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

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

### 1.1 $k$-Shingling

Documents (or videos) as set of $k$-shingles (a. k. a. $k$-grams). *$k$-shingle* is consecutive appearance of $k$ chars/words.
Binary *shingle matrix* $M\in\{0,1\}^{C\times N}$ where $M_{i,j}=1$ iff document $j$ contains shingle $i$, $N$ documents, $C$ $k$-shingles.

### 1.2 Distance functions

**Def.** $d: S\times S\to\mathbb{R}$ is *distance function* iff pos. definite except $d(x,x)=0$ ($d(x,x')>0 \iff x\neq x'$), symmetric ($d(x,x')=d(x',x)$) and triangle inequality holds ($d(x,x'')\leq d(x,x')+d(x',x'')$).

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

**Cosine** $\text{Sim}_c(A,B)=\dfrac{A\cdot B}{||A||\cdot||B||}$, $d_c(A,B)=\dfrac{\cos^{-1}(\text{Sim}_c(A,B))}{\pi}$.

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

### 1.3 LSH – local sensitive hashing

*Key Idea:* Similiar documents have similiar hash.
*Note:* Trivial for exact duplicates (hash-collision $\to$ candidate pair).

**Min-hash-family** $h_\pi(C)$ **for Jaccard** Hash is the *min (i.e. first) non-zero permutated row index*: $h_\pi(C)=\min_{i,C(i)=1}\pi(i)$, bin. vec. $C$, rand. perm. $\pi$.
*Note:* $\text{Pr}_\pi[h_\pi(C_1)=h_\pi(C_2)]=\text{Sim}_J(C_1,C_2)$ if $\pi\in_{\text{u.a.r.}} S_{|C|}$.

**Min-hash $L_r$-norm:** Fix $a\in\mathbb{R}$. Random line $\boldsymbol{w}$ paritioned in buckets of length $a$. Project $\boldsymbol{x},\boldsymbol{y}$ onto $\boldsymbol{w}$, if in same bucket, $h_{\boldsymbol{w}}(x)=h_{\boldsymbol{w}}(y)$. In 2-dim. forms a $(a/2,2\cdot a,1/2,1/3)$-sensitive hash-family. In d-dim. there exists a $(d1,d2,p1,p2)$-sensitive family $\forall d1<d2$ with $p1>p2$.

**Min-hash cos.** $h_{\boldsymbol{w}}(\boldsymbol{x})=\text{sgn}(\boldsymbol{w}^T\boldsymbol{x})$. $\text{Pr}_{\boldsymbol{w}}[h_{\boldsymbol{w}}(x)=h_{\boldsymbol{w}}(y)]=1-\frac{\theta_{\boldsymbol{x},\boldsymbol{y}}}{\pi}$.

**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** $h_\pi$ 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 $a\neq 0$.
Use as universal hash family. Only store $a$ and $b$. Much more efficient.

**Compute signature matix** $M_S$ For column $c\in[C]$, row $r\in[N]$ with $C_c(r)=1$, $M_S(i,c)\leftarrow\min\{h_i(C_c),M_S(i,c)\}$ for all $h_i$.

$(d_1,d_2,p_1,p_2)$**-sensitivity** of $F=\{h_1,\ldots,h_n\}$: $\forall x,y\in S: d(x,y)\leq d_1 \Longrightarrow P[h(x)=h(y)]\geq p_1$ and $d(x,y)\geq d_2\Longrightarrow P[h(x)=h(y)]\leq p_2$.

$r$**-way AND** $h=[h_1,\ldots,h_r]$, $h(x)=h(y)\Leftrightarrow\forall i\ h_i(x)=h_i(y)$ is $(d_1,d_2,p_1^r,p_2^r)$-sensitive.

$b$**-way OR** $h=[h_1,\ldots,h_b]$, $h(x)=h(y)\Leftrightarrow\exists i\ h_i(x)=h_i(y)$ is $(d_1,d_2,1-(1-p_1)^b,1-(1-p_2)^b)$-sensitive.

**Banding as boosting** Reduce FP/FN by $b$-way OR after $r$-way AND. Group sig. matrix into $b$ bands of $r$ rows. CP match in at least one band (check by hashing). Result is $(d_1,d_2,1-(1-p_1^r)^b,1-(1-p_2^r)^b)$-sensitive.

**Tradeoff FP/FN** Favor FP (work) over FN (wrong). Filter FP by checking signature matrix, shingles or even whole documents.

