CS 512 Data Mining Principles

Classification

Hanghang Tong, Computer Science, Univ. Illinois at Urbana-Champaign, 2021

Chapter 9. Classification: Advanced Methods

- Feature Selection 
- Support Vector Machines
- Weakly Supervised Learning
- Classification with Rich Data Type
- Summary

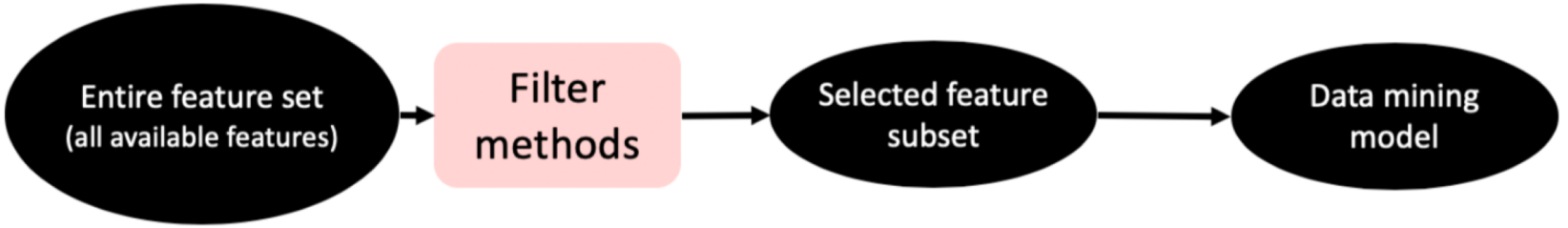
Feature Selection & Feature Engineering

- **Feature Selection**
  - Given a set of *p* initial features, how to select a few most effective ones?
  - Why?
    - Irrelevant features (student ID for predicting GPA)
    - Redundant features (monthly income vs. yearly income)
- **Feature Engineering**
  - Given the initial features, how to construct more effective ones?
    - # of daily positive cases, # of daily tests, # of daily hospitalization → weekly positive rate
  - (traditionally) domain knowledge is the key
  - Deep learning (chapter 11) provides an automatic way

Feature Selection Methods

- **Filter methods**
  - Select features based on some goodness measure
  - Independent of the specific classification model
- **Wrapper methods**
  - Combine the feature selection and classifier model construction steps together,
  - Iteratively
    - use the currently selected feature subset to construct a classification model
    - Use the current classification model to update the selected feature subset.
- **Embedded methods**
  - Simultaneously constructs the classification model and selects the relevant features
  - Embed the feature selection step during the classification model construction step

Filter Methods

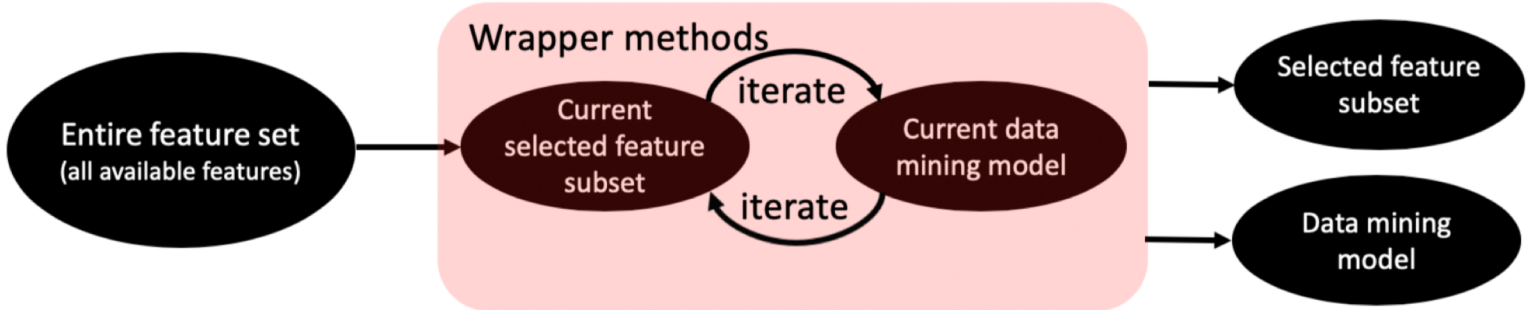


- **General Procedure**
  - Selects features based on some goodness measure
  - Independent of the specific classification model
- **Fisher Scores**
  - **Intuitions**: the feature x (e.g., income) is strongly correlated with the class label y (buy computer) if
    - the average income of all customers who buy a computer is significantly different from the average income of all customers who do not buy a computer,
    - all customers who buy a computer share similar income, and
    - all customers who do not buy a computer share similar income.
  - **Details**

$$s = \frac{\sum_{j=1}^{c} n_j (\mu_j - \mu)^2}{\sum_{j=1}^{c} n_j \sigma_j^2}$$

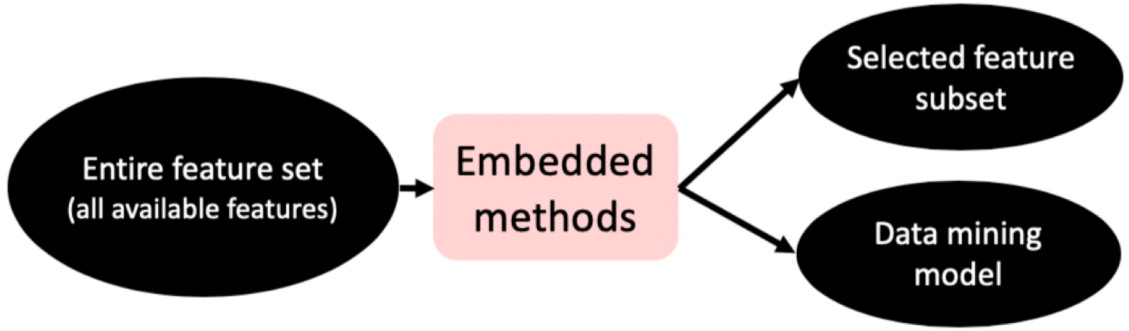
- **Other goodness measures**
  - χ^2 test (for categorical feature); information gain, mutual information.

Wrapper Methods



- **General Procedure**
  - Combines the feature selection and classifier model construction steps together,
  - Iteratively
    - use the currently selected feature subset to construct a classification model
    - Use the current classification model to update the selected feature subset.
- **Key: how to search for the best feature subset**
  - Exhaustive search: 2^p-1 (exponential)
  - Stepwise forward selection:
    - Start with an empty feature subset.
    - At each iteration, select an additional feature to improve performance most
  - Stepwise backward elimination: start with the full set, eliminate one feature at a time
  - Hybrid method

Embedded Methods



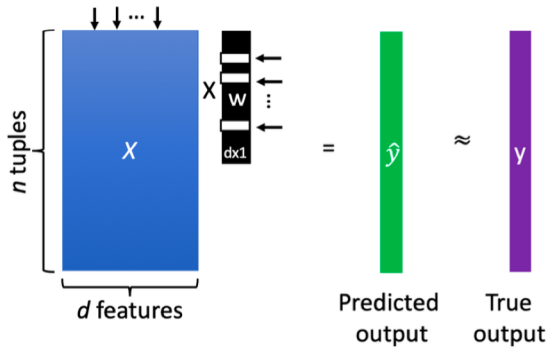
- **General Procedure**
  - Simultaneously constructs the classification model and selects the relevant features
  - Embed the feature selection step during the classification model construction step
- **LASSO**: Least Absolute Shrinkage and Selection Operator
  - Model

$$\hat{L}(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\|w\|_1 = \frac{1}{2}\sum_{i=1}^{n}(y_i - w^T x_i)^2 + \lambda\sum_{j=0}^{d}|w_j|$$

Goodness of prediction

(convex) approximation of # of selected features

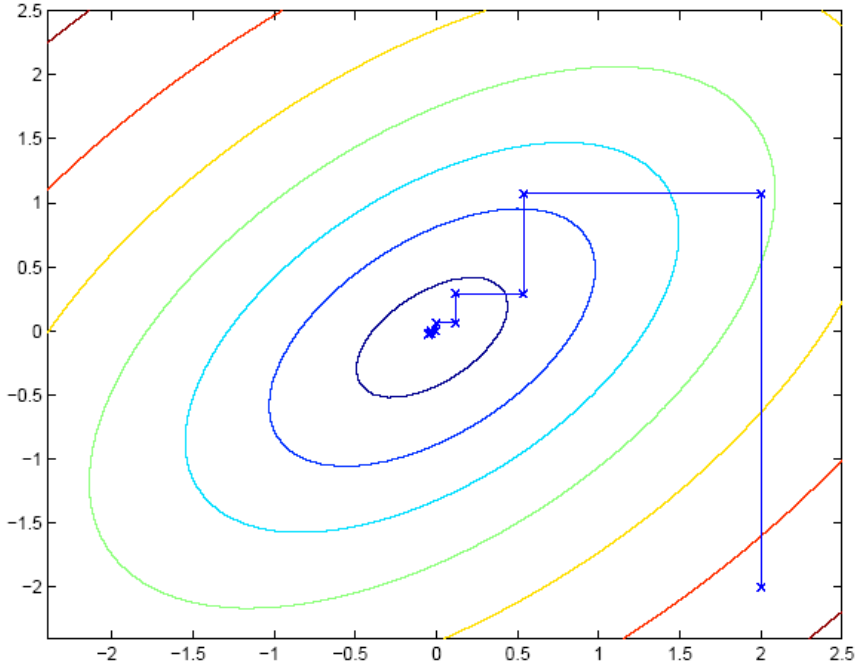
  - Training: Coordinate descent



Coordinate Descent: Minimize *f(x)*

- If *f(x)* is convex and smooth → global minima
- If *f(x)* is convex but not differential → no guarantee on optimality
- If *f(x)* is convex but not differential, but **separable** → global minima
  - g(x): convex and smooth
  - Each $h_i(x)$: convex (but maybe non-smooth)

$$f(x) = g(x) + \sum_{i=1}^{d} h_i(x_i)$$



Coordinate Descent for Lasso

- **LASSO**: Least Absolute Shrinkage and Selection Operator

$$\hat{L}(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\|w\|_1 = \frac{1}{2}\sum_{i=1}^{n}(y_i - w^T x_i)^2 + \lambda\sum_{j=0}^{d}|w_j|$$

  - Training: at each iteration, update one coefficient (w_t) while fixing others
    - Solve a least square regression problem, single dimension input ($x_{i,t}$) and output r

$$\beta_t = \text{argmin}_{w_t}\frac{1}{2}\sum_{i=1}^{n}(r_i - w_t x_{i,t})^2$$

$$r_i = y_i - \sum_{j=0, j\neq t}^{d} w_j x_{i,j}$$

    - Solve the following optimization problem (quadratic, single variable)

$$L(w_t) = \frac{1}{2}(w_t - \beta_t)^2 + \lambda|w_t|$$

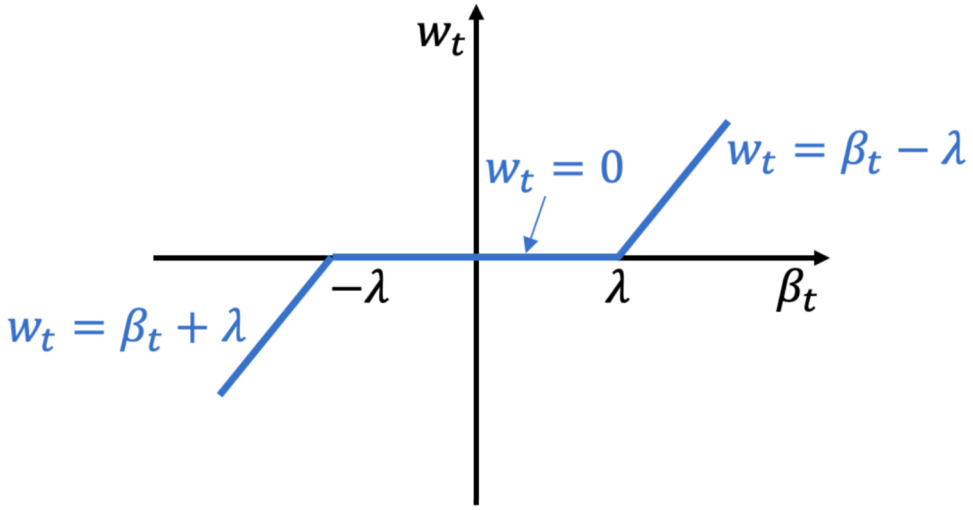
Coordinate Descent for Lasso

- Solve the following optimization problem (quadratic, single variable)

$$L(w_t) = \tfrac{1}{2}(w_t - \beta_t)^2 + \lambda |w_t|$$

- Solution: soft-thresholding

$$w_t = \begin{cases} \beta_t - \lambda & \text{if } \beta_t \geq \lambda \\ \beta_t + \lambda & \text{if } \beta_t \leq -\lambda \\ 0 & \text{otherwise} \end{cases}$$

