Machine Learning Review

Henrique Gubert

Columbia University

2015

Maximum Likelihood Estimation

- Likelihood $\prod_{i=1}^{n} p(\boldsymbol{x}_i; \boldsymbol{\theta})$

- MLE $\boldsymbol{\theta}_{\text{ML}} := \arg\max_{\theta \in \mathcal{T}} \prod_{i=1}^{n} p(\boldsymbol{x}_i; \boldsymbol{\theta})$

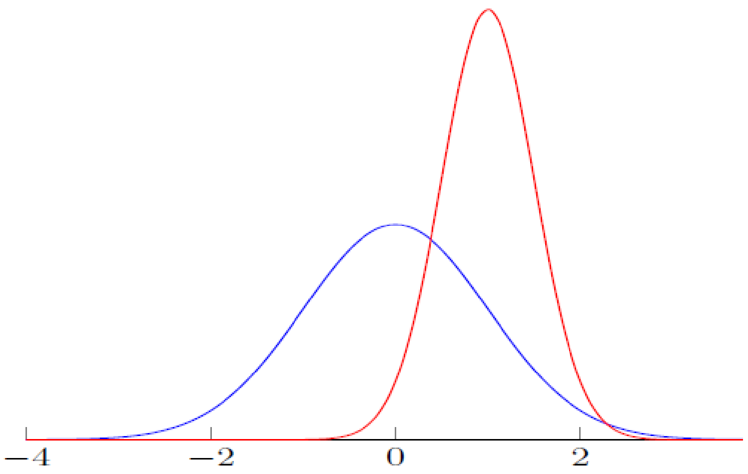
- Use Log-likelihood instead since $\arg\max_{y} \log(g(y)) = \arg\max_{y} g(y)$

- For strictly convex/concave problems: take derivative and set to 0

- You should know how to compute MLE for many distributions

Classifiers via Generative Models

- Model each class with a generative probability model
- Use Bayes Rule $$\Pr[Y=y \mid X=x] = \frac{\Pr[Y=y]\cdot\Pr[X=x \mid Y=y]}{\Pr[X=x]}.$$
- For classification: $$f(x) = \arg\max_{y\in\mathcal{Y}} \Pr[Y=y]\cdot\Pr[X=x \mid Y=y].$$
- Types of error: empirical x test x true

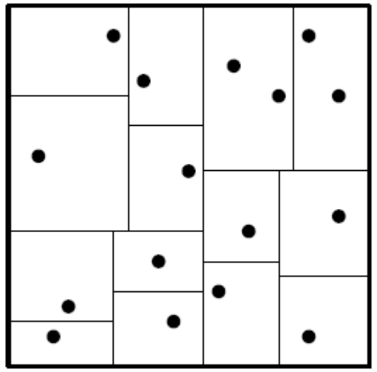


K Nearest Neighbors

- Distance metric matters. Euclidean is default $\|\boldsymbol{u} - \boldsymbol{v}\|_2 := \sqrt{\sum_{i=1}^{d} (u_i - v_i)^2}.$

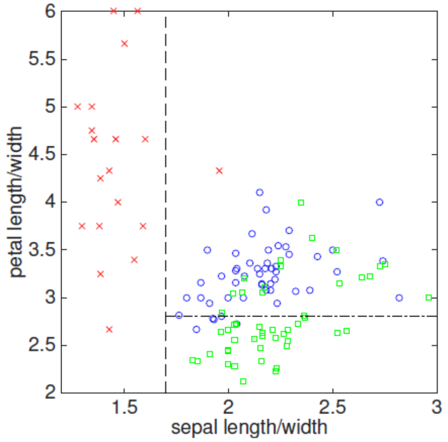
- Choose k minimizing hold-out or leave-one-out. If you use test you estimate will be biased

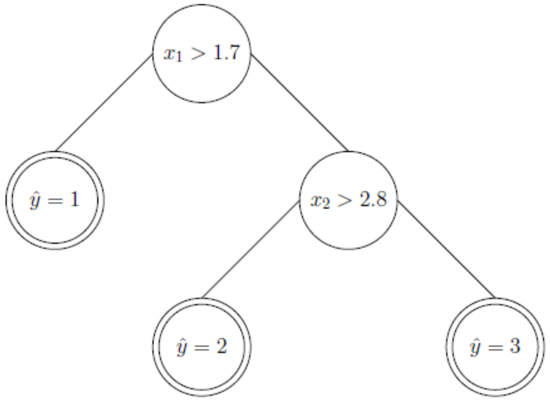
- Efficient but approximate ways of computing K-NN:
  - Locally Sensitive Hash
  - K-D Trees

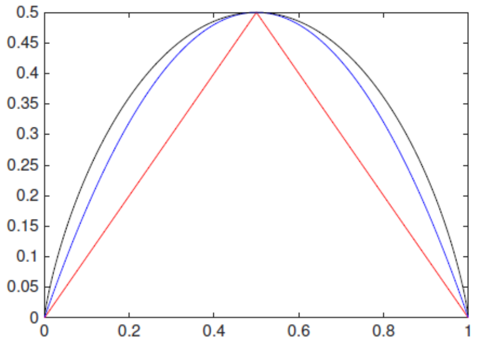


Decision Trees

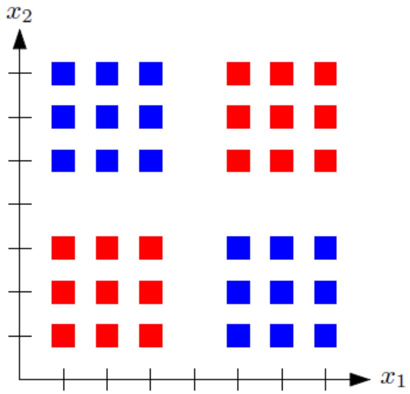
- 3 ways to choose next split:
  - Classification Error $u(S) := 1 - \max_{k \in \mathcal{Y}} p_k$
  - Gini Index (concave) $u(S) := 1 - \sum_{k \in \mathcal{Y}} p_k^2$
  - Entropy (concave) $u(S) := \sum_{k \in \mathcal{Y}} p_k \log \frac{1}{p_k}$



