1 Locality Sensitivity Hashing

Distance Function

$d : S \times S \to \mathbb{R}$ is a distance function if:
1. indentity is zero $\forall x \in S : d(x,x) = 0$
2. always positive $\forall x, x' \in S : d(x,x') > 0$
3. symmetric $\forall x, x' \in S : d(x,x') = d(x',x)$
4. triangle inequality $\forall x, x', x'' \in S : d(x,x'') \leq d(x,x') + d(x',x'')$

Curse of dimensionality

Fix $\epsilon < 0$, N. If data is truly high dimensional: $\lim_{D \to \infty} Pr[d_{max}(N,D) \leq (1+\epsilon) d_{min}(N,D)] = 1$

Shingling

Represent an document as a set of k-Shingles, which can keep trach of the order of the words.

Jaccard distance

Jaccard similarity: $\frac{|A \cap B|}{|A \cup B|} \in [0,1]$

Jaccard distance: $1 - \frac{|A \cap B|}{|A \cup B|} \in [0,1]$

Min-Hashing

Use random permutation π to reorder the rows:
$h(C)_\pi(C) = \min_{i:C(i)=1} \pi(i)$ which is the minimum row number in which permuted column contains a 1.

Property

$Pr[h_\pi(C_1) = h_\pi(C_2)] = Sim(C_1, C_2)$

Proof:
Define $\#\{1,1\} = a, \#\{1,0\} = b, \#\{0,1\} = c$
Assume we pick π u.a.r.
$Pr_\pi[h_\pi(C_1) = h_\pi(C_2)]$
$= Pr[\text{stop at}\{1,1\} | \text{ we stop at}\{1,0\},\{0,1\},\{1,1\}]$
$= \frac{a}{a+b+c} = Sim(C_1, C_2)$

Multiple Min-Hashing

To reduce misses, we can use multiple indep. random hash functions:
$s = Sim(C_1,C_2) = Pr[\text{hit with 1 function}]$
$\Rightarrow Pr[\text{miss with 1 function}] = 1 - s$
$\Rightarrow Pr[\text{miss with k functions}] = (1-s)^k$

Permutation function

representing perm. π as hash function h:
$\pi(i) = (a \cdot i + b \mod p) \mod N$, where $p > N$
$a \in_{u.r.t} \mathbb{R}, \quad b \in_{u.r.t} \mathbb{N}_0$
In this way we can store h very efficiently.

Min-Hashing Algorithm

Initialize all $M(i,c) = \infty$
1. For each column c:
2. For each row r:
3. if $c(r) = 1$:
4. for each hashfunction h_i:
5. $M(i,c) = min(h_i(r), M(i,c))$

False positive

By increasing the number of functions, we also increase the number of false positives.
TODO: insert the graphic

Signature matrix vs. similarity

Signature matrix represent similarity between documents.
Jaccard distance $\equiv \frac{\text{\# difference of columns}}{\text{\# hash functions}}$
but comparing all columns is very inefficient $O(N^2)$

Partitioning the signature matrix

TODO: add the graphic

Band Hashing

we have vectors $s = [s_1, ..., s_r]$
Pick r hash functions $h_1, ..., h_r$
$h_i(s_i) = (a_i s_i + b_i) \mod m$
$h(s) = \sum_{i=1}^r h_i(s_i) \mod m$

Analysis of partitioning

Fix a band C_j:
$Pr_h[h(B_{1,j}) = h(B_{2,j})] = s^r$
$Pr_h[h(B_{1,j}) \neq h(B_{2,j})] = Pr[\text{no collision on j-th band}] = 1 - s^r$
$Pr[\text{no collision on any band}] = (1-s^r)^b$
$Pr[\text{some collision}] = 1 - (1-s^r)^b = Pr[C_1, C_2 \text{ are candidates pair}]$
Now we can tune r and b to achieve our desired similarity threshold

General LHS

Consider a set S with distance d and a family F of hash functions $h : S \to B = \{1,..,n\}$
F is called $(d_1, d_2, p_1, _2)$-sensitive if:
$\forall x,y \in S : d(x,y) \leq d_1 \Rightarrow Pr[h(x) = h(y)] \geq p_1$
$\forall x,y \in S : d(x,y) \geq d_2 \Rightarrow Pr[h(x) = h(y)] \leq p_2$

r-way AND

decreases FP. Each member of F' consists of a *vector* of r hash functions from F.
For $h = [h_1, ..., h_r] \in F', h(x) = h(y) \Leftrightarrow h_i(x) = h_i(y)$ for all i
Theorem: F is $(d_1, d_2, p_1, _p 2)$-sensitive then F' is (d_1, d_2, p_1^r, p_2^r)-sensitive.

b-way OR

decreases FN. Each member of F' consists of a *vector* of b hash functions from F.
For $h = [h_1, ..., h_r] \in F', h(x) = h(y) \Leftrightarrow h_i(x) = h_i(y)$ for some i
Theorem: F is $(d_1, d_2, p_1, _p 2)$-sensitive then F' is $(d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-sensitive.