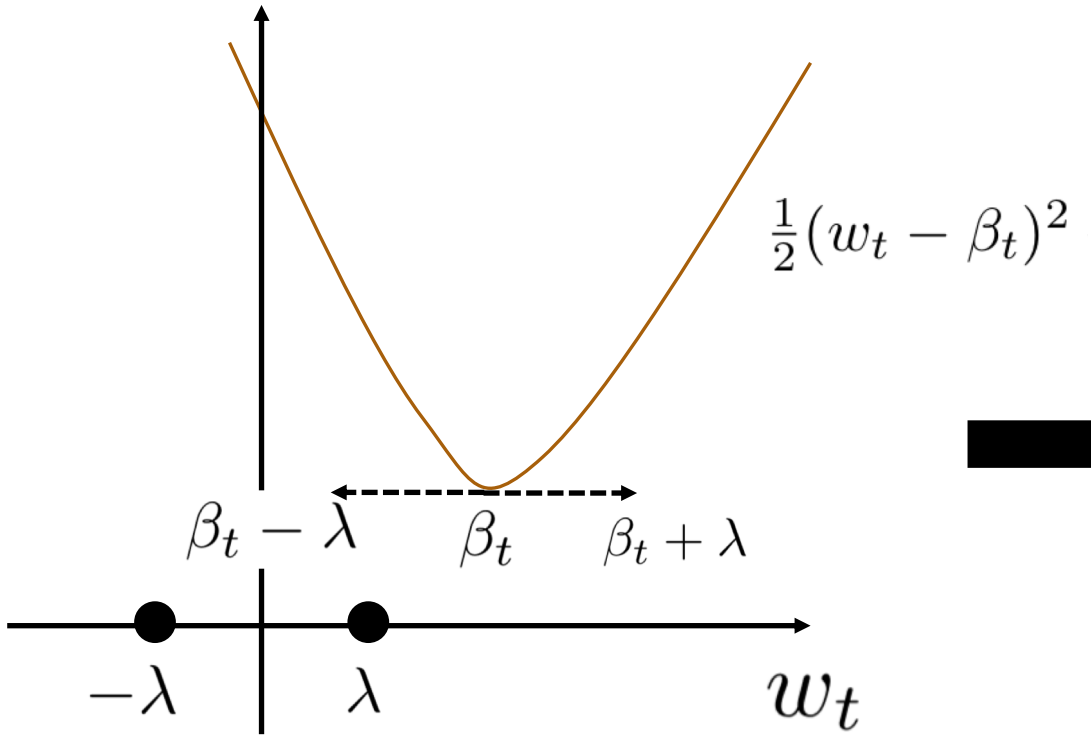


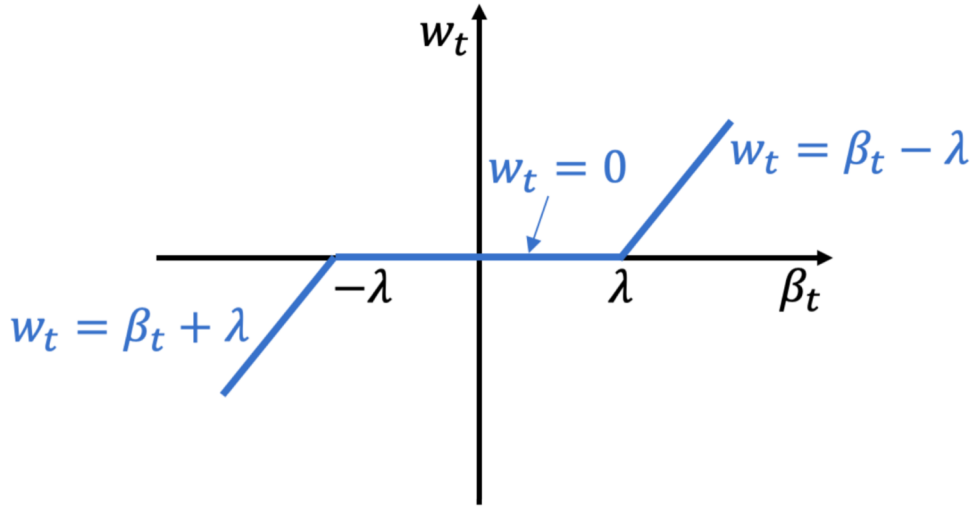
- Intuition: push the magnitude of β_t toward 0

Soft Thresholding

- Solve the following optimization problem (quadratic, single variable)

$$L(w_t) = \tfrac{1}{2}(w_t - \beta_t)^2 + \lambda|w_t|$$





Beyond Lasso: Sparse Learning

- Elastic net: L1+L2 regularization (i.e., ridge + lasso)

$$\frac{1}{2}\sum_{i=1}^{n}(y_i - w^T x_i)^2 + \lambda_1|w|_1 + \lambda_2\|w\|_2^2$$

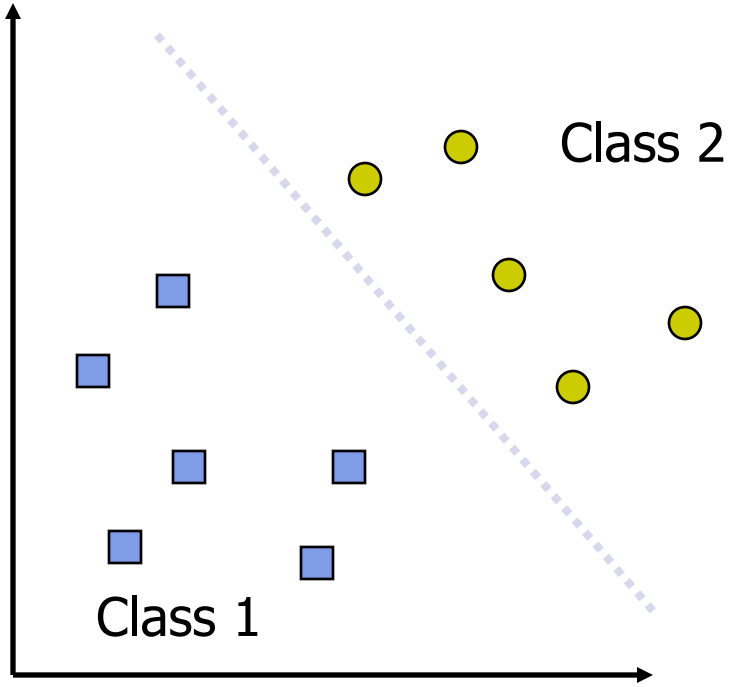
- Group Lasso $\frac{1}{2}\sum_{i=1}^{n}(y_i - w^T x_i)^2 + \sum_{j=1}^{m}\lambda_j\|w_j\|_2$
  - Encourage a group of features to be unselected simultaneously
  - w_j : coefficient vector for a group of features (group *j*)
  - L_2 norm within each group; L_1 norm across groups
- Fused Lasso $\frac{1}{2}\sum_{i=1}^{n}(y_i - w^T x_i)^2 + \lambda_1|w|_1 + \lambda_2\sum_{j=2}^{d}|w_j - w_{j-1}|$
  - Encourage co-select (or co-un-select) adjacent features
- Many more: e.g., graph-guided lasso, graphical lasso, matrix completion
  - …

Chapter 9. Classification: Advanced Methods

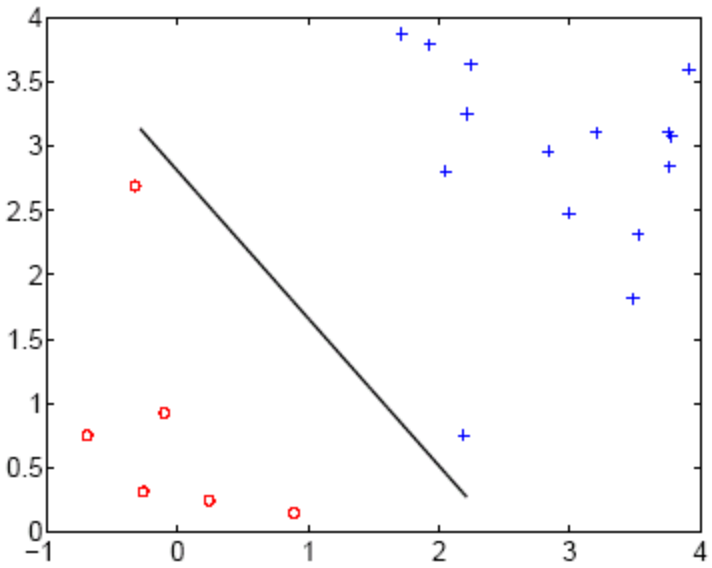
- Feature Selection
- Support Vector Machines 
- Weakly Supervised Learning
- Classification with Rich Data Type
- Summary

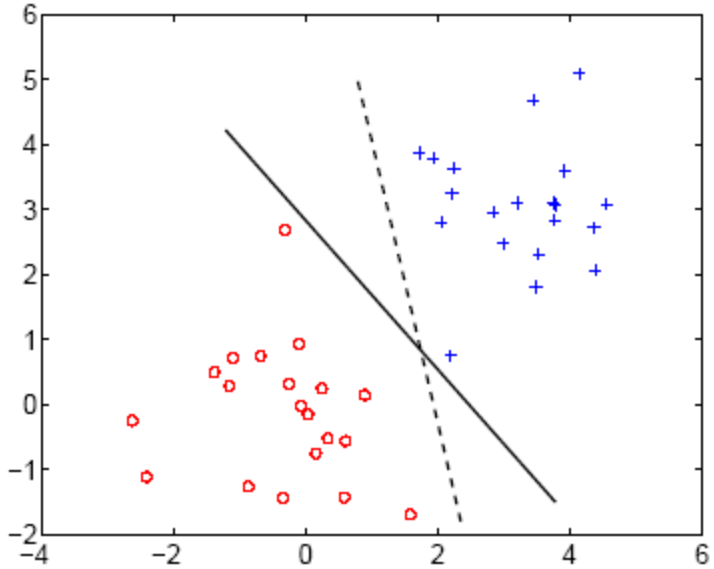
What is a good Decision Boundary?

- Consider a binary classification task with y = ±1 labels (not 0/1 as before).
- When the training examples are linearly separable, we can set the parameters of a linear classifier so that all the training examples are classified correctly
- Many decision boundaries!
  - Generative classifiers
  - Logistic regressions ...
- Are all decision boundaries equally good?



Not All Decision Boundaries Are Equal!



