Disclaimer

I wrote this to my best knowledge, however, no guarantees are given whatsoever.

Sources

If not noted differently, the source is the lecture slides and/or the accompanying book.

1 Approximate Retrieval

Nearest-Neighbor Find $x^* = \text{argmin}_{x \in X}\ d(x,y)$ given S, $y \in S$, $X \subseteq S$.

Near-Duplicate detection Find all $x,x' \in X$ with $d(x,x') \leq \epsilon$.

1.1 k-Shingling

Represent documents (or videos) as set of k-shingles (a. k. a. k-grams). *k-shingle*is a consecutive appearance of k characters/words.

Let there be N documents and C k-shingles.
Binary *shingle matrix* $M \in \{0,1\}^{CxN}$ where $M_{i,j}=1$ iff document j contains shingle i.

1.2 Distance functions

General $d : S \times S \to \mathbb{R}$ is a *distance function* iff $\forall x,x',x'' \in S$ it's positive definite except for $x = x'$ ($d(x,x') > 0 \iff x \neq x'$ and $d(x,x) = 0$), symmetric ($d(x,x') = d(x',x)$) and satisfies the Cauchy-Schwartz triangle inequality ($d(x,x'') \leq d(x,x') + d(x',x'')$).

L_r**-norm** $d_r(x,y) = ||x-y||_r = (\sum_i |x_i - y_i|^r)^{1/r}$. L_2 is *Euclidean.*

Cosine similarity $\text{Sim}_c(A,B) = \dfrac{A \cdot B}{||A|| \cdot ||B||}$.

Cosine distance $d_c(A,B) = \dfrac{\arccos(\text{Sim}_c(A,B))}{\pi}$.

Jaccard similarity $\text{Sim}_J(A,B) = \dfrac{|A \cap B|}{|A \cup B|}$.

Jaccard distance $d_J(A,B) = 1 - \text{Sim}_J(A,B) = 1 - \dfrac{|A \cap B|}{|A \cup B|}$.

1.3 LSH – local sensitive hashing

Key Idea: Similiar documents have similiar hash.
Note: Trivial for exact duplicates (hash-collisions → candidate pair).

Min-hash $h_\pi(C)$ Hash is the *minimum (i. e. first) row index* with a one after permutation: $h_\pi(C) = \min_{i,C(i)=1} \pi(i)$, given binary vector C and (random) permutation π.
Note: $\Pr_\pi[h_\pi(C_1) = h_\pi(C_2)] = \text{Sim}_J(C_1,C_2)$ if $\pi \in_{\text{u.a.r.}} S_{|C|}$.

Min-hash signature matrix $M_S \in [N]^{n \times C}$ with $M_S(i,c) = h_i(C_c)$ given n hash-fns h_i drawn randomly from a universal hash family.

Pseudo permutation h_π with $\pi(i) = (a \cdot i + b) \mod p \mod N$, N number of shingles, $p \geq N$ prime and $a,b \in_{\text{u.a.r.}} [p]$ with $a \neq 0$.
Instead of real permutations (slow, inefficient, large storage) use pseudo permutations as hash family. Pseudo permutations only need to store a and b.

Compute Min-hash signature matix M_S For all columns $c \in [C]$ and rows $r \in [N]$ with $C_c(r) = 1$, set $M_S(i,c) = \min\{h_i(C_c), M_S(i,c)\}$ for all hash functions h_i.

Banding as boosting Reduce FP/FN by AND/OR-boosting, respectively.
This is done by grouping the signature matrix into b bands of r rows each. A candidate pair matches in at least one band completely (check through normal hashing). This corresponds to a b-way OR after a r-way AND boosting.

Tradeoff FP/FN Favor false positives (more work) over false negatives (wrong result). Filter out false positives by checking signature matrix, shingles or even whole documents.

2 Supervised Learning

Linear classifier $y_i = \text{sign}(\boldsymbol{w}^T \boldsymbol{x}_i)$ assuming w goes through origin.

Homogeneous transform $\tilde{x} = [x,1]; \tilde{w} = [w,b]$, now w passes origin.

Kernels

Convex functin $f : S \to \mathbb{R}$ is convex iff $\forall x,x' \in S, \lambda \in [0,1], \lambda f(x) + (1-\lambda) f(x') \geq f(\lambda x + (1-\lambda)x')$, i. e. every segment lies above function. Equiv. bounded by linear fn at every point.