## 2 Supervised Learning

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

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

**Kernel** $k$ is inner product in high-dim. space: $k(\boldsymbol{x},\boldsymbol{y})=\langle\phi(\boldsymbol{x}),\phi(\boldsymbol{y})\rangle$.
*shift-invariance* $k(\boldsymbol{x},\boldsymbol{y})=k(\boldsymbol{x}-\boldsymbol{y})$.
*Gaussian* $k(\boldsymbol{x}-\boldsymbol{y})=\exp(-||\boldsymbol{x}-\boldsymbol{y}||_2^2/h^2)$.

**Convex function** $f: S\to\mathbb{R}$ 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. $\forall i: y_i\boldsymbol{w}^T\boldsymbol{x}_i\geq 1-\xi_i$, slack $C$.
*Hinge loss* $\min_{\boldsymbol{w}}\lambda\boldsymbol{w}^T\boldsymbol{w}+\sum_i\max(0,1-y_i\boldsymbol{w}^T\boldsymbol{x}_i)$ 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 $y=\text{sgn}(\sum_{i=1}^n\alpha_i y_i k(\boldsymbol{x}_i,\boldsymbol{x}))$.

### 2.2 Convex Programming

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

**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. $\boldsymbol{w}\in S_\lambda$, $l$ is loss and $S_\lambda$ some (norm-)constraint. Note: This is an OCP.

**Solving OCP** *Feasible set* $S\subseteq\mathbb{R}^d$ and *start pt.* $\boldsymbol{w}_0\in S$, OCP (as above). Round $t\in[T]$: pick feasible pt. $\boldsymbol{w}_t$, get convex fn. $f_t$, incur $l_t=f_t(\boldsymbol{w}_t)$. 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-constr.). For new pt. $\boldsymbol{x}_t$ classify $y_t=\text{sgn}(\boldsymbol{w}_t^T\boldsymbol{x}_t)$, incur $l_t=\max(0,1-y_t\boldsymbol{w}_t^T\boldsymbol{x}_t)$, update $\boldsymbol{w}_t$ (see later). Best $L^*=\min_{\boldsymbol{w}}\sum_{t=1}^T\max(0,1-y_t\boldsymbol{w}^T\boldsymbol{x}_t)$, regret $R_t=\sum_{t=1}^T l_t-L^*$.

**Online proj. gradient descent (OPGD)** Update for online SVM: $\boldsymbol{w}_{t+1}=\text{Proj}_S(\boldsymbol{w}_t-\eta_t\nabla f_t(\boldsymbol{w}_t))$ with $\text{Proj}_S(\boldsymbol{w})=\text{argmin}_{\boldsymbol{w}'\in S}||\boldsymbol{w}'-\boldsymbol{w}||_2$, 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, $\eta_t=\frac{1}{Ht}$ 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. $||\boldsymbol{w}||_2\leq\frac{1}{\sqrt{t}}$, $g_t(\boldsymbol{w})=\frac{\lambda}{2}||\boldsymbol{w}||_2^2+f_t(\boldsymbol{w})$.
$g_t$ is $\lambda$-strongly convex, $\nabla g_t(\boldsymbol{w})=\lambda\boldsymbol{w}+\nabla f_t(\boldsymbol{w})$.
*Performance* $\epsilon$-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* $||\boldsymbol{w}||_{\boldsymbol{G}}=||\boldsymbol{G}\boldsymbol{w}||_2$. $\boldsymbol{w}_{t+1}=\text{argmin}_{\boldsymbol{w}\in S}||\boldsymbol{w}-(\boldsymbol{w}_t-\eta\boldsymbol{G}_t^{-1}\nabla f_t(\boldsymbol{w}))||_{\boldsymbol{G}_t}$. 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 $G_t=\text{diag}(\ldots)$. $R_t\in O(\frac{||\boldsymbol{w}^*||_\infty}{\sqrt{T}}\sqrt{d})$, 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** Given shift-invariant kernel $k$.
$p(\omega)=\frac{1}{2\pi}\int e^{-j\omega'\delta}k(\delta)\text{d}\Delta$
$\omega_i\sim p=$ eg Gaussian, $b_i\sim U(0,2\pi)$
$\boldsymbol{z}(\boldsymbol{x})\equiv\sqrt{2/m}[cos(\omega_1'\boldsymbol{x}+b_1)...cos(\omega_m'\boldsymbol{x}+b_m)]$