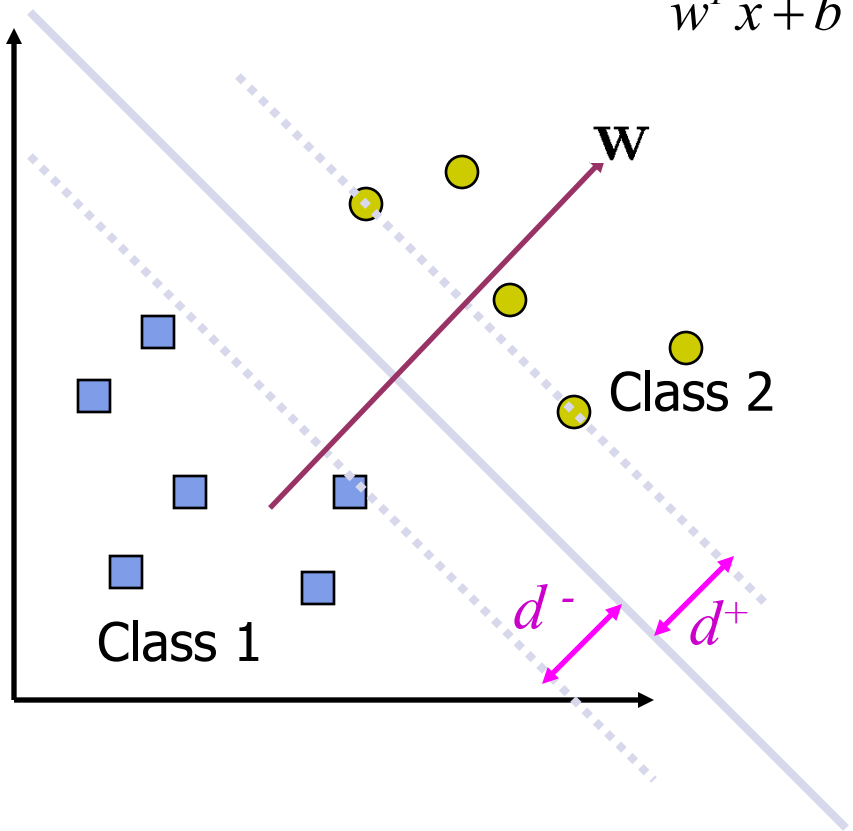


- Why we may have such boundaries (the black solid line)?
  - Irregular distribution
  - Imbalanced training sizes
  - outliers

Classification and Margin

- Parameterzing decision boundary
  - Let w denote a vector orthogonal to the decision boundary, and b denote a scalar "offset" term, then we can write the decision boundary as:

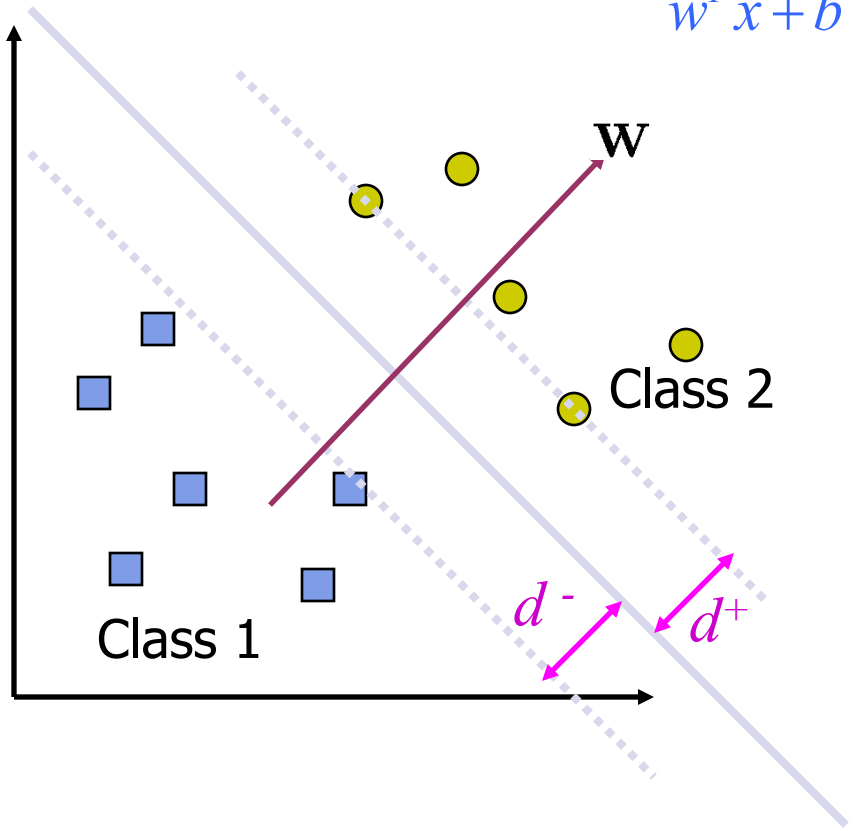
$$w^T x + b = 0$$



Classification and Margin

- Parameterzing decision boundary
  - Let w denote a vector orthogonal to the decision boundary, and b denote a scalar "offset" term, then we can write the decision boundary as:

$$w^T x + b = 0$$



- Margin

$w^T x_i + b > +c$ for all x_i in class 2

$w^T x_i + b < -c$ for all x_i in class 1

Or more compactly:

$$(w^T x_i + b) y_i > c$$

The margin between two points

$\boldsymbol{m} = d^- + d^+ =$

Maximum Margin Classification

- The margin is:

$$m = \frac{w^T}{\|w\|}\left(x_{i^*} - x_{j^*}\right) = \frac{2c}{\|w\|}$$

- Here is our Maximum Margin Classification problem:

$$\begin{aligned} &\max_{w,b} && \frac{c}{\|w\|} \\ &\text{s.t} && y_i(w^T x_i + b) \geq c, \quad \forall i \end{aligned}$$

Maximum Margin Classification, con'd.

- The optimization problem:

$$\max_{w,b} \quad \frac{c}{\|w\|}$$
$$\text{s.t} \quad y_i(w^T x_i + b) \geq c, \quad \forall i$$

- But note that the magnitude of c merely scales w and b, and does not change the classification boundary at all!
- So we instead work on this cleaner problem:

$$\max_{w,b} \quad \frac{1}{\|w\|}$$
$$\text{s.t} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i$$

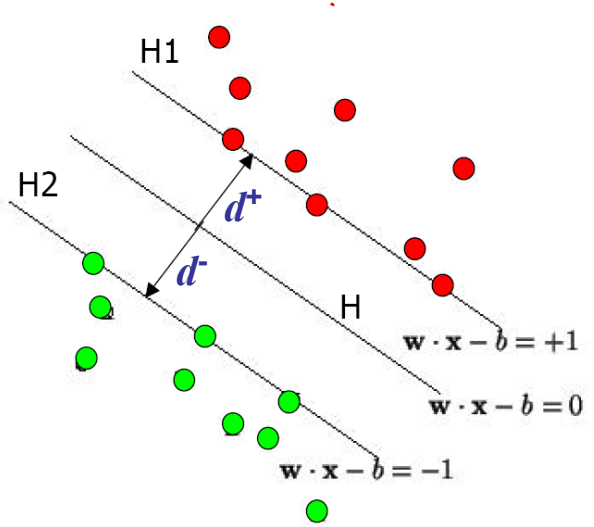
- The solution to this leads to the famous **Support Vector Machines** --- believed by many to be the best "off-the-shelf" supervised learning algorithm

Support vector machine

- A convex quadratic programming problem with linear constrains:

$$\max_{w,b} \quad \frac{1}{\|w\|}$$

$$\text{s.t} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i$$



  - The attained margin is now given by $\frac{1}{\|w\|}$
  - Only a few of the classification constraints are relevant ➔ **support vectors**

- Constrained optimization
  - We can directly solve this using commercial quadratic programming (QP) code

Solving optimal margin classifier

- Recall our opt problem:

$$\max_{w,b} \quad \frac{1}{\|w\|}$$
$$\text{s.t} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i$$

- This is equivalent to

$$\min_{w,b} \quad \frac{1}{2} w^T w$$
$$\text{s.t} \quad 1 - y_i(w^T x_i + b) \leq 0, \quad \forall i \qquad (*)$$

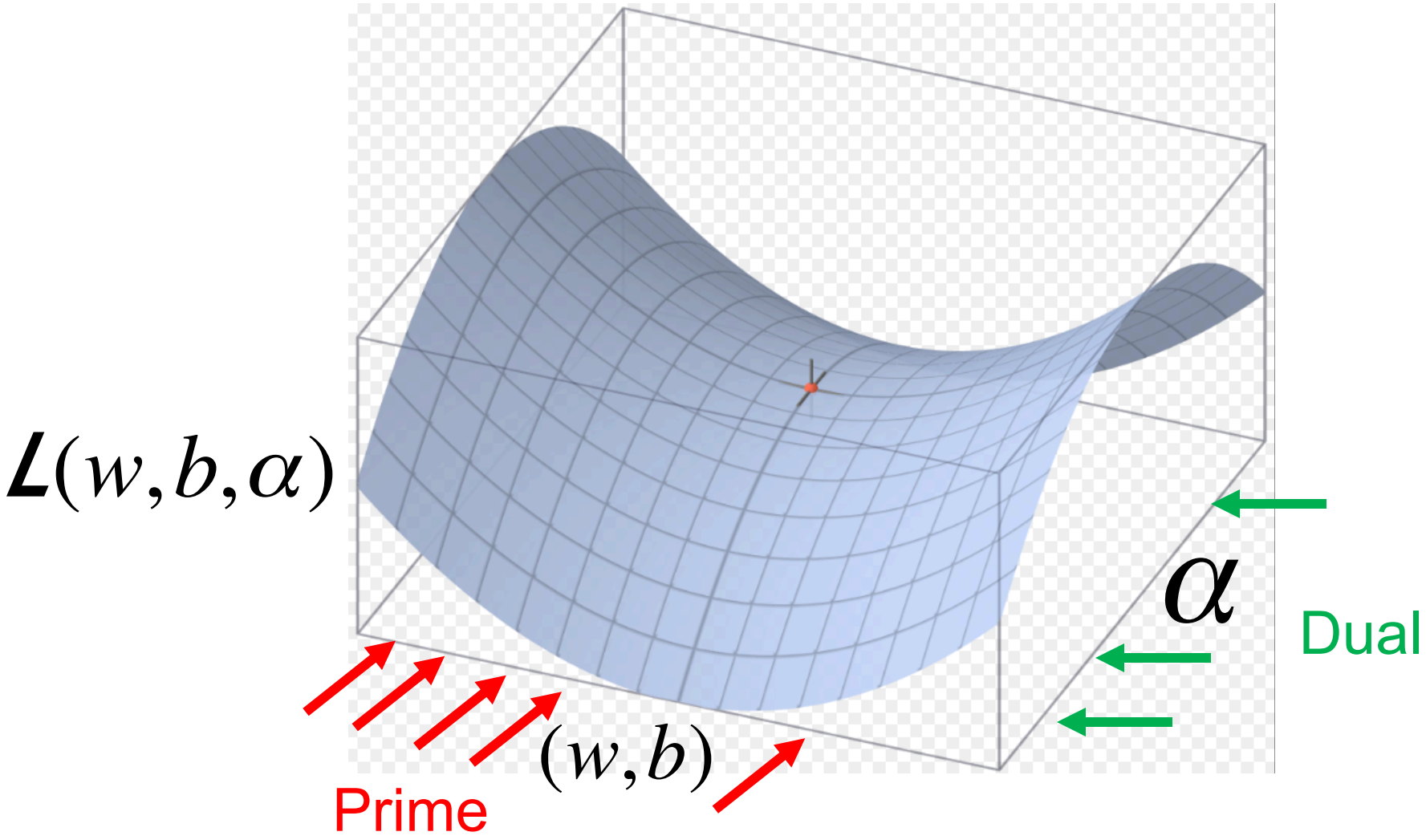
- Write the Lagrangian:

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{m} \alpha_i \left[y_i(w^T x_i + b) - 1 \right]$$

  - Note that (*) can be reformulated as $\min_{w,b} \max_{\alpha_i \geq 0} \mathcal{L}(w,b,\alpha)$

    Now we solve its **dual problem**: $\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w,b,\alpha)$

Prime vs. Dual



The Dual problem, cont.

