,y \in \mathbb{R}^d$ is called shift-invariant if $k(x,y) = k(x-y)$. Then the kernel has Fourier transform, such that:

$$k(x-y) = \int_{\mathbb{R}^d} p(w)\cdot e^{jw^T(x-y)}dw$$

where $p(w)$ is the Fourier transformation, i.e. we map $k(s)$ to another function $p(w)$.

Random fourier features (prerequisites) Interpret kernel as expectation

$$k(x-y) = \int_{\mathbb{R}^d} p(w)\cdot\underbrace{e^{jw^T(x-y)}}_{g(w)}dw = \mathbb{E}_{w,b}\left[z_{w,b}(x)z_{w,b}(y)\right]$$

where $z_{w,b}(x) = \sqrt{2}\cos\left(w^Tx+b\right)$, $b \sim U([0,2\pi])$, $w \sim p(w)$

Random fourier features (kernel approximation) The approximation goes as follows:

1. Draw $(w_1,b_1),...,(w_m,b_m)$, $w_i \sim p, b_i \sim U([0,2\pi])$ iid and fix them
2. Compute $z(x) = [z_{w_1,b_1}(x),...,z_{w_m,b_m}(x)]/\sqrt{m}$, so z is a feature map
3. $$z(x)^Tz(y) = \frac{1}{m}\sum_{i=1}^{m}z_{w_i,b_i}(x)\cdot z_{w_i,b_i}(y)$$
Now this is again an average. If $m \to \infty$, then $z(x)^Tz(y) \to \mathbb{E}_{w,b}(z_{w,b}(x)\cdot z_{w,b}(y)) = k(x-y)$ almost sure
4. After transformation, use SGD to train SVM (primal formulation) in transformed space

Online Convex Programming

Regret $R_T = (\sum_{t=1}^{T}l_t) - \min_{w\in S}\sum_{t=1}^{T}f_t(w)$

No-regret $\lim_{T\to\infty}\frac{R_T}{T} \to 0$

Online convex programming (OCP) An online convex programming algorithm is given by:

$$w_{t+1} = Proj_S(w_t - \eta_t\nabla f_t(w_t))$$

$$Proj_S(w) = arg\min_{w'\in S}\lvert\lvert w'-w\rvert\rvert_2$$

Regret for OCP

$$\frac{R_T}{T} \leq \frac{1}{\sqrt{T}}[||w_0 - w^*||_2^2 + ||\nabla f||_2^2]$$

where $||\nabla f||_2^2 = \sup_{w \in S, t \in \{1,...,T\}} ||\nabla f(w)||_2^2$

Parallel stochastic gradient descent
1. Split data into k subsets, k = number of machines
2. Each machine produces w_i on its subset
3. After T iterations, compute $w = \frac{1}{k}\sum_{i=1}^k w_i$

Active Learning

Uncertainty sampling Repeat until we can infer all remaining labels:
1. Assign uncertainty score $U_t(x)$ to each unlabeled data point: $U_t(x) = U\left(x|x_{1:t-1}, y_{1:t-1}\right)$
2. Greedily pick the most uncertain point and request label $x_t = \arg\ \max_x U_t(x)$ and retrain classifier

For SVM: $U_t(x) = \frac{1}{|w^T x|}$, where w^T obtained from points until $t-1$

Cost to pick m labels: $m \cdot n \cdot d + m \cdot C(m)$, where n = number of data points, d = dimensions, and $C(m)$ = cost to train classifier

Hashing a hyperplane query Draw $u, v \sim \mathcal{N}(0, I)$. Then resulting two-bit hash is:

$$h_{u,v}(a,b) = \left[\text{ sign } (u^T a),\ \text{sign } (v^T b)\right]$$

Now, define the hash family as:

$$h_{\mathcal{H}}(z) = \begin{cases} h_{u,v}(z,z) & \text{if } z \text{ is a database point vector} \\ h_{u,v}(z,-z) & \text{if } z \text{ is a query hyperplane vector} \end{cases}$$

Version space Set of all classifiers consistent with the data: $V(D) = \{w : \forall (x,y) \in D :\ \text{sign}(w^T x) = y\}$

Relevant version space $\hat{V}(D;U)$ describes all possible labelings h of all unlabeled data U that are still possible under some model w, or,

$$\hat{V}(D;U) = \{h : U \to \{+1,-1\} : \exists w \in V(D)$$
$$\forall x \in U : \text{sign}(w^T x) = h(x)\}$$

Generalized Binary Search (GBS) GBS works as follows:
1. Start with $D = \{\}$
2. While $|\hat{V}(D;U)| > 1$
   - For each unlabeled example x in U compute:

$$v^+(x) = |\hat{V}(D \cup \{(x,+)\};U)|$$
$$v^-(x) = |\hat{V}(D \cup \{(x,-)\};U)|$$

that is the number of labelings still left if x is $-/+$

3. Pick $x^* = \arg\min_x \max(v^-(x), v^+(x))$

Consider the following decision rules:

Max-min margin $\max_x \min\left(m^+(x), m^-(x)\right)$

Ratio margin $\max_x \min\left(\frac{m^+(x)}{m^-(x)}, \frac{m^-(x)}{m^+(x)}\right)$

where m denotes the margin of the resulting SVM.