H-strongly convex f H-strongly convex iff $f(x') \geq f(x) + \nabla f(x)^T (x' - x) + \frac{H}{2} ||x' - x|_2^2$, i. e. bounded by quadratic fn (at every point).

2.1 Support vector machine (SVM)

SVM primal

Quadratic $\min_{\boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{w} + C \sum_i \xi_i$, s. t. $\forall i : \boldsymbol{y}_i \boldsymbol{w}^T x_i \geq 1 - \xi_i$, slack C.

Hinge loss $\min_{\boldsymbol{w}} \boldsymbol{w}^T \boldsymbol{w} + C \sum_i \max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i)$,
where $l(\boldsymbol{w}; \boldsymbol{x}_i, y_i) = \max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i)$ is the *hinge loss.*
Also written $\min_{\boldsymbol{w}} \lambda \boldsymbol{w}^T \boldsymbol{w} + C \sum_i l(\boldsymbol{w}; \boldsymbol{x}_i, y_i)$ with $\lambda = \frac{1}{C}$.

Norm-constrained $\min_{\boldsymbol{w}} \sum_i \max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i)$ s. t. $||\boldsymbol{w}||_2 \leq \frac{1}{\sqrt{\lambda}}$.

Lagrangian dual $\max_{\boldsymbol{\alpha}} \sum_i \alpha_i + \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$, $\alpha_i \in [0,C]$.
Apply kernel trick: $\max_{\boldsymbol{\alpha}} \sum_i \alpha_i + \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\alpha_i \in [0,C]$, prediction becomes $y = \text{sign}(\sum_{i=1}^n \alpha_i y_i k(x_i, x))$.

2.2 Convex Programming

Convex program $\min_{\boldsymbol{x}} f(\boldsymbol{x})$, s. t. $\boldsymbol{x} \in S$.

Online convex program (OCP) $\min_{\boldsymbol{w}} \sum_{t=1}^T f_t(\boldsymbol{w})$, s. t. $\boldsymbol{w} \in S$.

General regularized form $\min_{\boldsymbol{w}} \sum_{i=1}^n l(\boldsymbol{w}; \boldsymbol{x}_i, y_i) + \lambda R(\boldsymbol{w})$, where l is a (convex) loss function and R is the (convex) regularizer.

General norm-constrained form $\min_{\boldsymbol{w}} \sum_{i=1}^n l(\boldsymbol{w}; \boldsymbol{x}_i, y_i)$, s. t. $\boldsymbol{w} \in S_\lambda$, where l is the loss function and S_λ some (norm-)constraint. Note how this is a OCP.

Solving OCP Input *feasible set* $S \subseteq \mathbb{R}^d$ and *starting point* $\boldsymbol{w}_0 \in S$, given OCP $\min_{\boldsymbol{w}} \sum_{t=1}^T f_t(\boldsymbol{w})$, s. t. $\boldsymbol{w} \in S$. For round $t \in [T]$, pick (feasible pt) $\boldsymbol{w}_t \in S$, receive (convex) fn $f_t : S \to \mathbb{R}$, incur loss $l_t = f_t(\boldsymbol{w}_t)$. Regret $R_T = (\sum_{t=1}^T l_t) - \min_{\boldsymbol{w} \in S} \sum_{t=1}^T f_t(\boldsymbol{w})$.

Online SVM $||\boldsymbol{w}||_2 \leq \frac{1}{\lambda}$ (norm-constrained). For new point $\boldsymbol{x}_t$ classify $y_t = \text{sign}(\boldsymbol{w}_t^T \boldsymbol{x}_t)$, incur loss $l_t = \max(0, 1 - y_t \boldsymbol{w}_t^T \boldsymbol{x}_t)$, update $\boldsymbol{w}_t$ (see later). Best possible $L^* = \min_{\boldsymbol{w}} \sum_{t=1}^T \max(0, 1 - y_t \boldsymbol{w}^T \boldsymbol{x}_t)$, regret $R_t = \sum_{t=1}^T l_t - L^*$.