- Now we have the following dual opt problem:

$$\max_{\alpha} \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- This is, (again,) a **quadratic programming** problem.
  - A global maximum of α_i can always be found.
  - After we solve it
  - $\boldsymbol{w}$ can be recovered by

$$w = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

Support vector machines

- Once we have the Lagrange multipliers $\{\alpha_i\}$, we can reconstruct the parameter vector w as a weighted combination of the training examples:

$$w = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$$

- For testing with a new data z
  - Compute

$$w^T z + b = \sum_{i \in SV} \alpha_i y_i \left(\mathbf{x}_i^T z\right) + b$$

    and classify z as class 1 if the sum is positive, and class 2 otherwise
  - Note: w need not be formed explicitly

Interpretation of support vector machines

- The optimal $\boldsymbol{w}$ is a linear combination of a small number of data points. This "sparse" representation can be viewed as data compression as in the construction of kNN classifier

- To compute the weights $\{\alpha_i\}$, and to use support vector machines we need to specify only the inner products (or kernel) between the examples

$$\mathbf{x}_i^T \mathbf{x}_j$$

- We make decisions by comparing each new example $\boldsymbol{z}$ with only the support vectors:

$$y^* = \text{sign}\left(\sum_{i \in SV} \alpha_i y_i \left(\mathbf{x}_i^T z \right) + b \right)$$

Kernelize SVM: non-linear classifier

- To compute the weights $\{\alpha_i\}$, and to use support vector machines we need to specify only the inner products (or kernel) between the examples

- Simply replace <x_i, x_j> by $K(x_i, x_j)$

$$\max_\alpha \ell(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, k$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

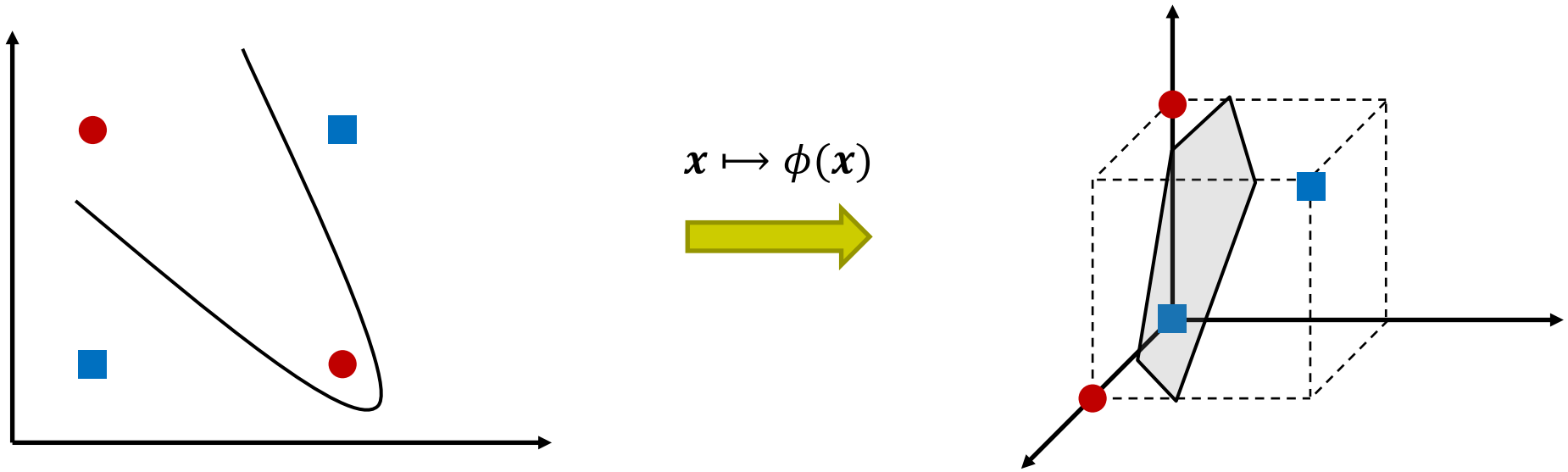
- Choices of Kernels
  - Linear kernel $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$
  - Polynomial kernel $K(\mathbf{x}, \mathbf{x}') = \left(1 + \mathbf{x}^T \mathbf{x}'\right)^p$
  - Radial basis kernel $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right)$

- Meaning of Kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

SVM for Linearly Inseparable Data

- Alternatively, for linearly inseparable data, we can map them to a higher dimensional space
- We search for a linear separating hyperplane in the new space
  - Example: The XOR problem



An Example

1 **(Kernel, 10 pts)** Given the following dataset in 1-d space (Figure 1), which consists of 3 positive data points $\{-1, 0, 1\}$ and 3 negative data points $\{-3, -2, 2\}$.

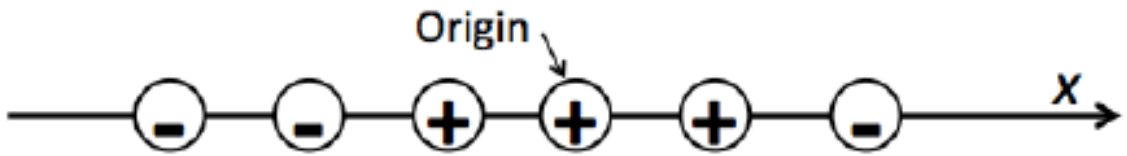


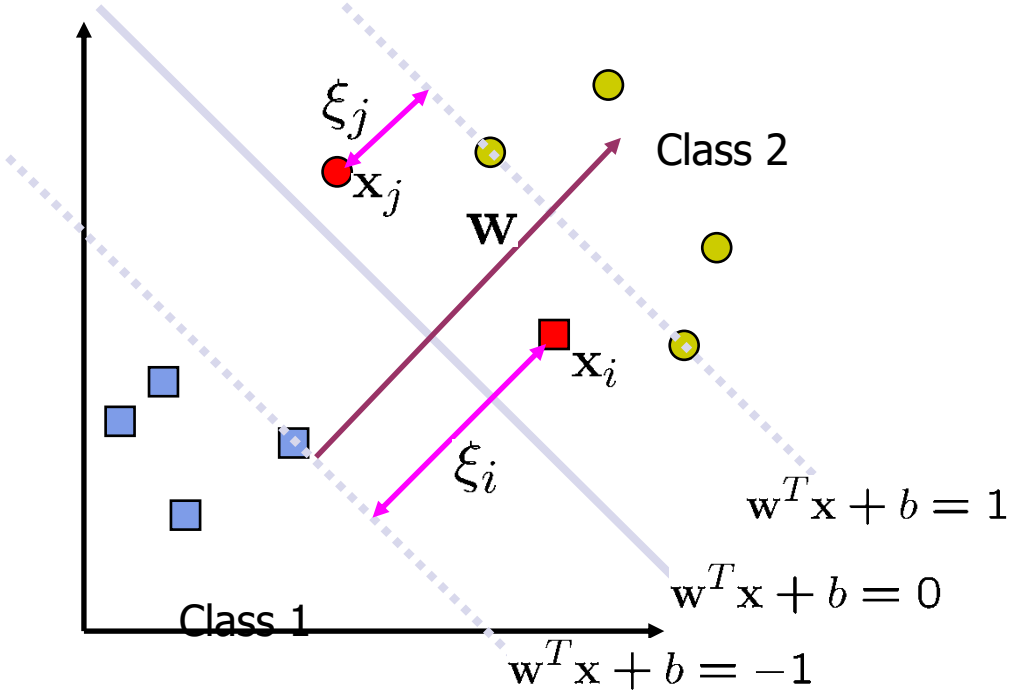
Figure 1: Training Data Set for SVM Classifiers

(1) Find a feature map($\{\mathrm{R}^1 \rightarrow \mathrm{R}^2\}$), which will map the original 1-d data points to 2-d space so that the positive set and negative set are linearly separable with each other. Plot the dataset after mapping in 2-d space.

(2) In your plot, draw the decision boundary given by hard-margin linear SVM. Mark the corresponding support vector(s).

(3) For the feature map you choose, what is the corresponding kernel $K(x_1, x_2)$?

Non-linearly Separable Problems



- We allow "error" ξ_i in classification; it is based on the output of the discriminant function $\boldsymbol{w}^T\boldsymbol{x}+b$
- ξ_i approximates the number of misclassified samples
  - (an upper bound of the approximate error)

Soft Margin SVM

- Now we have a slightly different opt problem:

$$\min_{w,b} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i$$

$$\text{s.t} \quad y_i (w^T x_i + b) \geq 1 - \xi_i, \quad \forall i$$

$$\xi_i \geq 0, \quad \forall i$$

- ξ_i are "slack variables" in optimization
- Note that ξ_i=0 if there is no error for $\mathbf{x}_i$
- ξ_i approximates the number of misclassified samples
- *C* : tradeoff parameter between error and margin

The Optimization Problem

- The dual of this new constrained optimization problem is

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound C on α_i now
- Once again, a QP solver can be used to find α_i

The SMO algorithm

- Consider solving the unconstrained opt problem:

$$\max_{\alpha} W(\alpha_1, \alpha_2, \ldots, \alpha_m)$$

- We've already seen this kind of opt algorithms!

- Coordinate ascend:

Sequential minimal optimization

- Constrained optimization:

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- Question: can we do coordinate ascend along one direction at a time (i.e., hold all $\alpha_{[-i]}$ fixed, and update α_i?)

The SMO algorithm

Repeat till convergence

1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).

2. Re-optimize J(α) with respect to α_i and α_j, while holding all the other α_k 's ($k \neq i; j$) fixed.

An Example

(SMO, 5 pts) Suppose we are given 4 data points in 2-d space: $x_1 = (0, 1)$, $y_1 = -1$; $x_2 = (2, 0)$, $y_2 = +1$; $x_3 = (1, 0)$, $y_3 = +1$; and $x_4 = (0, 2)$, $y_4 = -1$. We will use these 4 data points to train a soft-margin linear SVM. Let $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ be the Lagrange multipliers for x_1, x_2, x_3, x_4 respectively. And also let the regularization parameter C be 100.

(1) Write down the dual optimization formulation for this problem

(2) Suppose we initialize $\alpha_1 = 5, \alpha_2 = 4, \alpha_3 = 8, \alpha_4 = 7$. And we want to update α_1 and α_4 (keep α_2 and α_3 fixed) in the 1st iteration. Derive the update equations for α_1 and α_4 (in terms of α_2 and α_3). What are the values for α_1 and α_4 after update?

An Example: Solutions

(1) Write down the dual optimization formulation for this problem

Solutions:

$$\text{argmax}_{\alpha_1,\alpha_2,\alpha_3,\alpha_4}\alpha_1+\alpha_2+\alpha_3+\alpha_4-1/2(\alpha_1^2+4\alpha_2^2+\alpha_3^2+4\alpha_4^2+4\alpha_2\alpha_3+4\alpha_1\alpha_4)$$

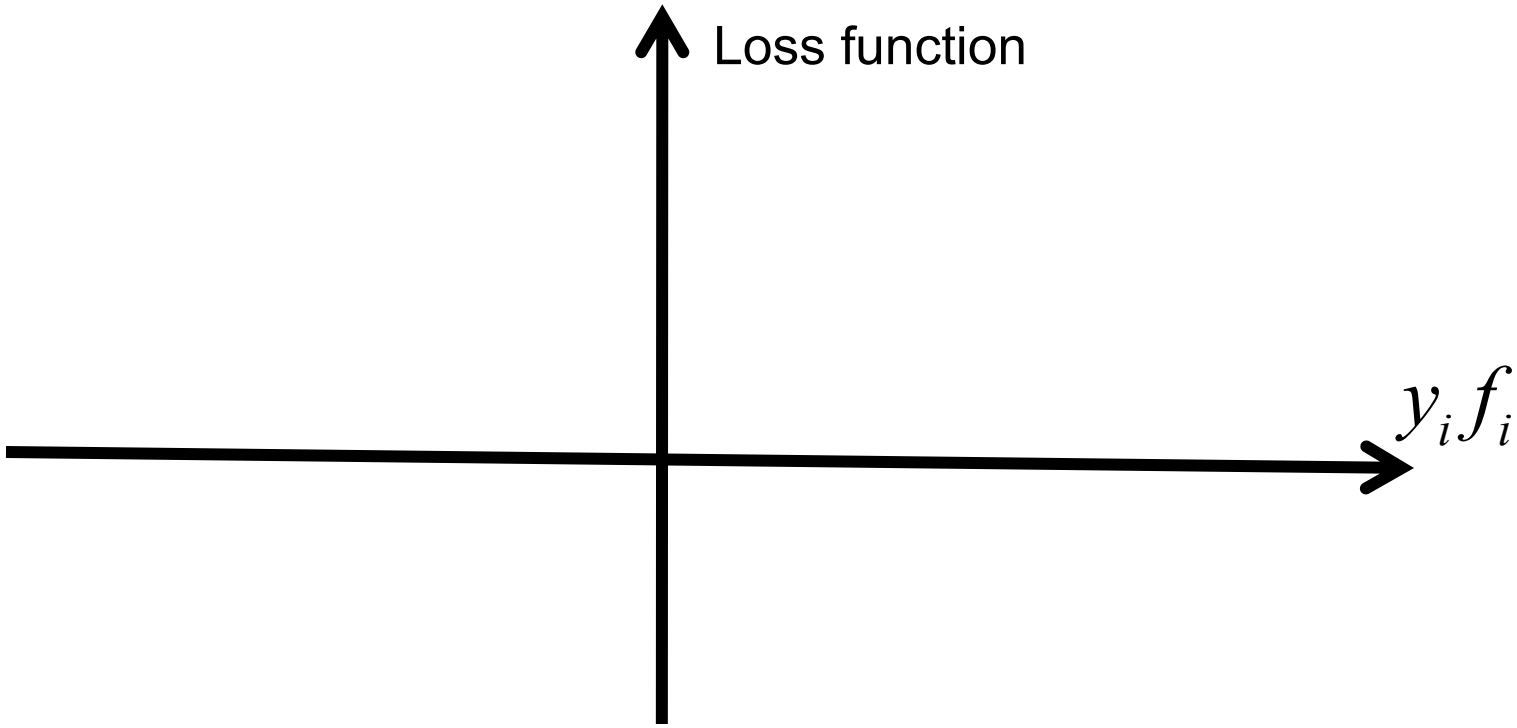
$$\text{Subject to :} 0 \leq \alpha_1, \alpha_2, \alpha_3, \alpha_4 \leq 100$$

$$-\alpha_1+\alpha_2+\alpha_3-\alpha_4=0 \quad (3)$$

(2) Suppose we initialize $\alpha_1 = 5, \alpha_2 = 4, \alpha_3 = 8, \alpha_4 = 7$. And we want to update α_1 and α_4 (keep α_2 and α_3 fixed) in the 1st iteration. Derive the update equations for α_1 and α_4 (in terms of α_2 and α_3). What are the values for α_1 and α_4 after update?