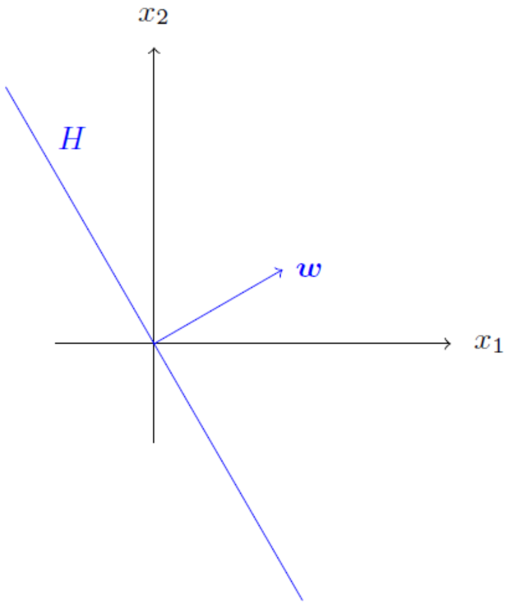


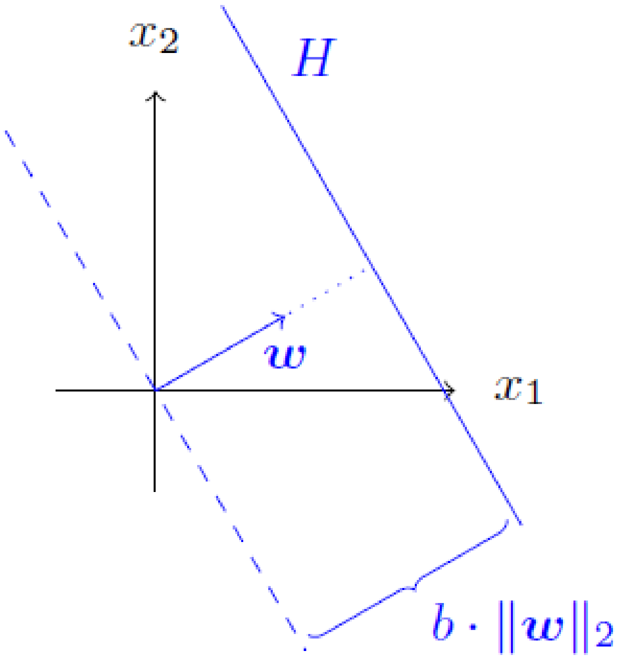


- Avoid overfitting by pruning (and not stoping early)



Linear Classifiers





- Geometric Interpretation

- Homogeneous vs Non-Homogeneous

- Empirical Risk Minimization

$$\arg\min_{\boldsymbol{w},t} \mathrm{err}(f_{\boldsymbol{w},t}, S) = \arg\min_{\boldsymbol{w},t} \frac{1}{|S|}\sum_{(\boldsymbol{x},y)\in S} \mathbb{1}\{\mathrm{sign}(\langle \boldsymbol{w},\boldsymbol{x}\rangle - t) \neq y\}$$

- Linearly Separable datasets: possible to get 0 empirical risk

Perceptron

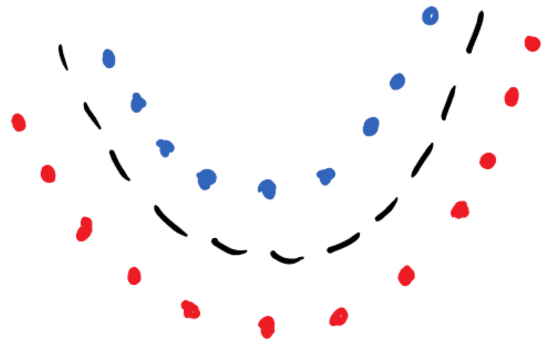
- If data is separable, converges to zero empirical risk in finite iterations

Input: training data S

- **Let** $w_1 = \vec{0}$.
- **For** $t = 1,2, \ldots$:
  - **If** there is $(x_t, y_t) \in S$ such that $f_{w_t}(x_t) \neq y_t$, **then**:
    - **Update**: $w_{t+1} := w_t + y_t x_t$
  - **Else**: **return** w_t

- Variations:
  - Online Perceptron: not guaranteed it will find linear separator
  - Voted Perceptron: use of survival time
  - Averaged Perceptrion: average w weighted by survival time

Feature Expansion and Kernels



Original feature vector: $x = (1, x_1, x_2)$

New feature vector: $\phi(x) = \left(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$

- Kernel Trick:
  - Eficient inner product in expanded space $\langle \phi(x), \phi(x') \rangle = K(x, x')$
  - Can build kernels by summing or multiplying other kernels
  - Allow us to expand features into the infinite-dimensional space (Gaussian kernel)

$$\langle \phi(x), \phi(x') \rangle = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

SVM

- Goal: maximize margin. Why? Better true error bound
- Hard-margin dual problem:

$$\max_{\alpha_1,\dots,\alpha_n} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y^{(i)} y^{(j)} K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \quad \alpha_i \geq 0.$$

$$\langle \hat{\boldsymbol{w}}, \cdot \rangle := \sum_{i=1}^{n} \hat{\alpha}_i y^{(i)} K(\boldsymbol{x}^{(i)}, \cdot);$$

$$\hat{\theta} := \frac{\min_{\boldsymbol{x}\in S_\oplus} K(\hat{\boldsymbol{w}}, \boldsymbol{x}) + \max_{\boldsymbol{x}\in S_\ominus} K(\hat{\boldsymbol{w}}, \boldsymbol{x})}{2}$$

- Soft-margin primal problem (add slack):

$$\min_{\boldsymbol{w}\in\mathbb{R}^d, \theta\in\mathbb{R}, \boldsymbol{\xi}\in\mathbb{R}^n} \quad \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad y^{(i)}\left(\langle \boldsymbol{w}, \boldsymbol{x}^{(i)} \rangle - \theta\right) \geq 1 - \xi_i \qquad \text{for } i = 1, 2, \dots, n$$