**Nyström features (need entire dataset)** In practice: pick random samples $S=\{\hat{\boldsymbol{x}}_1...\hat{\boldsymbol{x}}_n\}\subseteq X$
$\boldsymbol{K}_{SX\,i,j}=k(\hat{\boldsymbol{x}}_i,\hat{\boldsymbol{x}}_j)$, $\boldsymbol{K}_{SS\,i,j}=k(\hat{\boldsymbol{x}}_i,\hat{\boldsymbol{x}}_j)$
approximate $\boldsymbol{K}=\boldsymbol{K}_{XS}\boldsymbol{K}_{SS}^{-1}\boldsymbol{K}_{SX},\boldsymbol{K}_{SS}=\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T$.
new point $\boldsymbol{x}'$: $\boldsymbol{z}(\boldsymbol{x}')=\boldsymbol{D}^{-1/2}\boldsymbol{V}^T[k(\boldsymbol{x}',\hat{\boldsymbol{x}}_1),...,k(\boldsymbol{x}',\hat{\boldsymbol{x}}_m)]$

## 3 Pool-based active Learning (semi-supervised)

**Uncertainty sampl.** $U_t(x)=U(x|x_{1:t-1},y_{1:t-1})$, request $y_t$ for $x_t=\text{argmax}_x U_t(x)$. **SVM:** $x_t=\text{argmin}_{x_i}|\boldsymbol{w}^T\boldsymbol{x}_i|$, i.e. $U_t(\boldsymbol{x})=\frac{1}{|\boldsymbol{w}_t^T\boldsymbol{x}|}$.

**Sub-linear time w/ LSH** $|\boldsymbol{w}^T\boldsymbol{x}_i|$ small if $\angle_{\boldsymbol{w},\boldsymbol{x}_i}$ close to $\pi$.
Hash hyperplane: $h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{a},\boldsymbol{b})=[h_{\boldsymbol{u}}(\boldsymbol{a}),h_{\boldsymbol{v}}(\boldsymbol{b})]=[\text{sgn}(\boldsymbol{u}^T\boldsymbol{a}),\text{sgn}(\boldsymbol{v}^T\boldsymbol{b})]$.
LSH hash family: $h_H(z)=h_{u,v}(\boldsymbol{z},\boldsymbol{z})$ if $z$ datapoint, $h_H(z)=h_{u,v}(\boldsymbol{z},-\boldsymbol{z})$ if $z$ query hyperplane. $\text{Pr}[h_H(\boldsymbol{w})=h_H(\boldsymbol{x})]=\text{Pr}[h_{\boldsymbol{u}}(\boldsymbol{w})=h_{\boldsymbol{u}}(\boldsymbol{x})]\text{Pr}[h_{\boldsymbol{v}}(-\boldsymbol{w})=h_{\boldsymbol{v}}(\boldsymbol{x})]=\frac{1}{4}-\frac{1}{\pi^2}(\angle_{\boldsymbol{x},\boldsymbol{w}}-\frac{pi}{2})^2$.
Hash all unlabeled. Loop: Hash $\boldsymbol{w}$, req. labels for hash-coll., update.

**Informativeness** Metric of "information" gainable; $\neq$ uncertainty.

**Version Space** $\mathcal{V}(D)=\{\boldsymbol{w}\mid\forall(\boldsymbol{x},y)\in D\ \text{sgn}(\boldsymbol{w}^T\boldsymbol{x})=y)\}$

**Relevant version space** given unlabeled pool $U=\{x_1',\ldots,x_n'\}$. $\tilde{\mathcal{V}}(D;U)=\{h: U\to\{\pm 1\}\mid\exists\boldsymbol{w}\in\mathcal{V}(D)\ \forall x\in U\ \text{sgn}(\boldsymbol{w}^T\boldsymbol{x})=h(\boldsymbol{x})\}$.

**Generalized binary search** Init $D\leftarrow\{\}$. While $|\tilde{\mathcal{V}}(D;U)|>1$, comp. $v^{\pm}(x)=|\tilde{\mathcal{V}}(D\cup\{(x,\pm)\};U)|$, label of $\text{argmin}_x\max\{v^-(x),v^+(x)\}$.

**Approx.** $|\mathcal{V}|$ Margins of SVM $m^{\pm}(x)$ for labels $\{+,-\},\forall x$. *Max-min* $\max_x \min\{m^+(x), m^-(x))\}$ or *ratio* $\max_x \min\{\frac{m^+(x)}{m^-(x)}, \frac{m^-(x)}{m^+(x)}\}$.