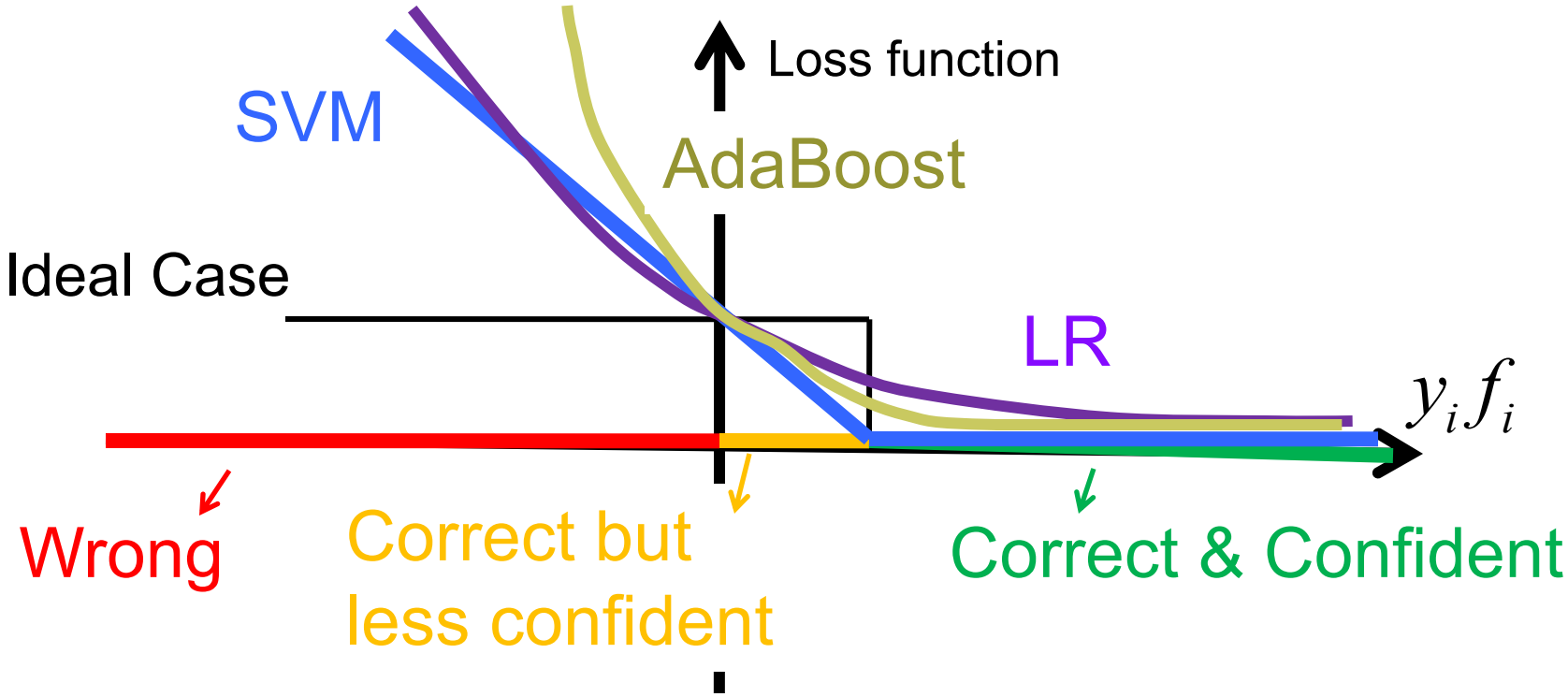
Solutions: $\alpha_1 = \alpha_2 + \alpha_3 = 12$ and $\alpha_4 = 0$.

Logistic Regression vs. SVM (vs. boosting)



How to get a good and confident classifier based on data set?

Logistic Regression vs. SVM (vs. boosting)



How to get a good and confident classifier based on data set?

SVM vs. Logistic Regression

- SVM as unconstraint opt problem

2 (**Hinge Loss, 5 pts**) Given m training data points $\{x_i, y_i\}_{i=1}^m$, remember that the soft-margin linear SVM can be formalized as the following constrained quadratic optimization problem:

$$\begin{aligned} &\text{argmin}_{\{w,b\}} \frac{1}{2} w^t w + C \sum_1^m \epsilon_i \\ &\text{Subject to : } y_i(w^t x_i + b) \geq 1 - \epsilon_i \\ &\epsilon_i \geq 0 \; \forall i \end{aligned} \tag{1}$$

(1) Prove that the above formulation is equivalent to the following unconstrained quadratic optimization problem:

$$\text{argmin}_{\{w,b\}} w^t w + \lambda \sum_1^m \max(1 - y_i(w^t x_i + b), 0) \tag{2}$$

- LR as unconstraint opt problem

$$\text{argmin}_{\{w,b\}} w^t w + \lambda \sum_{i=1}^n (\ln(1 + \exp(-w^T x_i)) + (1 - y_i) w^T x_i)$$

SVM Summary

- Pros
  - Elegant mathematical formulation, guaranteed global optimal with optimization
  - Trains well on small or medium data sets
  - Flexibility through kernel functions
  - Conformity with semi-supervised training
- Cons
  - Not naturally scalable to large data sets

SVM Related Links

- SVM Website: http://www.kernel-machines.org/
- Representative implementations
  - **LIBSVM**: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - **SVM-light**: simpler but performance is not better than LIBSVM, support only binary classification and only in C
  - **SVM-torch**: another recent implementation also written in C

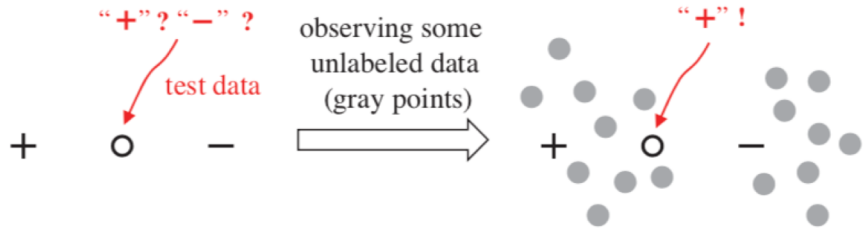
Chapter 9. Classification: Advanced Methods

- Feature Selection
- Support Vector Machines
- Weakly Supervised Learning 
- Classification with Rich Data Type
- Summary

Weakly Supervised Learning

- Semi-supervised learning
- Active learning
- Transfer learning
- Distant supervision
- Zero-shot learning

Semi-supervised Learning



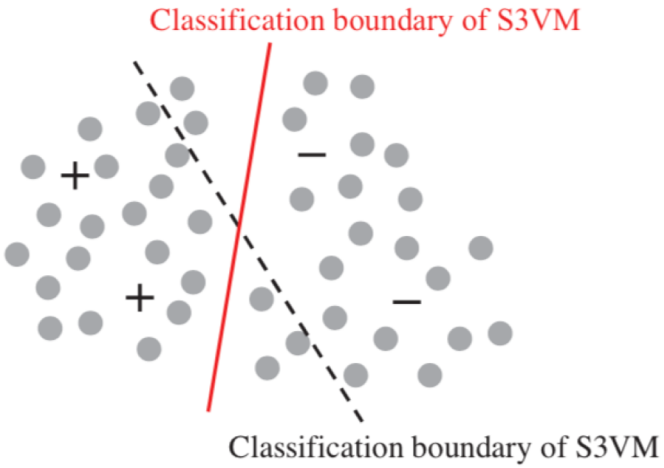
- **Goal**: uses labeled data and unlabeled data to build a classifier
- **Self-training**

1. Select a learning method such as, say, Bayesian classification. Build the classifier using the labeled data, X_l.
2. Use the classifier to label the unlabeled data, X_u.
3. Select the tuple $x \in X_u$ having the highest confidence (most confident prediction). Add it and its predicted label to X_l.
4. Repeat (i.e., retrain the classifier using the augmented set of labeled data).

- **Co-training**

1. Define two separate nonoverlapping feature sets for the labeled data, X_l.
2. Train two classifiers, f_1 and f_2, on the labeled data, where f_1 is trained using one of the feature sets and f_2 is trained using the other.
3. Classify X_u with f_1 and f_2 separately.
4. Add the most confident $(x, f_1(x))$ to the set of labeled data used by f_2, where $x \in X_u$. Similarly, add the most confident $(x, f_2(x))$ to the set of labeled data used by f_1.
5. Repeat.

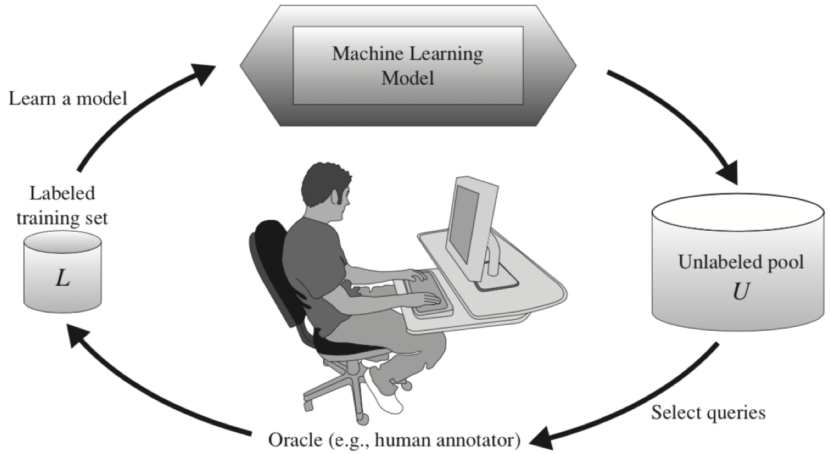
When does SSL Work?