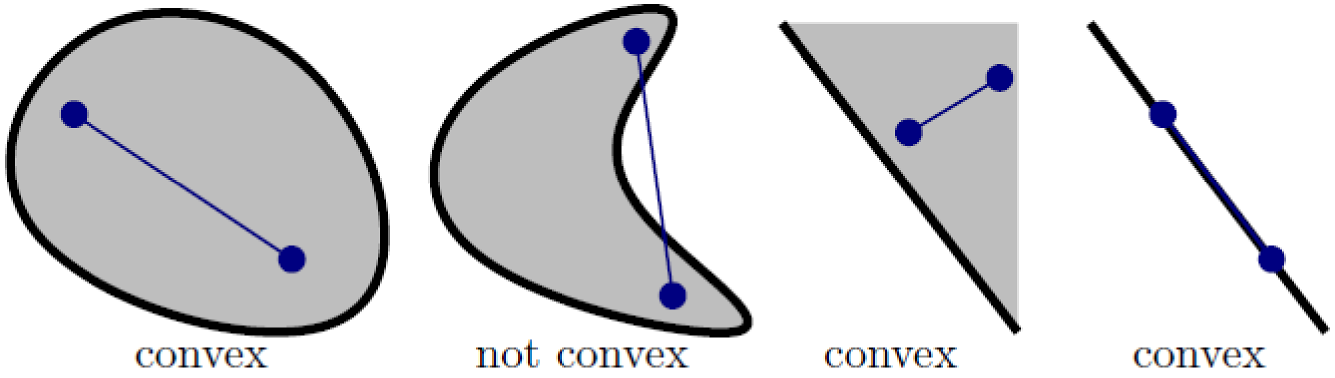
$$\xi_i \geq 0 \qquad \text{for } i = 1, 2, \dots, n$$

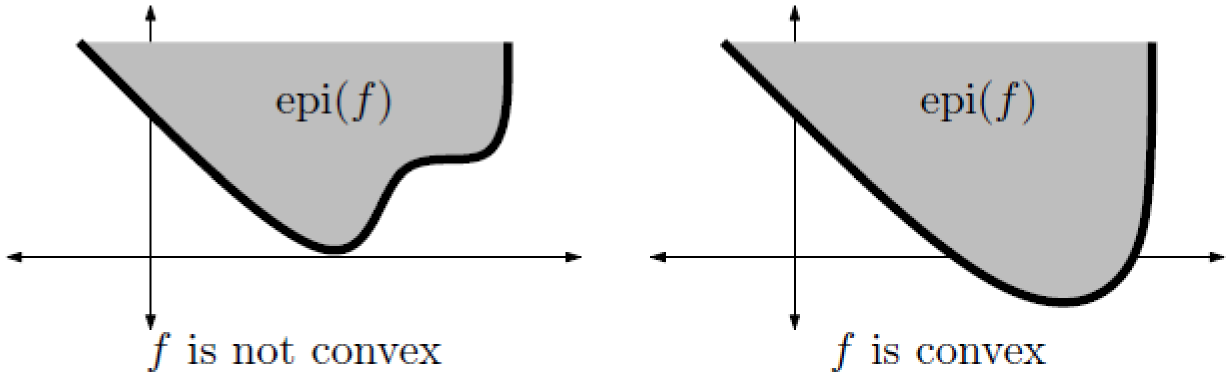
Convexity

- Convexity
- Convex Function
  - Epigraph is convex

- Jensen's Inequality
  - For convex function f:

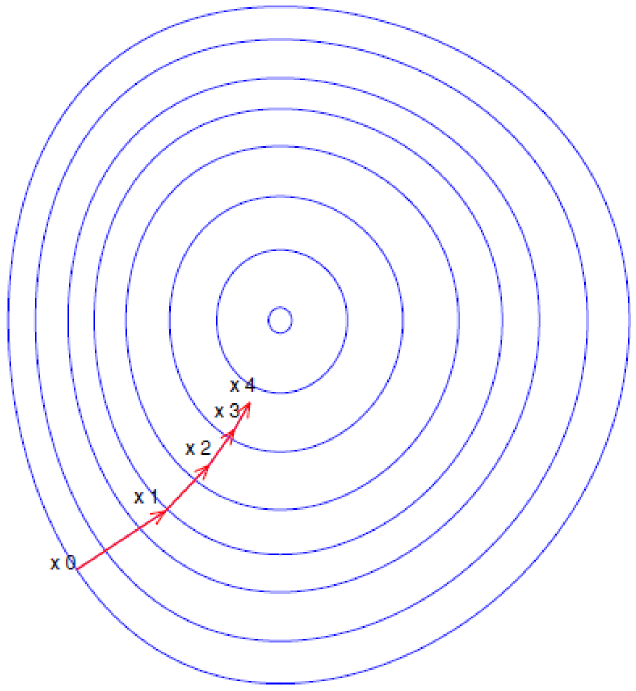
$$f\left(\sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i\right) \leq \sum_{i=1}^{n} \alpha_i \cdot f(\boldsymbol{x}_i)$$





- For convex minimization problems: global optimum = local optimum

Convex Optimization



- Gradient Descent

$$\boldsymbol{\lambda}^{(t)} := \nabla f(\boldsymbol{x}^{(t)})$$
$$\boldsymbol{x}^{(t+1)} := \boldsymbol{x}^{(t)} - \eta_t \boldsymbol{\lambda}^{(t)}$$