2 Support Vector Machine

Quadratic Programming

$\min_{w,\epsilon} w^T w + C \sum_{i=1}^n \epsilon_i$
$\text{s.t.} y_i w^T x_i \geq 1 - \epsilon_i$

Regularized hinge loss minimization

Given w, we can solve for optimal ϵ.
$\min_w w^T w + C \sum max(0, 1 - y_i w^T x_i)$

Norm-contrained hinge loss

$\min_w \sum_i \max(0, 1 - y_i w^T x_i)$, s.t. $\|w\|_2 \leq \frac{1}{\sqrt{\lambda}}$

Online Convex Programming

Regret: $R_T = \sum_{t=1}^T l_t - \min_w \sum_{t=1}^T f_t(w)$

OCP Regret: $R_T \leq \frac{\|S\|^2 \sqrt{T}}{2} + (\sqrt{T} - 1/2)\|\nabla f\|^2$ for $\eta_t = 1/\sqrt{t}$

If new point violates margin $y_t(w_t x_t + b) < 1$
Update $w_{t+1} = w_t - \eta_t \nabla f_t(w_t)$
Project $\min\{w, \frac{w}{\|w\|\lambda}\}$

Modifications

SGD: training samples picked at random.

PEGASOS: $\|_1\| \leq 1/\sqrt{\lambda}$ for each t: sample $A_t \subseteq X$, $A_t^+ = \{(x,y) \in A_t : y\langle_t x\rangle < 1\}$, $\nabla_t = \lambda_t - \eta_t/|A_t| \sum_{(,y) \in A_t^+} y$, $\eta_t = 1/(t\lambda)$, $'_t =_t - \eta_t \nabla_t$, $_{t+1} = \min\{1, (1/\sqrt{\lambda})/\|'_t\|\} *'_t$
Strongly convex loss function for better convergence.

Adagrad: $_{t+1} = \arg\min_{w \in S} \| - (_t - \eta_t^{-1} \nabla f_t(_t))\|_t$, $_t = (\sum_{\tau=1}^t \nabla f_\tau(_\tau) f_\tau(_\tau)^T)^{\frac{1}{2}}$, or just diag.

PSGD: randomly partition data to k machines which run SGD independently. After T iterations take weighted sum of obtained weights.
$w_T = \frac{1}{k} \sum_{i=1}^k w_i$. Parallelization helps if $k = O(1/\lambda)$

L1-ball projection: $Proj_S(w) = \arg\min_{\|w'\|_1 \leq c} \|w' - w\|_2$ using $w_i = \text{sign}(w_i)\max\{w_i - \beta, 0\}$

Feature selection

L1 regularization: replace $\|w\|_2$ with $\|w\|_1$ for sparse solutions.
modified projection: $\overline{w}_i = \text{sign}(w_i)\max\{|w_i| - \beta, 0\}$ where β is computed in linear time.

2.1 Random Fourier Features

For shift-invariant kernels ($k(x,y) = k(x-y)$)
$p(\omega) = \frac{1}{2\pi} \int e^{-j\omega'\delta} k(\delta)\Delta$
$\omega_i \sim p, b_i \sim U(0, 2\pi)$
$() \equiv \sqrt{2/m}[cos(\omega_1' + b_1) \ldots cos(\omega_m' + b_m)]$
In practice: pick random samples $S = \{\hat{x}_1 \ldots \hat{x}_n\} \subseteq X$
$_{SX ij} = k(\hat{x}_i, x_j)$, $_{SS ij} = k(\hat{x}_i, \hat{x}_j)$
approximate $=_{XS} {}_{SS}^{-1} {}_{SX}$

3 Active Learning

Pool-based AL

Obtain a large pool of unlabeled data. Selectively requests a few label, until we can infer all remaining data.