- **Clustering assumption**
  - Data tuples from same cluster are likely to share same label
  - An example: S3VM
    - Seeking a max-margin hyperplane (same as SVM)
    - avoiding to disrupt the clustering structure of unlabeled tuples (go through low-density region of unlabeled data)
- **Manifold assumption**
  - a pair of close tuples are likely to share the same class label
  - An example: graph-based SSL

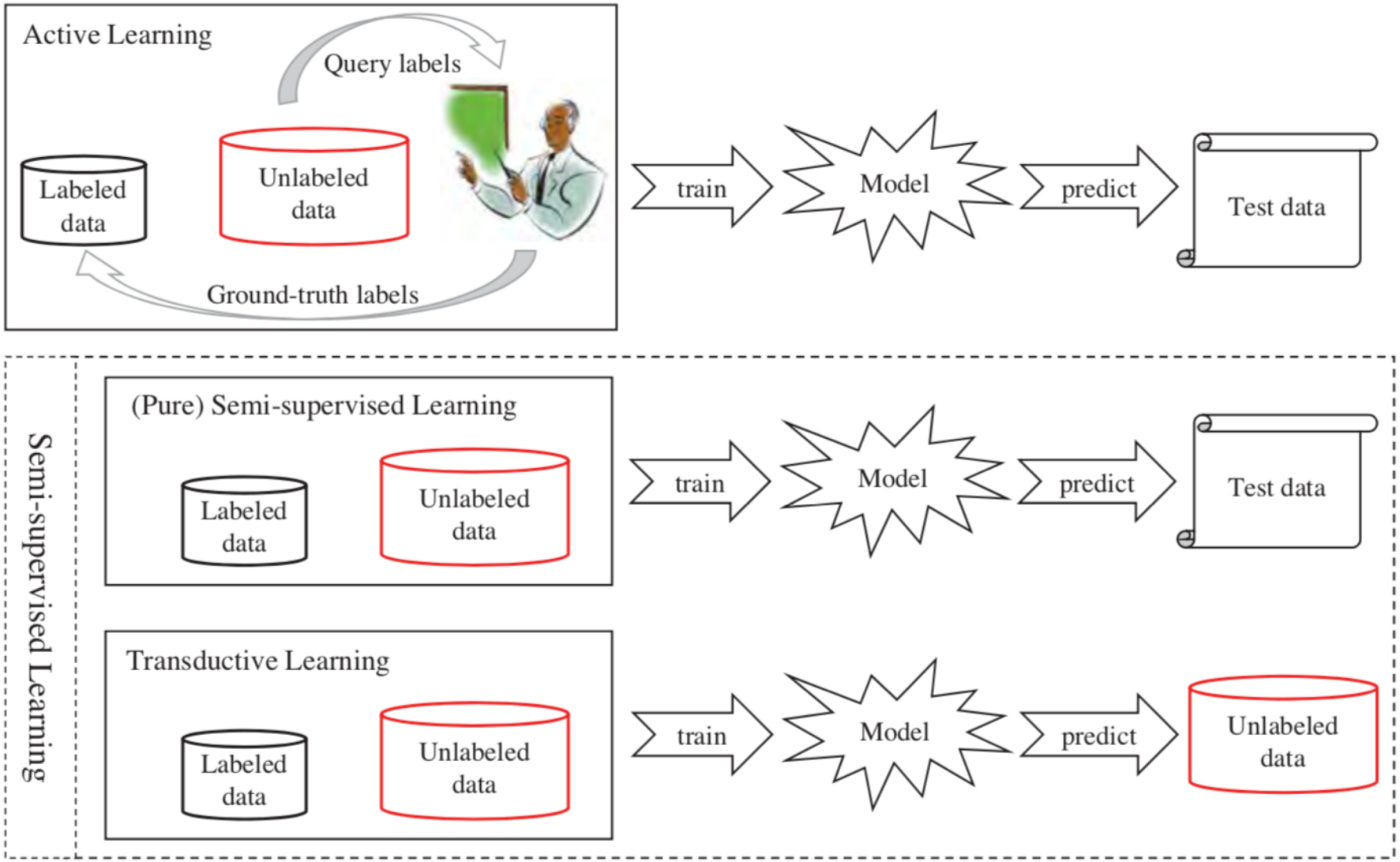
Active Learning

- **Goal**: find 'the best' unlabeled data samples to ask the oracle for their labels, so as to maximally improve the classification performance.
- **Pool-based active learning**



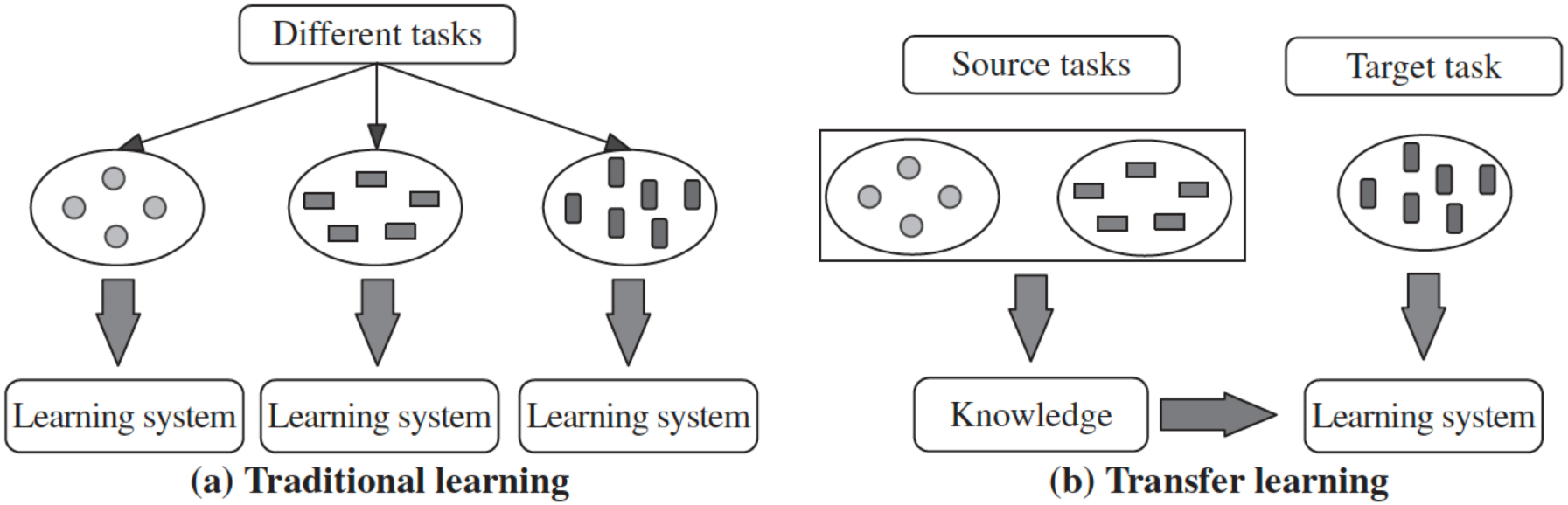
- **Key:** how to choose the data tuples to be queried
  - Uncertainty sampling
  - Query-by-committee
  - Version space
  - Decision-theoretic approach

Active Learning vs. SSL (vs. transductive learning)



Transfer Learning

- **Goal**: aims to extract the knowledge from one or more source tasks and apply the knowledge to a target task
- **An example**
  - Source task: review sentiment classification on electronics
  - Target task: review sentiment classification on movies



TrAdaBoost

- **General strategy: Instance-based transfer learning**: reweight some of the data from the source task and uses it to learn the target task.
  - Intuition: `transfer' most relevant/similar data from source to target
- **Details**
  - **Boosting:** each base learner pays more attention to difficult training examples (e.g., misclassified by the previous base learner)
  - **TrAdaBoost:** if a source tuple is misclassified, reduce its weight (e.g., not very relevant to the target task)
- **Key challenge in transfer learning**: negative transfer
  - Quantify the difference between the source and target tasks
  - Transfer margin, divergence metric
- **Related problems**
  - Multi-task learning; pre-training+fine-tuning

Distant Supervision

- **Goal**: automatically generate a large number of labelled tuples.
- **Characteristic of the generated labels**
  - Noisy, but large in volume
- **Example #1:** Twitter sentiment classification (positive vs. negative)
  - if a tweet contains :-), treat it as a positive tuple
  - If a tweet contains :-(, treat it as a negative tuple
- **Example #2:** Twitter category classification (e.g., news, health, science, games, etc.)
  - If a tweet contains a URL, use the ODP (open directory project) category of the URL as the label of the tweet
  - If a tweet contains a Youtube video link, treat the label of Youtube video as the label of the tweet

Distant Supervision

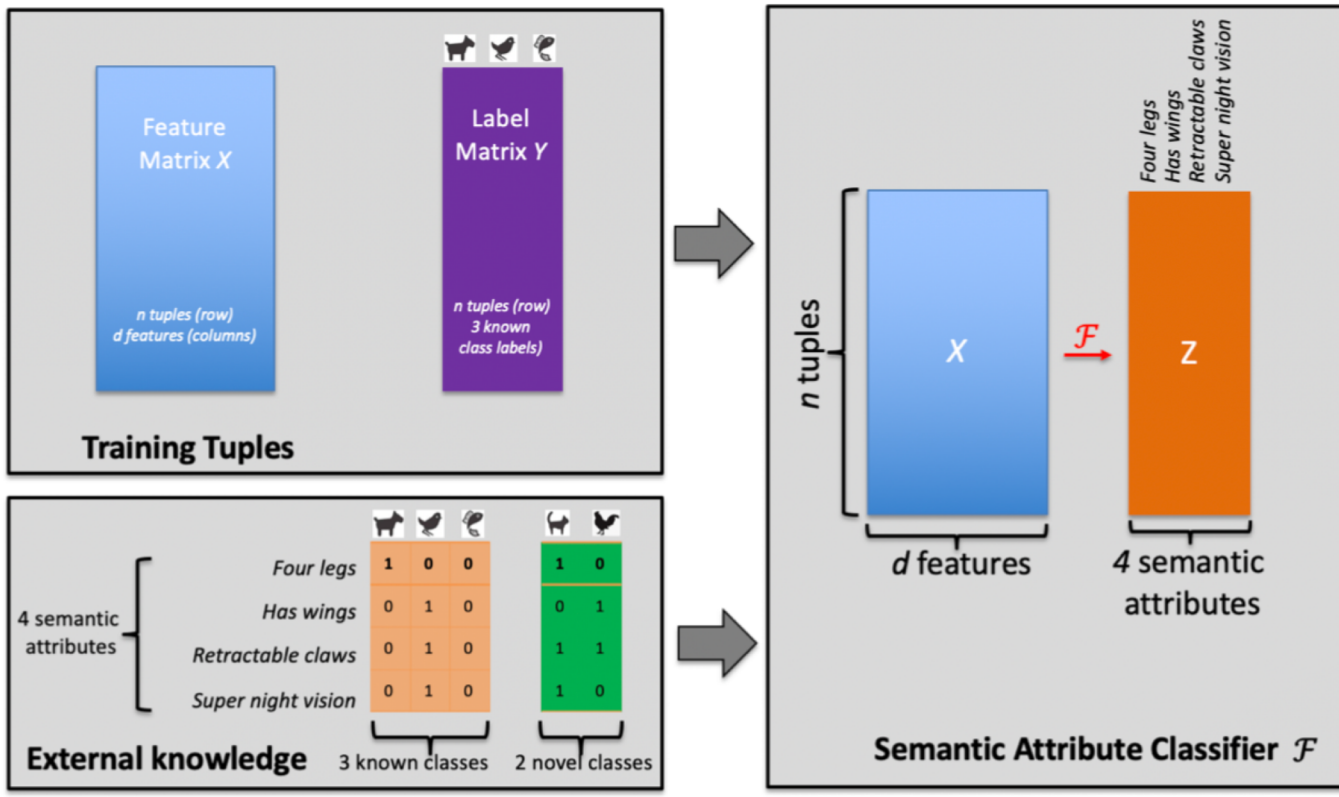
- **Applications**
  - Social media post classification
  - Relation extraction for NLP
- **Active research area:** how to obtain *labeling function*
- **Main Challenge: noisy labels**
  - :) in a tweet post could have neutral or even negative sentiment

Zero-shot Learning

- **A motivating example**:
  - A trained classifier to classify an animal image into **owl** vs. **dog** vs. **fish**.
  - But, the test image is actually about **cat.**
- **Goal:** predict a test tuple whose class label was never observed during the training stage
- **General strategies**: leverage external knowledge or side information
- **Applications:**
  - Image recognition
  - Neural activity recognition
  - Graph anomaly detection
- **Generalized zero-shot learning**: the test image might come from known or novel classes

Semantic attribute classifier

- **Details**
  - Use the training tuples to build semantic attribute classifier
  - Use the semantic attribute classifier to infer semantic attributes
  - Use the semantic attributes to predict the novel class
- **Key Idea**: leverage the semantic attributes to transfer the output of the semantic classier



Chapter 9. Classification: Advanced Methods

- Feature Selection
- Support Vector Machines
- Weakly Supervised Learning
- Classification with Rich Data Type 
- Summary