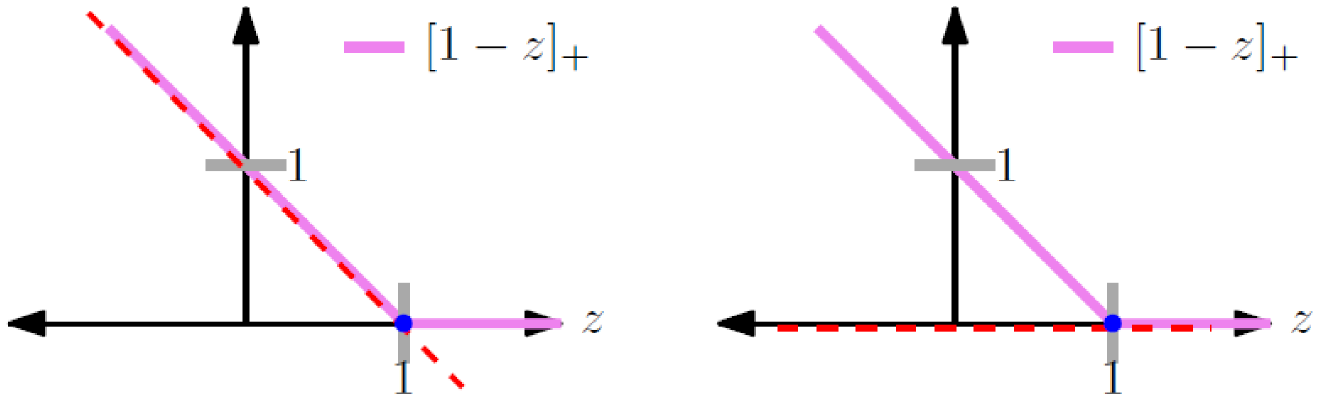
- Step Size
  - Constant $\eta_t := c$
  - Decreasing $\eta_t := c/\sqrt{t}$
  - Line Search
    - Backtracking Line Search
      - Start with $\eta := 1$.
      - While $f(\boldsymbol{x} - \eta\boldsymbol{\lambda}) > f(\boldsymbol{x}) - \frac{1}{2}\eta\|\boldsymbol{\lambda}\|_2^2$: Set $\eta := \frac{1}{2}\eta$.

- Stopping Condition
  - General: stop when gradient is close to zero
  - Machine Learning: stop when hold-out error is minimum

Non-Differentiability in Convex Optimization

- Example: Hinge Loss

- Subgradients

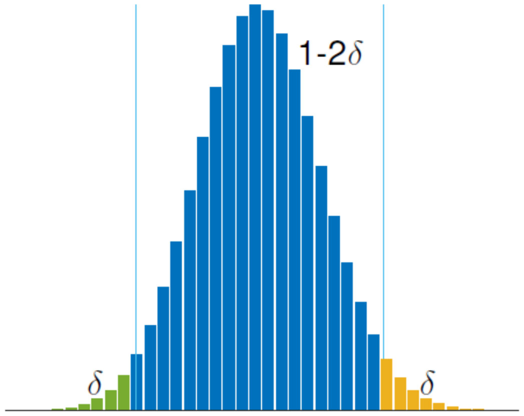


- Subgradient Descent: use any subgradient

- Stopping Condition:
  - General: complicated since must compute all subgradients
  - Machine Learning: use hold-out error

Tail Bounds

- Large deviation of a binomial distribution are exponentially unlikely
  - Concept: Relative Entropy $\mathrm{RE}(a\|b) \;:=\; a\ln\frac{a}{b} + (1-a)\ln\frac{1-a}{1-b}$
  - Tail bounds
  - Union bound



Union bound: $\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$

Upper tail bound: For any $u > p$,

$$\Pr[S \geq n \cdot u] \;\leq\; \exp(-n \cdot \mathrm{RE}(u\|p)).$$

Lower tail bound: For any $\ell < p$,

$$\Pr[S \leq n \cdot \ell] \;\leq\; \exp(-n \cdot \mathrm{RE}(\ell\|p)).$$

Both exponentially small in n.

Learning Theory

- Realizability assumption: there exists classifier with zero true error
- Consistent classifier: converges to zero true error as training goes to infinity
- PAC Theory: shows that a consistent classifier under the realizability assumption has na upper bound on true error (with high probability)
- Upper bound decreases as we...
  - have more training examples
  - have a lower model complexity (avoid overfitting)

Crossvalidation

- Hold-out Validation

Training	Validation	Test

Classifier $\hat{f}_h$ trained on Training data $\longrightarrow \mathrm{err}(\hat{f}_h, \text{Validation})$.

Training + Validation	Test

Classifier $\hat{f}_h$ trained on Training + Validation data $\longrightarrow \mathrm{err}(\hat{f}_h, \text{Test})$.