Clustering

K-Means

Cost Function

$$L(\mu) = L(\mu_1, ..., \mu_k) = \sum_{i=1}^N \underbrace{\min_{j \in \{1,...,k\}} ||x_i - \mu_j||_2^2}_{d(\mu, x_i)}$$

Objective $\mu^* = \arg\ \min_\mu L(\mu)$

Algorithm The k-means algorithm incorporates the following two steps:
1. Assign each point x_i to closest center

$$z_i \leftarrow \arg \min_{j \in \{1,...,l\}} ||x_i - \mu_j^{(t-1)}||_2^2$$

2. Update center as mean of assigned data points:

$$\mu_j^{(t)} \leftarrow \frac{1}{n_j} \sum_{i: z_i = j} x_i$$

Online k-means algorithm

$$\frac{d}{d\mu_j} d(\mu, x_t) = \begin{cases} 0 & \text{if } j \notin \arg\ \min_i ||\mu_i - x_t|| \\ 2(u_j - x_t) & \text{else} \end{cases}$$

1. Initialize centers randomly
2. For $t = 1 : N$
   - Find $c = \arg\ \min ||\mu_j - x_t||_2$
   - $\mu_c = \mu_c + \eta_t(x_t - \mu_c)$
3. For convergence: $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$, e.g. $\eta_t = \frac{c}{t}$.

Coresets

Key idea Replace many points by one weighted representative, thus, obtain C $L_k(u;C) = \sum_{(w,x)\in C} w \cdot \min_j ||u_j - x||_2^2$

(k,ϵ)-coreset C is called a (k,ϵ)-coreset for D, if for all μ: $(1-\epsilon)L_k(\mu;D) \leq L_k(\mu;C) \leq (1+\epsilon)L_k(\mu;D)$

Bandits

ϵ-greedy The algorithm goes as follows:
1. Set $\epsilon_t = \mathcal{O}(\frac{1}{t})$
2. With probability ϵ_t: explore by picking uniformly at random
3. With probability $1-\epsilon_t$: exploit by picking arm with highest empirical mean

Regret: $R_T = \mathcal{O}(k \log(T))$

UCB1 & LinUCB

Hoeffding's inequality $\Pr(|\mu - \frac{1}{m}\sum_{t=1}^m X_t| \geq b) \leq 2 \cdot \exp(-2b^2 m)$

UCB/Mean update $UCB(i) = \hat{\mu_i} + \sqrt{\frac{2\ln t}{\eta_i}}$; $\hat{\mu_j} = \hat{\mu_j} + \frac{1}{\eta_j}(y_t - \hat{u}_j)$

Contextual bandits Reward is now $y_t = f(x_t, z_t) + \epsilon_t$ with z_t user features. For us: $f(x_i, z_t) = w_{x_i}^T z_t$

Hybrid model Reward is now $y_t = w_{x_t}^T z_t + \beta^T \phi(x_t, z_t) + \epsilon_t$

Evaluating bandits To evaluate, first obtain data log through pure exploration, and then reiterate:
1. Get next event $(x_t^{(1)}, .., x_t^{(k)}, z_t, a_t, y_t)$ from log
2. Use algorithm that is testing to pick a_t':
   - If $a_t' = a_t \Rightarrow$ Feed back reward y_t to the algorithm
   - Else ignore log line
3. Stop when T event have been kept

Submodular Functions

Submodulatity A function $F : 2^V \mapsto \mathbb{R}$ is called submodular iff for all $A \subseteq B, s \notin B$:

$$F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B)$$

Closedness properties The following closedness properties hold for submodular functions

Linear Combinations $F(A) = \sum_i \lambda_i F_i(A)$, $\lambda_i \geq 0$

Restriction $F'(S) = F(S \cap W)$

Conditiong $F'(S) = F(S \cup W)$

Reflection $F'(S) = F(V \setminus S)$

Misc 1 For $F_{1,2}(A)$, $\max\{F_1(A), F_2(A)\}$ or $\min\{F_1(A), F_2(A)\}$ **not** submodular in general.

Misc 2 $F(A) = g(|A|)$ where $g : \mathbb{N} \mapsto \mathbb{R}$, then F submodular iff g concave.

Lazy Greedy Lazy greedy algorithm for optimizing submodular functions,
1. Pick $s_1 = \arg\ \max_s F(A_i \cup \{s\}) - F(A_i)$
2. Keep an ordered list (priority queue) of marginal benefits δ_i form previous iteration ($\sim$ upper bound on gain).
3. Re-evaluate δ_i only for top element
4. δ_i stays on top, use it, otherwise re-sort.
5. Works because marginal gain is diminishing