Classification with Rich Data Type

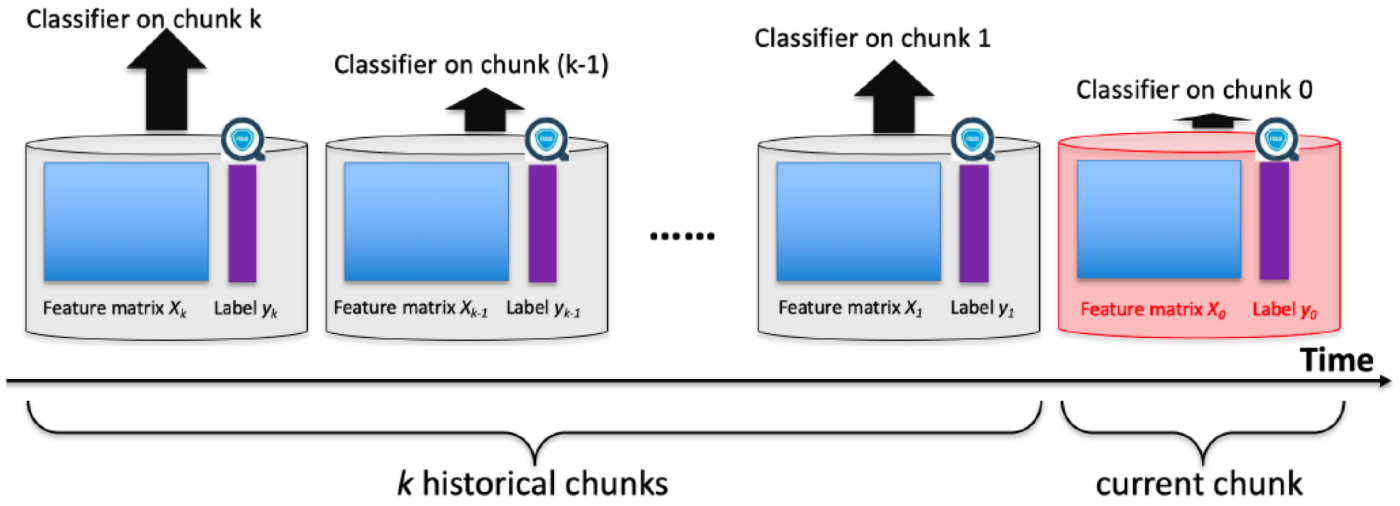
- Stream Data Classification
- Sequence Classification
- Graph Data Classification

Stream Data Classification

- **A motivating example**:
  - Data mining tool to detect fraudulent transaction
  - the transactions arrive sequentially at different times in a stream fashion
  - Training set → build classification model → receiving new data → retraining …
- **Challenges:**
  - High arrival speed
  - Infinite length
  - One-pass constraint
  - Concept drifting

Stream Data Classification via Ensemble

- **Key Ideas:**
  - Only use the most recent chunk to train the new classifier → high arrival rate of the stream data.
  - Each incoming data tuple is accessed once to train the current classifier as well as to update the weights of individual classifiers --> one pass constraint.
  - Adjust the weights of the individual classifiers to pay more attention to the most relevant chunks → captures the drifted class concept over time.



Stream Data Classification

- **Other applications**
  - Marketing
  - Network monitoring
  - Sensor networks
- **VFDT: very fast decision tree**
  - Hoeffding trees: build the decision tree by using a sampled subset of training tuples,
  - Sliding window mechanism to obtain classifier to focus on the most recent stream data.

Sequence Classification

- **Sequence: an ordered list of values** $(x^1, x^2, ..., x^T)$
  - A sentence
  - A DNA segment
  - List of transactions of a customer over time
- **Sequence Classification:** build a classifier to predict the label of a given sequence
  - the positive vs. negative sentiment of the sentence in NLP
  - the gene coding area vs. non-coding area in genomic analysis,
  - high-value vs. ordinary customer in market analysis.
- **Other forms of sequence classification:** predict label for each time stamp

Sequence Classification via Feature Engineering

- **General Ideas**
  - Convert the input sequence to a vector of features
  - Train a conventional classifier
- **For symbolic sequence: n-gram**
  - Input DNA segment: `ACCCCCGT'
  - Output:

Unigrams | Bigrams

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Candidate features	A	C	G	T	AC	AG	AT	CG	CT	CA	GA	GC	GT	TA	TC	TG	AA	CC	GG	TT
Feature vector (binary)	1	1	1	1	1	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
Feature vector (frequency)	1	5	1	1	1	0	0	1	0	0	0	0	1	0	0	0	0	4	0	0

- **For numerical sequence: convert to symbolic sequence by discretization**
- **More recent approaches:** RNN and related techniques

Sequence Classification via Distance/Kernel Function

- **Observations:**
  - K-NN classifier: rely on distance or similarity measure
  - Non-linear SVM: rely on a kernel function
- **General strategy: define appropriate distance measure or kernel function on sequence**
- **Distance measure on sequence**
  - Euclidean distance
  - Dynamic time warping (DTW): to account for distortion in time
- **Kernel function on sequence**
  - String kernel

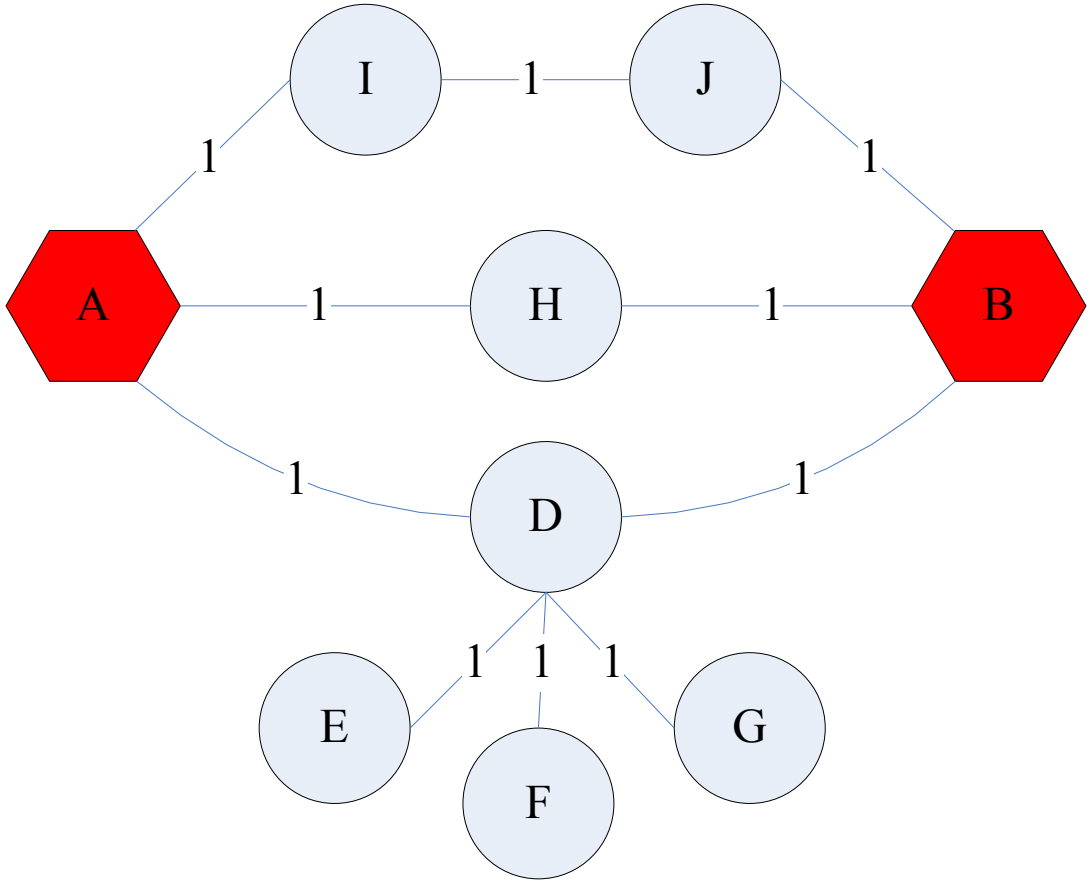
Graph Data Classification

- **Graph Data (aka network data): collection of nodes linked by edges**
  - Social networks
  - Power grid
  - Transaction networks
  - Biological networks
- **Node-level vs. graph-level classification**
  - Node-level classification: webpage classification
  - Graph-level classification: toxicity classification

Graph Data Classification Methods

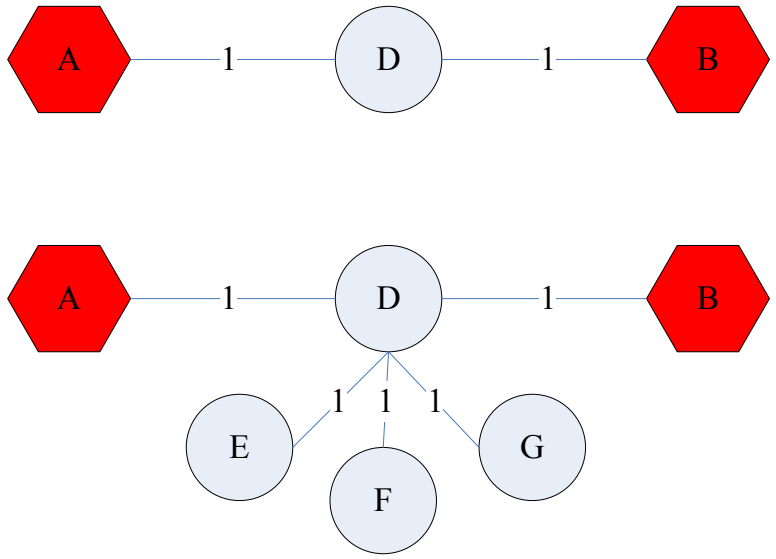
- **Graph Data Classification via Feature engineering**
  - Extract a set of features describing each node or each graph
  - Fed into a conventional classier to build a node-level or graph-level classifier
    - Node-level features: degree, # of triangles, centrality, pagerank, etc.
    - Graph-level features: size, diameter, # of triangles, etc.
    - Alternative methods: using deep learning (GNN) to “automatically” extract node/graph level features
- **Graph Data Classification via Proximity measure**
  - Measure proximity between nodes or graphs
  - Build KNN type of classifier

Node Proximity on Graph: What?



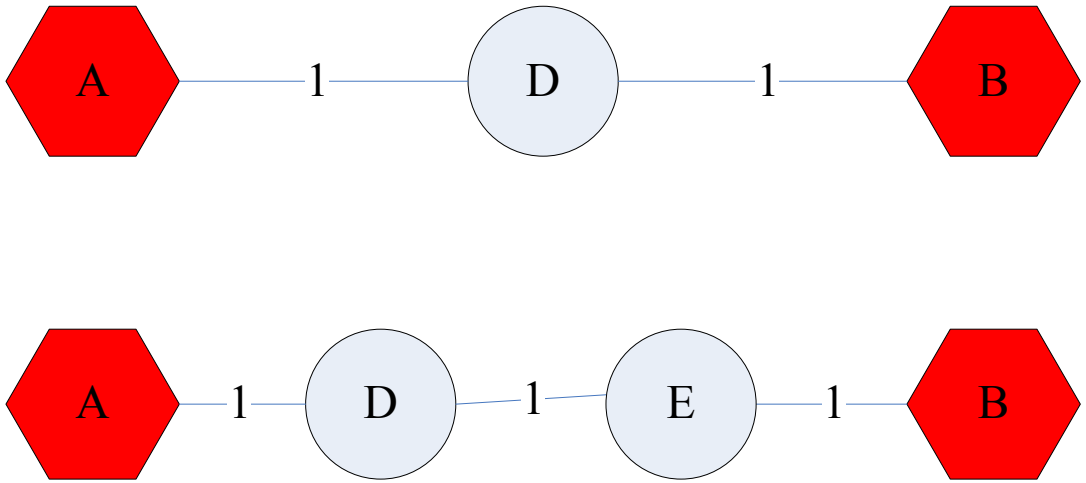
a.k.a Relevance, Closeness, 'Similarity'…

Why not shortest path?