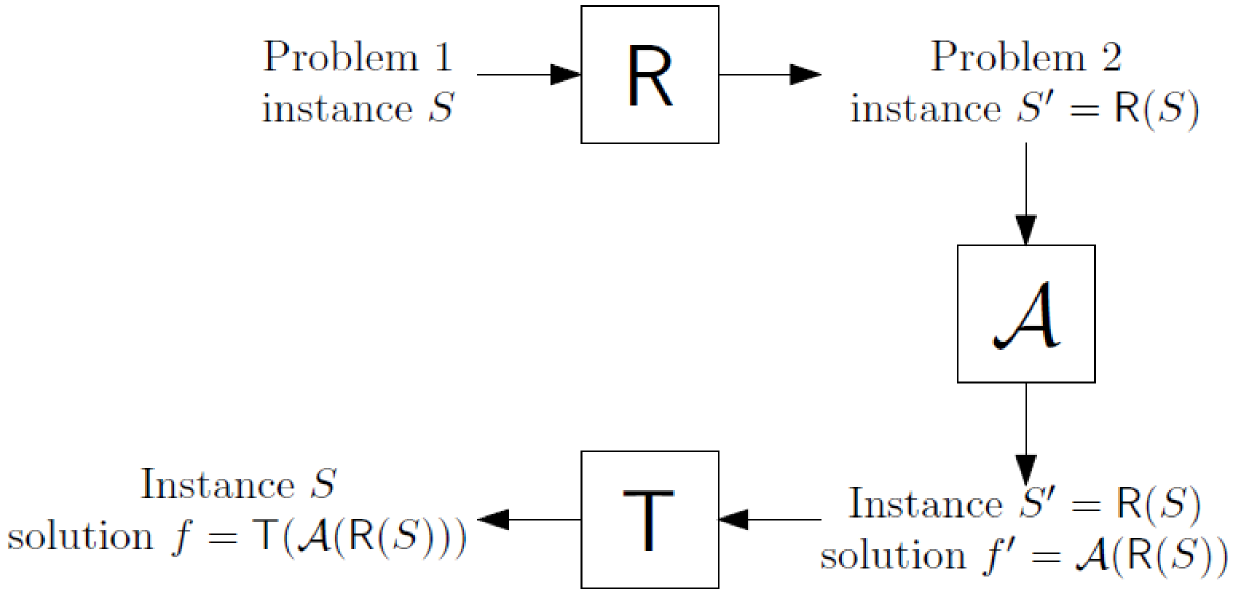
- K-fold crossvalidation

$K = 4$

Validation	Training	Training	Training
Training	Validation	Training	Training
Training	Training	Validation	Training
Training	Training	Training	Validation

Reductions

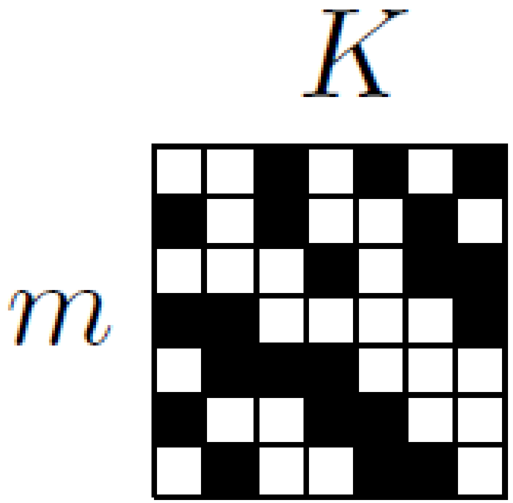
- Use algorithm A, suited to solve problem X, to solve problem Y
- Has 3 steps:
  - Map instance of Y to X
  - Solve with A
  - Map result to the original space



Reductions - Examples

- Importance-weighted Classification
  - Use Rejection Sampling Reduction

- Multi-class Classification
  - One-vs-all
  - Error Correcting Output Codes
    - Can still get a correct output if some classifiers fail



Boosting

- Use many weak learners to form a good classifier

$$\alpha_t = \frac{1}{2} \ln \frac{1+z_t}{1-z_t}$$

$$D_{t+1}(x,y) \propto D_t(x,y) \exp(-\alpha_t \cdot y f_t(x))$$

- Adaboost
  - Each selected weak learner is assigned a weight
  - Weights of the training examples are updated
    - Misclassified examples have weights increased
    - Correctly classified have weights decreased

$$f_{\text{final}}(x) := \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot f_t(x)\right).$$

- Boosting tends to increase margin with more rounds, therefore get better generalization error. Less prone to overfitting.

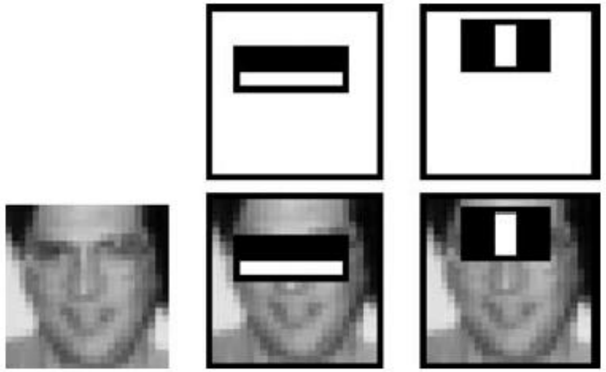
Boosting

- In the end AdaBoost is a linear classifier over a feature map defined by “weak learners”

$$f_{\text{final}}(x) := \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot f_t(x)\right).$$

- The AdaBoost algorithm corresponds to some sort of “coordinate ascent”, where you find the optimal coeficient, one feature at a time

- Example: Viola-Jones Face Detector



Linear Regression

- Ordinary Least Squares
  - Has closed form solution
  - MLE under Gaussian assumption

$$\hat{\boldsymbol{w}}_{\text{ols}} := \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{(\boldsymbol{x},y)\in S} \Big(y - \langle \boldsymbol{x}, \boldsymbol{w}\rangle\Big)^2 .$$

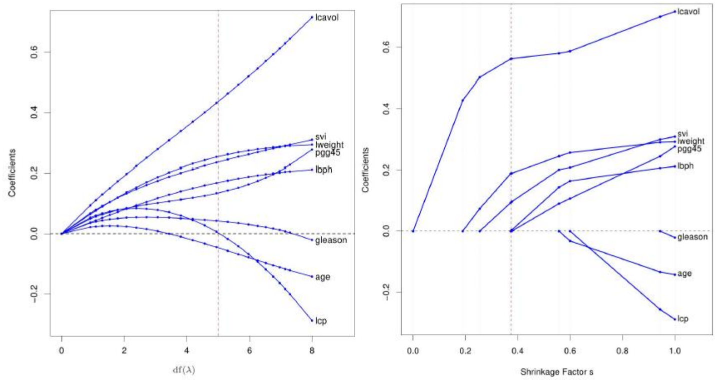
$$\boldsymbol{r} := \boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ols}}$$