## 4 Model-based clustering – Unsupervised learning

**k-means problem** $\min_\mu L(\mu)$ with $L(\mu) = \sum_{i=1}^N \min_j ||\boldsymbol{x}_i - \mu_j||_2^2$ and *cluster centers* $\mu = \mu_1, ..., \mu_k$. Non-convex! NP-hard in general!

**LLoyd's** Init $\mu^{(0)}$ (somehow). *Assign* all $\boldsymbol{x}_i$ to closest center $z_i \leftarrow \operatorname{argmin}_{j\in[k]} ||\boldsymbol{x}_i - \mu_j^{(t-1)}||_2^2$, *Update* to mean: $\mu_j^{(t)} \leftarrow \frac{1}{n_j}\sum_{i:z_i=j}\boldsymbol{x}_i$. Always converge to *local minimum*.

**Online k-means** Init $\mu$ somehow. For $t \in [n]$ find $z = \operatorname{argmin}_j ||\mu_j - \boldsymbol{x}_t||_2$, set $\mu_c \leftarrow \mu_c + \eta_t(\boldsymbol{x}_t - \mu_c)$. For local optimum: $\sum_t \eta_t = \infty \wedge \sum_t \eta_t^2 < \infty$ suffices, e.g. $\eta_t = \frac{c}{t}$, $c \in \mathbb{R}$.

**Weighted rep.** $C$ $L_k(\mu; C) = \sum_{(w,\boldsymbol{x})\in C} w \cdot \min_j ||\mu_j - \boldsymbol{x}||_2^2$.

$(k,\epsilon)$**-coreset** iff $\forall \mu : (1-\epsilon)L_k(\mu; D) \leq L_k(\mu; C) \leq (1+\epsilon)L_k(\mu; D)$.

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

**Merge coresets** union of $(k,\epsilon)$-coreset is also $(k,\epsilon)$-coreset.

**Compress** a $(k,\delta)$-coreset of a $(k,\epsilon)$-coreset is a $(k,\epsilon+\delta+\epsilon\delta)$-coreset.

**Coresets on streams** Bin. tree of merge-compress. Error $\propto$ height.

**Mapreduce k-means** Construct $(k,\epsilon)$-coreset C, solve k-means (w/ many restarts) on coreset. (Repeat.) Near-optimal solution.

## 5 $k$-armed bandits as recommender systems

**$k$-armed bandit** $k$ arms with diff. prob. dist. For $t \in [T]$ rounds, pick $i_t \in [k]$, sample $y_t \in P_i$ (indep. of other rounds). Max. $\sum_{t=1}^T y_t$.

**Regret** $\mu_i$ mean of $P_i$ (arm $i$), $\mu^* = \max_i \mu_i$. Regret $r_t = \mu^* - \mu_{i_t}$. Total regret $R_T = \sum_{t=1}^T r_t$.

**$\epsilon$-greedy** *Explore* u.a.r. with prob. $\epsilon_t$, *exploit* with prob. $1-\epsilon_t$: choose $\operatorname{argmax}_i \hat{\mu}_i$. Suitable $\epsilon_t \in O(1/t)$ gives $R_T \in O(k\log T)$. Clearly unoptimal: !TODO! Why?

**UCB1** Init $\hat{\mu}_i \leftarrow 0$; try all arms once. Following $t \in [T-k]$ rounds: $UCB(i) \leftarrow \hat{\mu}_i + \sqrt{\frac{2\text{logt}}{n_i}}$, pick $i_t \leftarrow \operatorname{argmax}_i UCB(i)$, observe $y_t$. Update $n_{i_t} \leftarrow n_{i_t}+1, \hat{\mu}_{i_t} \leftarrow \hat{\mu}_{i_t} + \frac{y_t - \hat{\mu}_{i_t}}{n_{i_t}}$.

**Contextual bandits** Round $t$: Obs. *context* $\boldsymbol{z}_t \in \mathcal{Z} \subseteq \mathbb{R}^d$; *recommend* $\boldsymbol{x}_t \in \mathcal{A}_t$. Reward $y_t = f(\boldsymbol{x}_t, \boldsymbol{z}_t) + \epsilon_t$. Regret $r_t = \max_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{z}_t) - f(\boldsymbol{x}_t, \boldsymbol{z}_t)$. Often $f(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{w}_{\boldsymbol{x}}^T \boldsymbol{z}$ linear.