Uncertainty sampling

Given pool of n unlabeled examples
Repeat until we can infer all remaining labels:
1. Assign each unlabeled data x an score: $U_t(x) = U(x | x_{1:t-1}, y_{1:t-1})$
2. Greedily pick the most uncertain example: $x_t = \arg\max_x U_t(x)$

Example: SVM

Select point nearest to hyperplane decision boundary for labeling
$x^* = \arg\min_{x_i \in U} |w^T x_i|$
$U_t(x) = \frac{1}{|x_t^T x|}$

Sublinear time AL

We want to map hyperplane query directly to its nearest points
$h(w) \to \{x_1, ..., x_k\}$

Hashing hyperplane query

Let $h_{u,v}(a,b) = [h_u(a), h_v(b)] = [sign(u^T a), sign(v^T b)]$
where $u, v \sim N(0, I)$
Define hash familiy:
$$h_H(z) = \begin{cases} h_{u,v}(z,z) \text{ , if z is a point vector} \\ h_{u,v}(z,-z) \text{ , if z is a hyperplane vector} \end{cases}$$
LHS collision probability:
$Pr[h_H(w) = h_H(x)] = Pr[h_u(w) = h_u(x)] Pr[h_v[-w] = h_v(x)]$
$\frac{1}{4} - \frac{1}{\pi^2}(\omega_{x,w} - \frac{\pi}{2})^2$

Issues with uncertainty sampling

uncertain $\neq$ informative
We need to capture how much information we gain about the true calssifier for each label.

Version Space

Set of all Classifers consistent with the data:
$V(D) = \{w : \forall (x,y) \in D, sign(w^T x) = y\}$
Our idea is to shrink the version space as fast as possible

Relevant version space

Set of labeling of pool consistent with the data:
$\hat{V}(D,U) = \{h : U \to \{+1,-1\} : \exists w \in V(D),$
$\forall x \in U : sign(w^T x) = h(x)\}$

General binary search

1. Start with $D = \{\}$
2. While $|\hat{V}(D,U)| > 1$:
for each unlabeled example $x \in U$ compute:
$v^+(x) = |\hat{V}(D \cup \{(x,+)\}, U)|$
$v^-(x) = |\hat{V}(D \cup \{(x,-)\}, U)|$
Pich $x = \arg\min_x \max(v^-(x), v^+(x))$

Achieving balanced splits

We look at how labels affect the classifer.
For each possible label $\{+,-\}$ we calculate resulting SVM with margin m^+, m^-.
Define informativeness score:
Max-min margin: $\max_x \min(m^+(x), m^-(x))$
Ratio margin: $\max_x \min : \left(\frac{m^+(x)}{m^-(x)}, \frac{m^+(x)}{m^-(x)}\right)$

This method is comp. very expensive!

4 Clustering

K-means

Pick centers to min. avg. sq. distance.

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

$\mu^* = \arg\min_\mu L(\mu)$, which is NP-hard to solve

Lloyd's heuristic algorithm

Initialize $\mu^{(0)} = [\mu_1^{(0)}, ..., \mu_k^{(0)}]$

While note converged:

Assign each point to the closest center:

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

Update center:

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

Lloyd's properties

monotonically decrease in each iteration:

$$L(\mu^{(t)}) = \sum_{i=1}^{N} \min_j \|\mu_j^{(t)} - x_i\|_2^2$$

Online k-means

Initilaize center randomly

For $t = 1 : N$:

find $c = \arg\min_j = \|\mu_j - x_t\|_2$

set $\mu_c =_c + \eta_t(x_t - \mu_c)$

with $\sum_t \eta_t = \infty, \sum_t \eta_t^2 < \infty$, E.g. $\eta_t = \frac{c}{t}$

Coresets

C is called a (k, ϵ)-coreset for D, if for all μ:

$(1-\epsilon)L_k(\mu; D) \leq L_k(\mu; C) \leq (1+\epsilon)L_k(\mu; D)$

where $L_k(\mu; C) = \sum_{(w,x) \in C} w \cdot \min_{j \in \{1,...,k\}} \|\mu_j - x\|_2^2$

Importance sampling

$L(\mu; D) = \sum_{i=1}^{N} \min_j \|x_i - \mu_j\|_2^2 = \sum_{i=1}^{N} d(x_i, \mu)$

$= \sum_{i=1}^{N} \frac{1}{N}(N \cdot d(x_i, \mu)) = \sum_{i=1}^{N} p(i) \cdot \frac{d(x_i,\mu)}{p(i)}$

$= \mathbb{E}_{i \sim p}\left[\frac{d(x_i,\mu)}{p(i)}\right] = \mathbb{E}_{i \sim q}\left[\frac{d(x_i,\mu)}{q(i)}\right] = \frac{1}{m}\sum_{j=1}^{m} \frac{d(x_{i_j})}{q(i_j)}$

Importance weights

Sampling distribution $q(x)$

Weights $\gamma \propto \frac{1}{q}$

Sensitivity

$\sigma(x) = \max_\mu \frac{d(x,\mu)^2}{\frac{1}{N}\sum_{i=1}^{N} d(x_i,\mu)}$ (Hard to compute)

Sensitivty measures the wirst case effect fo data point x on any clustering cost.