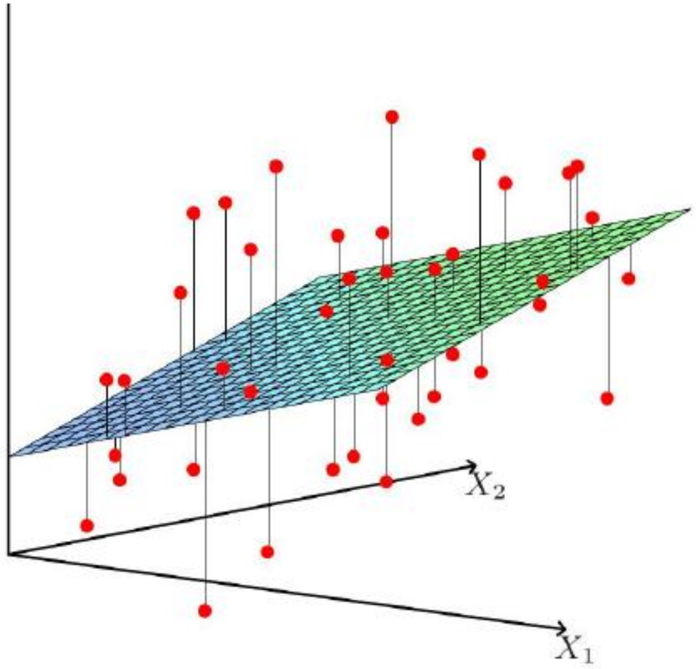
$$\hat{\boldsymbol{w}}_{\text{ols}} := (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$$

- Cost can be decomposed in bias/variance

$$\mathbb{E}(Z-t)^2 = \underbrace{(\mu - t)^2}_{\text{squared bias}} + \underbrace{\mathbb{E}(Z-\mu)^2}_{\text{variance}}$$

- Use regularization
  - Ridge Regression $\|\boldsymbol{w}\|_2^2$
  - Lasso $\|\boldsymbol{w}\|_1$
  - Sparse Regression



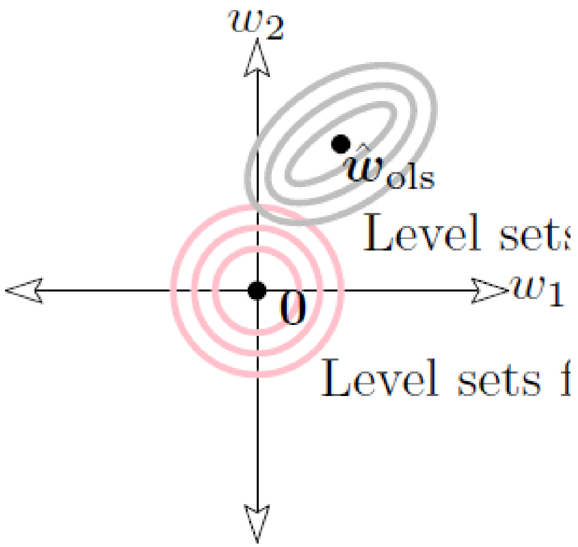


Linear Regression

- Ridge Regression

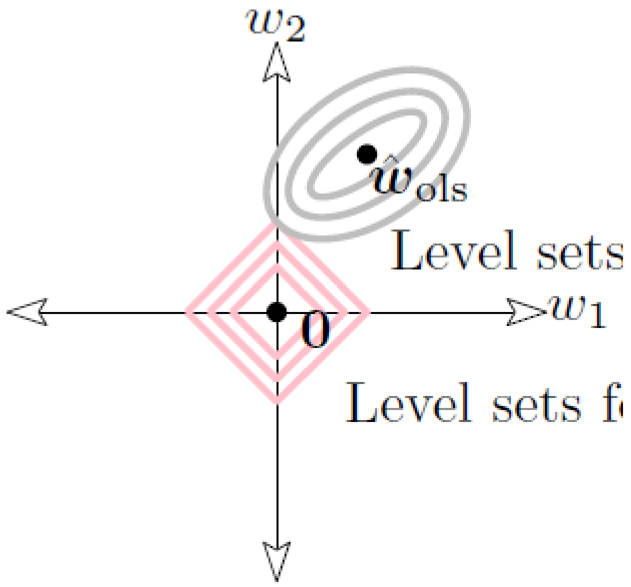
$$\hat{\boldsymbol{w}}_\lambda := \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \frac{1}{n}\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2.$$

$$\hat{\boldsymbol{w}}_\lambda := (\boldsymbol{X}^\top\boldsymbol{X} + n\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y}.$$



- Lasso Regression

$$\hat{\boldsymbol{w}}_{\text{lasso}\,\lambda} := \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \frac{1}{n}\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_1$$

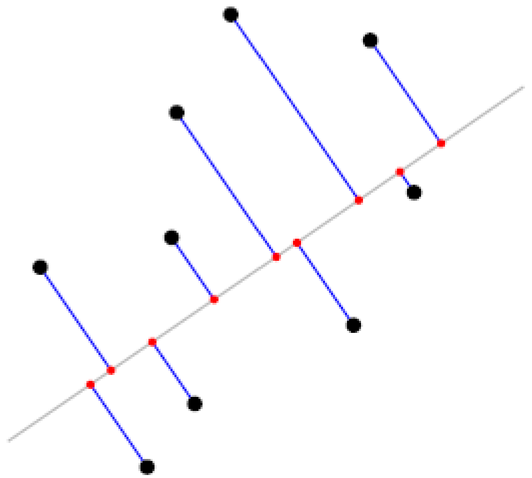


- Sparse Regression
  - Exact solution is intractable
  - Chose one feature at a time until you get to k

PCA and SVD

eigenvalue

$$\boldsymbol{A} = \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^\top = \sum_{i=1}^{d} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$$



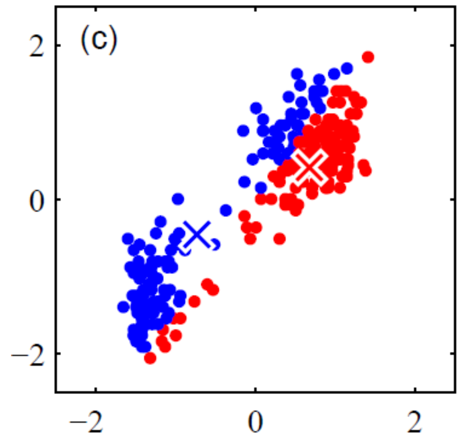
Feature map: $\phi(\boldsymbol{x}) := (\langle \boldsymbol{v}_1, \boldsymbol{x} - \boldsymbol{\mu} \rangle, \langle \boldsymbol{v}_2, \boldsymbol{x} - \boldsymbol{\mu} \rangle, \ldots, \langle \boldsymbol{v}_k, \boldsymbol{x} - \boldsymbol{\mu} \rangle)$