**Idea behind LinUCB** Estimate $\hat{\boldsymbol{w}}_i = \operatorname{argmin}_{\boldsymbol{w}} \sum_{t=1}^m (y_t - \boldsymbol{w}^T\boldsymbol{z}_t) + ||\boldsymbol{w}||_2^2$. Closed form: $\hat{\boldsymbol{w}}_i = M_i^{-1}D_i^T\boldsymbol{y}_i$, $M_i = D_i^T D_i + I$, $D_i = [\boldsymbol{z}_1|...|\boldsymbol{z}_m], \boldsymbol{y}_i = (y_1|...|y_m)^T$.

*Confidence* If $\alpha = 1 + \sqrt{\ln(\frac{2}{\delta})/2}$:

$$\Pr\left[|\hat{\boldsymbol{w}}_i^T\boldsymbol{z}_t - \boldsymbol{w}_i^T\boldsymbol{z}_t| \leq \alpha\sqrt{\boldsymbol{z}_t^T M_i^{-1}\boldsymbol{z}_t}\right] \geq 1-\delta.$$

**LinUCB (Algorithm)** For $t = [T]$ receive *action set* $A_t$ and features $\boldsymbol{z}_t$. For all $\boldsymbol{x} \in A_t$: if $\boldsymbol{x}$ new, set $M_{\boldsymbol{x}} \leftarrow \mathbb{I}$ and $b_x \leftarrow 0$; set $\hat{w}_{\boldsymbol{x}} \leftarrow M_{\boldsymbol{x}}^{-1}b_{\boldsymbol{x}}$; set $UCB_{\boldsymbol{x}} \leftarrow \hat{w}_{\boldsymbol{x}}^T\boldsymbol{z}_t + \alpha\sqrt{\boldsymbol{z}_t^T M_{\boldsymbol{x}}^{-1}\boldsymbol{z}_t}$. Recommend action $\boldsymbol{x}_t = \operatorname{argmax}_{\boldsymbol{x}\in A_t} UCB_{\boldsymbol{x}}$; observe $y_t$. Set $M_{\boldsymbol{x}} \leftarrow M_{\boldsymbol{x}} + \boldsymbol{z}_t\boldsymbol{z}_t^T$ and $b_{\boldsymbol{x}} \leftarrow b_{\boldsymbol{x}} + y_t\boldsymbol{z}_t$.

**Hybrid Model** $y_t = \boldsymbol{w}_i^T\boldsymbol{z}_t + \beta^T\phi(\boldsymbol{x}_i, \boldsymbol{z}_t) + \epsilon_t$ captures sep. and shared effects.

**Rejection Sampling** Evaluate bandit: For $t \in \mathbb{N}$ read log $(\boldsymbol{x}_1^{(t)}, \dots, \boldsymbol{x}_k^{(t)}, \boldsymbol{z}_t, a_t, y_t)$. Pick $a'_t$ by algo. If $a'_t = a_t$ feed $y_t$ to algo., else ignore line. Stop after $T$ feedbacks.

## 6 Submodularity

$F : 2^V \to \mathbb{R}$ *subm.* iff $F(A\cup\{s\}) - F(A) \geq F(B\cup\{s\}) - F(B)$, $\forall A \subseteq B \subseteq V, s \in V$ with $s \notin B$.

**Closedness:** If $F_{1,\dots,m}(A)$ subm. then $\lambda_i > 0$: $F'(A) := \sum_i \lambda_i F_i(A)$ subm.

**Other properties:** If $F(S)$ subm. on $V$, then $F(S\cap W), F(S\cup W), F(V\setminus S)$ subm. where $W \subseteq V$.

**Marginal gain:** $\Delta_F(s|A) = F(\{s\}\cup A) - F(A)$

**Greedy algo:** In round $i+1$, previously picked $A_i = \{s_1, ..., s_i\}$; pick $s_{i+1} = \operatorname{argmax}_s \Delta_F(s|A_i) = \operatorname{argmax}_s(F(\{s\}\cup A_i) - F(A_i))$.

**Lazy Greedy:** *Observation:* Submodularity implies $\Delta(s|A_i) \geq \Delta(s|A_{i+1})$. Algo.: $A_0 \leftarrow \{\}$; first iteration as usual. Then keep ordered list of $\Delta_i$ from prev. iteration. For $i \in [k]$ do: $\Delta_i = F(A_{i-1}\cup\{s^*\})$, $s^* = \operatorname{argmax}_s \Delta_F(s|A_{i-1})$ (*top element*). If $s^*$ is still top, then $A_i \leftarrow A_{i-1}\cup\{s^*\}$ else resort and pick top element (as in Greedy).