Coreset Construction

1. D^2-sampling $p(x) = \frac{d(x,B)^2}{\sum_{x' \in X} d(x',B)^2}$
2. Importance sampling

$q(\alpha) \propto \frac{\alpha d(x,B)^2}{c_\phi} + 2\alpha \sum_{x' \in B_i} \frac{d(x',B)^2}{|B_i| c_\phi} + \frac{4|X|}{B_i}$

where $c_\phi = \frac{1}{|X|} \sum_{x \in X} d(x,B)^2$

Composition of Coresets

Merge: The union of (k,ϵ)-coresets os a (k,ϵ)-coreset.

Compress: A (k,δ)-coreset of a (k,ϵ)-coreset is a $(k, \epsilon + \delta + \epsilon\delta)$-coreset.

5 Bandits

k-armed bandits

Each arm i wins with probability μ_i. All drawns are independent given $\mu_1, ..., \mu_k$

Stochastic k-armed bandits

discrete set of k choices, each associated with unknow ind. prob. distr. P_i supported in $[0,1]$. We play for T rounds, in each we pick an arm i, and obtain an random sample y_i from P_i. Goal: $\max \sum_{t=1}^{T} y_t$

Regret

Let μ_i be the mean of P_i

Payoff of best decision: $\mu^* = \max_i \mu_i$

Let $i_1, ..., i_T$ be the sequence of decisions

Total expected regret: $R_T = \sum_{t=1}^{T} r_t$

Typical Goal: $\frac{R_T}{T} \to 0$ as $T \to \infty$

Exploration-Exploitation Tradeoff

Exploration: Gathering data about payoffs

Expoitation: Making choices based on data already gathered

Exporation-Exploitation Algo.

For $t = 1 : T$

Set $\epsilon_t = O\left(\frac{1}{t}\right)$

With prob. $\epsilon_t : Exploreuniformlyatrandom$

With prob. $1 - \epsilon_t$: Exploit picking highest mean payoff

Theorem: For suitable ϵ_t: $R_T = O(k \log T)$, which is no-regret $\lim_{T\to\infty} O\left(\frac{k \log T}{T}\right) \to 0$

Calculating confidence bounds

Suppose we fix arm i

Let $Y_i, ..., Y_m$ be the reward of arm i so far

Mean payoff: $\mu = \mathbb{E}[Y] \equiv \frac{1}{m}\sum_{l=1}^{m} Y_l = \hat{\mu}_m$

We want to obtain b, s.t. $P(|\mu - \hat{\mu}_m| \leq b)$ w.h.p

Hoeffding's inequality

Let $X_1, ..., X_k$ be i.i.d. RV taking values in $[0,1]$

$mu = \mathbb{E}[X], \hat{mu}_k = \frac{1}{k}\sum_{l=1}^{k} X_l$

then $P(|\mu - \hat{\mu}_k^{(i)}| \geq b) \leq 2\exp(-2b^2 k)$

UCB1 confidence bound

By using the Hoeffdings inequality:

$e^{2b^2 m} \leq \frac{\delta}{2} \Leftrightarrow b \geq \sqrt{\frac{1}{2m}\log(\frac{2}{\delta})}$

if we pick b, then $|\mu - \hat{\mu}_m^{(i)}| \leq b$ w.p $\geq 1 - \delta$

We want this to hold for all rounds!