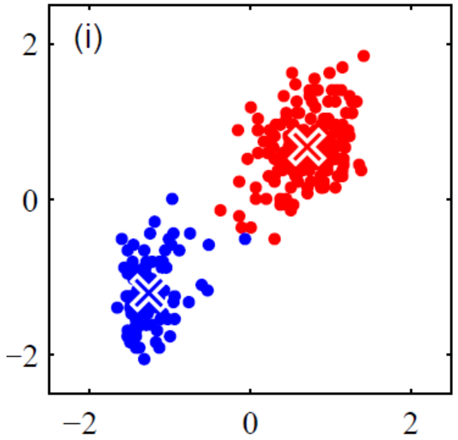
Reconstruction: $\boldsymbol{x} \mapsto \boldsymbol{\mu} + \boldsymbol{V} \phi(\boldsymbol{x})$

account for rows and columns at the same time

$$\boldsymbol{A} = \boldsymbol{U} \boldsymbol{S} \boldsymbol{V}^\top = \sum_{i=1}^{r} s_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$$

Clustering – K-means



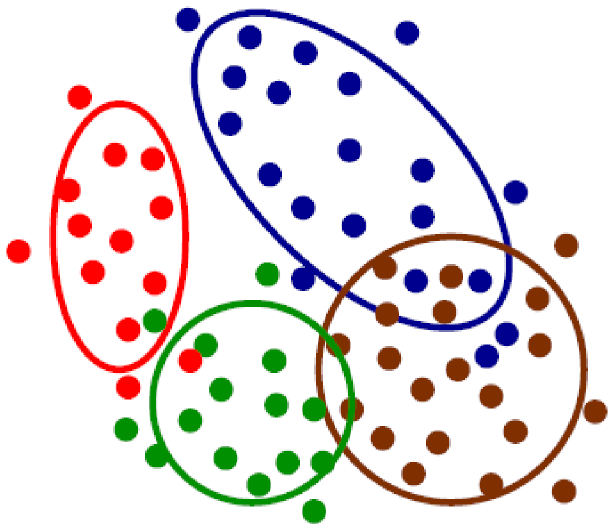


- Lloyd's Algorithm (traditional k-means)
  - Initialization: important in initialization
    - Farthest-first
    - D^2 sampling $\Pr(\boldsymbol{c}_j = \boldsymbol{x}^{(i)}) \propto \min_{j'<j} \|\boldsymbol{x}^{(i)} - \boldsymbol{c}_{j'}\|_2^2.$

- Hierarchical Clustering
  - Divisive (top-down)
  - Aglomerative (bottom-up)

- Mixed-membership model a point belong to multi classes at the same time

Mixture of Gaussians



- We don't know to which Gaussian each point belongs to
- Cannot compute MLE analytically anymore
- Using Jensen's Inequality, derive concave lower bound on likelihood and minimize it instead. Iteratively increase lower bound.
- EM Algorithm

E step: For each $i \in [n]$, $j \in [k]$,

$$w_j^{(i)} \propto \pi_j \cdot p_{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j}(\boldsymbol{x}^{(i)})$$

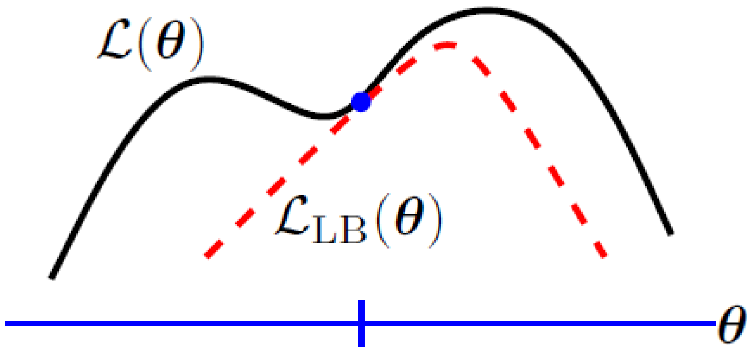
M step: For each $j \in [k]$,

$$\pi_j := \frac{1}{n} \sum_{i=1}^{n} w_j^{(i)}$$

the mean!!!
to check!!!

$$\boldsymbol{\mu}_j := \frac{1}{n\pi_j} \sum_{i=1}^{n} w_j^{(i)} \boldsymbol{x}^{(i)}$$

$$\boldsymbol{\Sigma}_j := \frac{1}{n\pi_j} \sum_{i=1}^{n} w_j^{(i)} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_j)(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_j)^\top.$$



Maximum Entropy

- Pick the distribution that agrees with your empirical statistics, but otherwise expresses as much uncertainty (entropy) as possible

$$H(P) \;=\; -\sum_{x\in\mathcal{X}} P(x)\log_2 P(x) = \;\mathbb{E}_{X\sim P}\left[\log_2 \frac{1}{P(X)}\right]$$

- Alternatively minimize relative entropy with some base distribution, subject to constraints

$$\min_{P\in\Delta(\mathcal{X})} \qquad \underbrace{\mathrm{RE}(P\|\pi)}_{\text{de-similiarity}} \qquad \text{s.t.} \qquad \sum_{x\in\mathcal{X}} P(x)\boldsymbol{T}(x) \;=\; \boldsymbol{b}.$$

- Solution in the form: (exponential family)

$$P_{\eta}(x) = \frac{1}{Z(\boldsymbol{\eta})}\cdot\exp\Big\{\langle\boldsymbol{\eta},\boldsymbol{T}(x)\rangle\Big\}\cdot\pi(x)$$

Exponential Family

- Includes: Gaussian, Bernoulli, Poisson and many others
- Can be expressed in a uniform way, with natural parameters

$$P_\eta(x) = \exp\left\{ \langle \eta, T(x) \rangle - G(\eta) \right\} \cdot \pi(x) \quad \forall x \in \mathcal{X}.$$

- There is a connection between η and $E[T(x)]$. We usually parametrize distributions based on expectations

$$\nabla G(\eta) = \mathbb{E}[T(X)]$$

which feature?