Online proj. gradient descent (OPGD) Update for online SVM: $w_{t+1} = \text{Proj}_S(w_t - \eta_t \nabla f_t(\boldsymbol{w}_t))$ with $\text{Proj}_S(\boldsymbol{w}) = \text{argmin}_{w' \in S} ||w' - w||_2$, gives regret bound $\frac{R_T}{T} \leq \frac{1}{\sqrt{T}} (||\boldsymbol{w}_0 - \boldsymbol{w}^*||_2^2 + ||\nabla f||_2^2)$.

For H-strongly convex fn set $\eta_t = \frac{1}{Ht}$ gives $R_t \leq \frac{||\nabla f||^2}{2H} (1 + \log T)$.

Stochastic PGD (SGD) Online-to-batch. Compute $\tilde{\boldsymbol{w}} = \frac{1}{T} \sum_{t=1}^T \boldsymbol{w}_t$. If data i. i. d.: exp. *error (risk)* $\mathbb{E}[L(\tilde{\boldsymbol{w}})] \leq L(\boldsymbol{w}^*) + R_T/T$, $L(\boldsymbol{w}^*)$ is best error (risk) possible.

PEGASOS OPGD w/ mini-batches on strongly convex SVM form. $\min_w \sum_{t=1}^T g_t(\boldsymbol{w})$, s.t. $||w||_2 \leq \frac{1}{\sqrt{t}}$, $g_t(\boldsymbol{w}) = \frac{\lambda}{2} ||\boldsymbol{w}||_2^2 + f_t(\boldsymbol{w})$.
g_t is λ-strongly convex, $\nabla g_t(\boldsymbol{w}) = \lambda \boldsymbol{w} + \nabla f_t(\boldsymbol{w})$.

Performance ϵ-accurate sol. with prob. $\geq 1 - \delta$ in runtime $O^*(\frac{d \cdot \log \frac{1}{\delta}}{\lambda \epsilon})$.

ADAGrad Adapt to geometry. *Mahalanobis norm* $||\boldsymbol{w}||_{\boldsymbol{G}} = ||\boldsymbol{G}\boldsymbol{w}||_2$. $w_{t+1} = \text{argmin}_{\boldsymbol{w} \in S} ||\boldsymbol{w} - (\boldsymbol{w}_t - \eta \boldsymbol{G}_t^{-1} \nabla f_t(\boldsymbol{w}))||_{\boldsymbol{G}_t}$. Min. regret with $G_t = (\sum_{\tau=1}^t \nabla f_\tau(\boldsymbol{w}_\tau) \nabla f_\tau(\boldsymbol{w}_\tau)^T)^{1/2}$. Easily inv'able matrix with $G_t = \text{diag}(...)$. $R_t \in O(\frac{||\boldsymbol{w}^*||_\infty}{\sqrt{T}} \sqrt{d})$, even better for sparse data.

ADAM Add 'momentum' term: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \mu \bar{g}_t$, $g_t = \nabla f_t(\boldsymbol{w})$, $\bar{g}_t = (1-\beta) g_t + \beta \bar{g}_{t-1}$, $\bar{g}_0 = 0$. Helps for dense gradients.

Parallel SGD (PSGD) Randomly partition to k (indep.) machines. Comp. $\boldsymbol{w} = \frac{1}{k} \sum_{i=1}^k \boldsymbol{w}_i$. $\mathbb{E}[\text{err}] \in O(\epsilon(\frac{1}{k\sqrt{\lambda}} + 1))$ if $T \in \Omega(\frac{\log \frac{k\lambda}{\epsilon}}{\epsilon \lambda})$. Suitable for MapReduce cluster, multi. passes possible.

Hogwild! Shared mem., no sync., sparse data. [. . .]

Implicit kernel trick Map $x \in \mathbb{R}^d \to \phi(x) \in \mathbb{R}^D \to z(x) \in \mathbb{R}^m$, $d \ll D, m \ll D$. Where $\phi(x)$ corresponds to a kernel $k(x,x') = \phi(x)^T \phi(x')$.

Random fourier features !TODO!

Nyström features !TODO!

3 Active Learning (semi-supervised)

Stream-based* Data point arrives online, decide if label needed.

Pool-based Unlabeled data-set given, (sequentially) request labels.

Uncertainty sampling

4 Unsupervised learning

5 Bandits