$P(\cup_{i,m} E_{i,m}) \leq \sum_{i,m} P(E_{i,m}) = \sum_{i,m} \delta_m \leq \delta$

This can be fullfilled by choosing $\delta_m = \frac{c}{m^2}$

UCB1 algorithm

Set $\hat{\mu}_1, ..., \hat{\mu}_k = 0, n_1, ..., n_k = 0$

Try all arms once

For $t = 1 : T - k$

1. For each arm i: $UCB(i) = \hat{\mu}_i + \sqrt{\frac{2\log(t)}{n_i}}$
2. Pick arm $j = \arg\max_i UCB(i)$ and observe y_t
3. Set $n_j = n_j + 1, \hat{\mu}_j = \hat{\mu}_j + \frac{1}{n_j}(y_t - \hat{\mu}_j)$

Contextual bandits

In each round t do:

1. Observe context $z_t \in \mathcal{Z}$
2. Pick $x_t \in \mathcal{A}_t$
3. Observe $y_t = f(x_t, z_t) + \epsilon_t$
4. Incur regret $r_t = \max_{x \in A_t} f(x, z_t) - f(x_t, z_t)$

The comulative contextual regret: $R_t = \sum_{t=1}^{T} r_t$

Stochastic linear bandits

In each round t do:

1. Observe user features $z_t \in \mathbb{R}^d$
2. Pick recommendation $x_t \in \mathcal{A}_t = \{1, ..., k\}$
3. Observe CTR $y_t = w_{x_t}^T z_t + \epsilon_t$

Goal: minimize square loss:

$\hat{w} = \arg\min_w \sum_{t=1}^{m} (y_t - w^T z_t)^2 + \|w\|_2^2$

Closed form: $\hat{w}_i = (D_i^T D_i + I_d)^{-1} D_i^T y_i$,

where $D_i = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} \quad y_i = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$

$|_i^T z_t - w_i^T z^T| \leq \alpha\sqrt{z_t^T (D_i^T D_i + I)^{-1} z_t}$ with probability at least $1 - \delta$ as long as $\alpha = 1 + \sqrt{\log(2/\delta)/2}$

We set $M_{i,t} = D_i^T D_i + i_d$

LinUCB Algorithm

For $t = 1 : T$

1. Receive action set $\mathcal{A}_t$ and features z_t
2. For all $x \in \mathcal{A}_t$:

if x is new, $M_x = I, b_x = 0$

Set $\hat{w}_x = M_x^{-1} b_x$

Set $UCB(x) = \hat{w}_x^T z_t + \alpha\sqrt{z_t^T M_x^{-1} z_t}$

3. Recommend $x_t = \arg\max_{x \in A_t} UCB(x)$
4. Observe reward y_t
5. Set $M_x = M_x + z_t z_t^T, b_x = b_x + y_t z_t$

Hybrid model

Capture separate and shared effects: $y_t = w_i^T z_t + \beta^T \phi(x_i, z_t) + \eta_t$

$\phi(x_i, z_t) = vec(x_i z_t^T)$

Now need to estimate both w_i and β

rejection sampling

For $t = 1 : \infty$ 1. Get next event $(x_t^(1), ..., x_t^(k), z_t, a_t, y_t)$ from log

2. Use bandit algorithm to pick $\hat{a}_t$
3. If $\hat{a}_t = a_t$: Feedback reward y_t
4. Stop when T events have been kept

This approach is unbiased

Coarse to fine selection

Recommend article i that maximizes:

$\arg\max_i \hat{w}_i^T z_t + \alpha\sqrt{z_t^T M_t^{-1} z_t} + \beta\sqrt{z_t^T M_t^{-1} U \hat{M}_t^{-1} U^T z_t}$

6 Submodularity

Approximation guarantee

the greedy maximum coverage produces a A, where $F(A) \geq 1 - \frac{1}{e} \approx 64\%$ of optimal value. F must be nonneg. monotone and submodular.

Submodularity

F is submodular on V: $\forall A \subseteq B, s \notin B$

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

Closedness property

nonnegative linear combination:

$F(A) = \sum_i \lambda_i F_i(A), \lambda_i \geq 0$ is submodular

Restriction: $\hat{F}(S) = F(S \cap W)$, for a fix $W \in V$

Conditioning: $\hat{F}(S) = F(S \cup W)$, for a fix $W \in V$

Reflexion: $\hat{F}(S) = F(V \backslash S)$

Submodularity and Concavity

$g : \mathbb{N} \to \mathbb{R}, F(A) = g(|A|)$, then $F(A)$ submodular if and only if g is concave.

"Lazy" greedy algorithm

Key observation: Marginal benefits $\Delta(s|A_i) = F(A_i \cup \{s\}) - F(A_i)$ can never increase for a fix s.

1. evaluate all marginal benefits Δ_i and take the best
2. order the list of marginal benefits
3. Re-evaluate Δ_i for top element
4. If Δ_i stays on top, use it, otherwise resort.