$g := \nabla G$ is an invertible map between natural parameters and expectations:

$$\mu = g(\eta) \iff \eta = g^{-1}(\mu).$$

Exponential Family – Parameter Estimation

- This means that knowing the empirical expectation we can compute the natural parameters

$$\mu = g(\eta) \quad \Longleftrightarrow \quad \eta = g^{-1}(\mu).$$

- Problem: except for well known distributions $g^{-1}(\mu)$ is hard to compute

- Can be seen as Maximum Likelihood problem

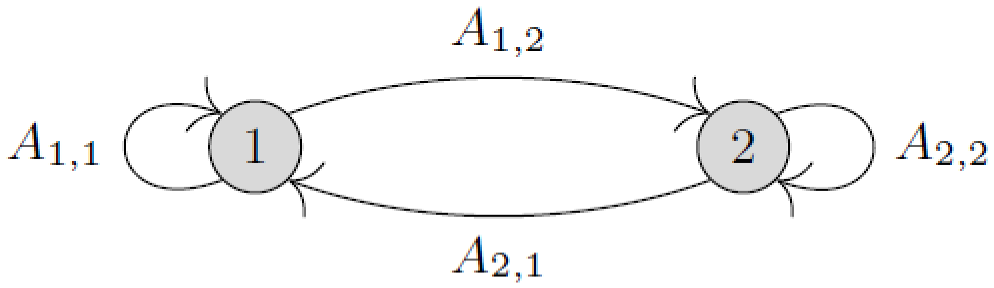
$$\nabla \mathcal{L}(\eta_{\mathrm{ML}}) \;=\; \sum_{i=1}^{n} \Big(T(x_i) - \nabla G(\eta_{\mathrm{ML}}) \Big) \;=\; 0,$$

- One solution: Iterative Projection Algorithm ("Coordinate Ascent")

Markov Models

- Parameters: initial and transition probabilities

$$\boldsymbol{\pi} = \begin{array}{c} \text{state 1} \\ \text{state 2} \end{array} \begin{pmatrix} 0.1 \\ 0.9 \end{pmatrix}, \quad \boldsymbol{A} = \begin{array}{c c} & \begin{array}{cc} \text{state 1} & \text{state 2} \end{array} \\ \begin{array}{c} \text{state 1} \\ \text{state 2} \end{array} & \begin{pmatrix} 0.3 & 0.7 \\ 0.6 & 0.4 \end{pmatrix} \end{array}.$$



- Properties:
  - Irreducible (single SCC)
  - Aperiodic (no oscilation)
  - Both necessary for single stationary state

- Stationary State: find with power method

initialize q arbitrarily.
repeat
  $q^\top := q^\top \boldsymbol{A}$.
until bored.
return q.

PageRank

- Find the stationary state of Web Graph, but with a slight twist:

$$\widetilde{\boldsymbol{A}} := (1-\alpha)\boldsymbol{A} + \frac{\alpha}{d}\begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}$$

add chance to jump into other website

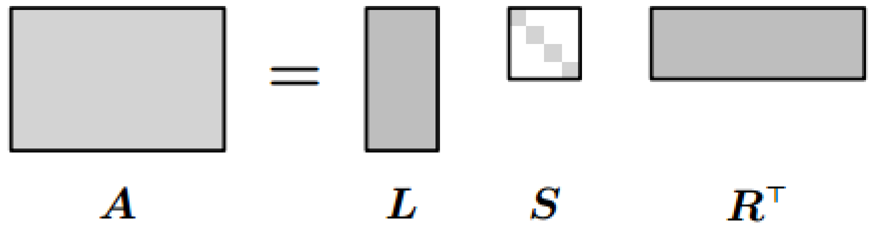
- Now Web Graph is both irredutible and aperiodic

Hidden Markov Model

- Parameters: initial, transition and emission probabilities

- HMM Learning Problems
  - Conditional probability (of hidden states): Forward-Backward Algorithm
  - Decoding: Viterbi
  - Parameter Estimation: EM (Baum-Welch)

Collaborative Filtering



- Idea: decompose ratings matrix using SVD.
  But what if we don’t know full matrix?
  (If we knew there would be nothing to predict)

- Instead, use ALS to fid the solution to the non-convex problem of regularized low-rank matrix completion

$$f(\boldsymbol{U},\boldsymbol{V}) \ := \ \sum_{(i,j)\in\Omega} \left(a_{i,j} - \langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle\right)^2 + \lambda \left(\sum_{i=1}^{m} \|\boldsymbol{u}_i\|_2^2 + \sum_{j=1}^{n} \|\boldsymbol{v}_j\|_2^2 \right)$$

- Prediction is simply

$$\hat{a}_{i,j} := \langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle.$$

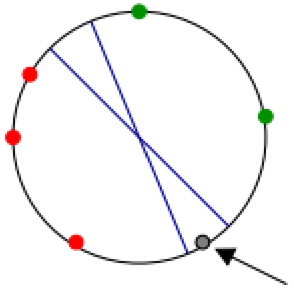
- Other details: removing user and item biases; different loss function

Active Learning

- Ability to draw unlabeled examples from a distribution, and ask them to be labeled (has costs involved)

- Sampling Bias: the fact that you are not using labeled examples that follow the true distribution, may cause your classifier to lose consistency

- Selective Sampling: label examples that are in the region of disagreement



- For non-separable case: Agnostic Notion of Uncertainty

Active Learning

- Importance weighted active learning
  - Stochastic way of doing selection
  - The more uncertain we are about a point, higher the probability that we will query its label
  - As time passes, we decrease the probability of selection

Lagrangian Duality

- Last class, no need to review, right?