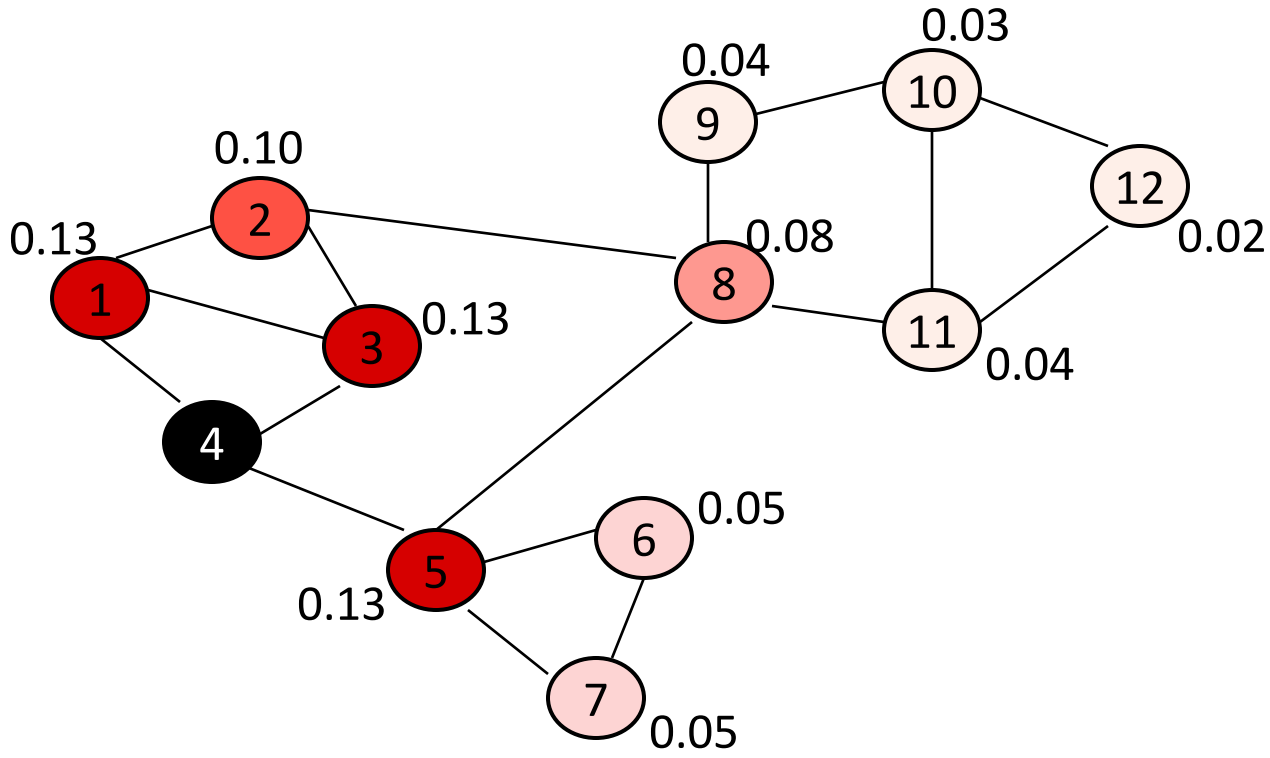
‘pizza delivery guy’ proble

Why not max. netflow?



No punishment on long paths

Random Walk with Restart



	Node 4
Node 1	0.13
Node 2	0.10
Node 3	0.13
Node 4	0.22
Node 5	0.13
Node 6	0.05
Node 7	0.05
Node 8	0.08
Node 9	0.04
Node 10	0.03
Node 11	0.04
Node 12	0.02

Nearby nodes, higher scores

More red, more relevant

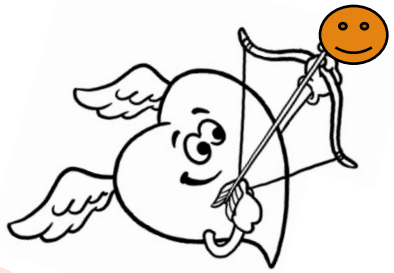
Ranking vector

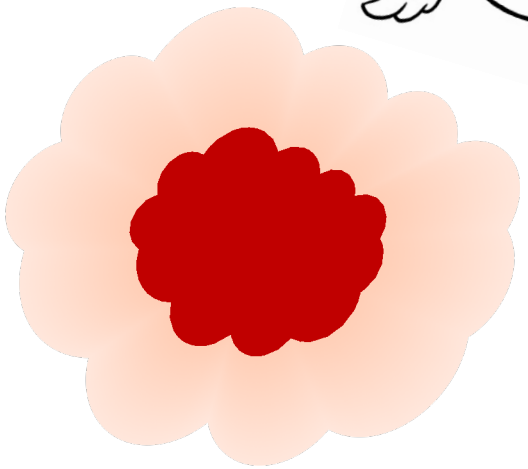
$\vec{r}_4$

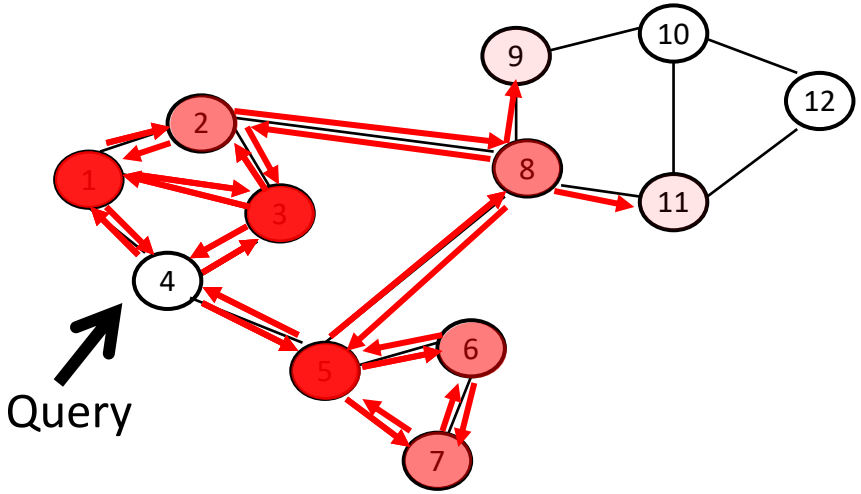
RWR: Think of it as Wine Spill

1. Spill a drop of wine on cloth
2. Spread/diffuse to the neighborhood

RWR: Wine Spill on a Graph





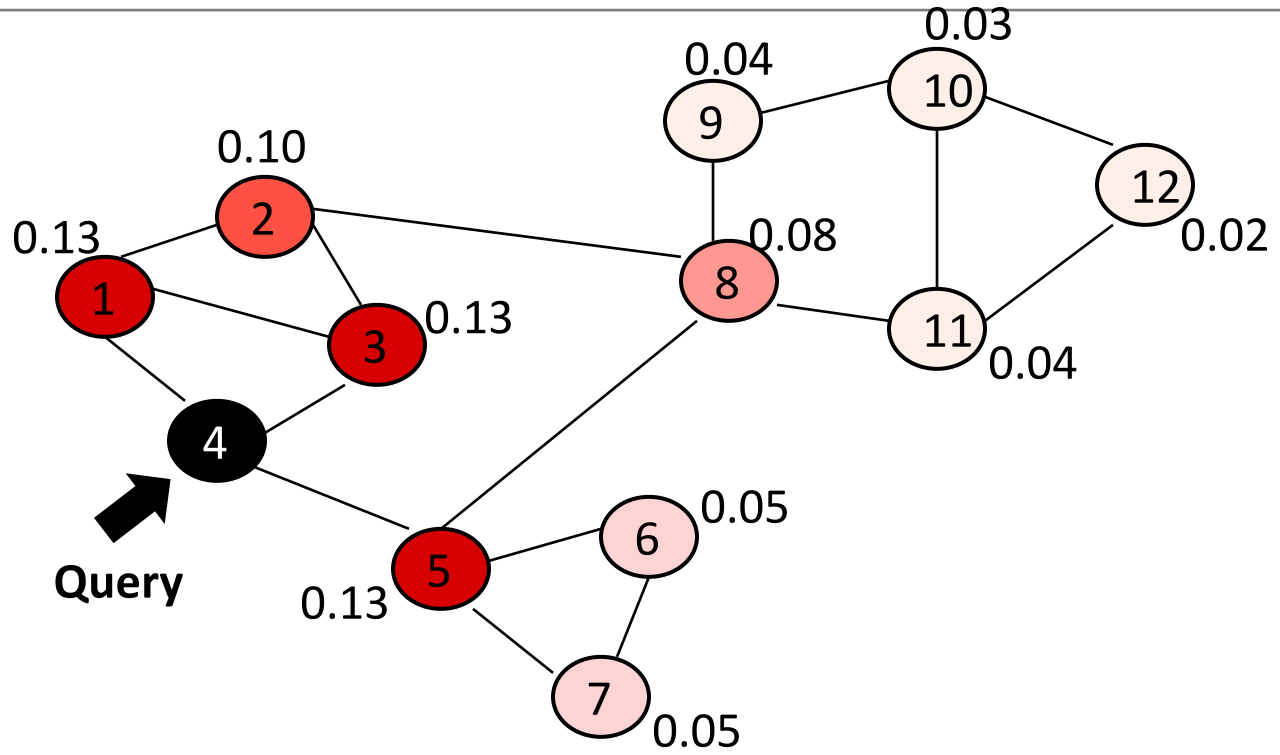


wine spill on cloth

RWR on a graph

Same Diffusion Eq.

Random Walk with Restart: Formulation



	Node 4
Node 1	0.13
Node 2	0.10
Node 3	0.13
Node 4	0.22
Node 5	0.13
Node 6	0.05
Node 7	0.05
Node 8	0.08
Node 9	0.04
Node 10	0.03
Node 11	0.04
Node 12	0.02

Nearby nodes, Higher scores

More red, More relevant

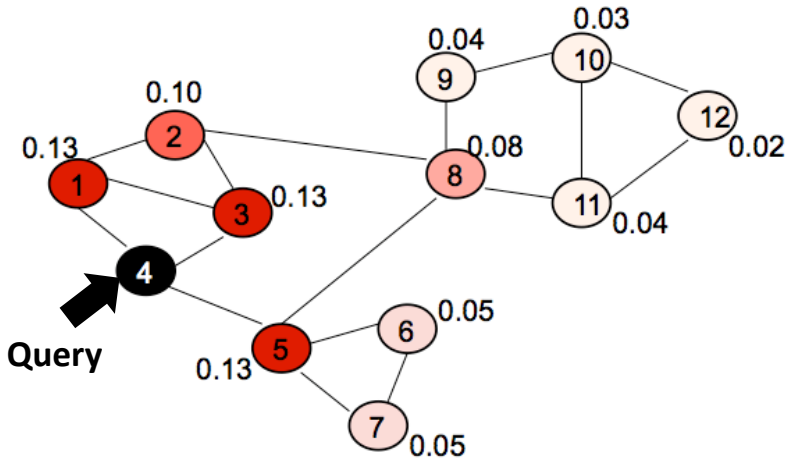
Ranking vector

r_4

$$r_i = c \times A \times r_i + (1\text{-}c) \times e_i$$

"Maxwell Equation" for Web -- *Soumen Chakrabarti*

Random Walk with Restart: Optimization



	Node 4
Node 1	0.13
Node 2	0.10
Node 3	0.13
Node 4	0.22
Node 5	0.13
Node 6	0.05
Node 7	0.05
Node 8	0.08
Node 9	0.04
Node 10	0.03
Node 11	0.04
Node 12	0.02

Ranking vector

r_4

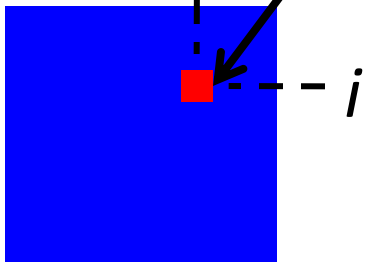
$$r_i = c \times A \times r_i + (1\text{-}c) \times e_i$$

$$= \textbf{argmin}\ \underbrace{c r_i'(I - A) r_i}_{\text{Network Smoothness}} + \underbrace{(1\text{-}c) \times \| r_i - e_i \|^2}_{\text{Query Preference}}$$

An Optimization Viewpoint of "Maxwell Equation" for Web (Symmetric *A*)

Why RWR is a good score?

$$Q = (I - c\tilde{W})^{-1} =$$



$Q(i, j) \propto r_{i,j}$

$\tilde{W}$: adjacency matrix.
c: damping factor

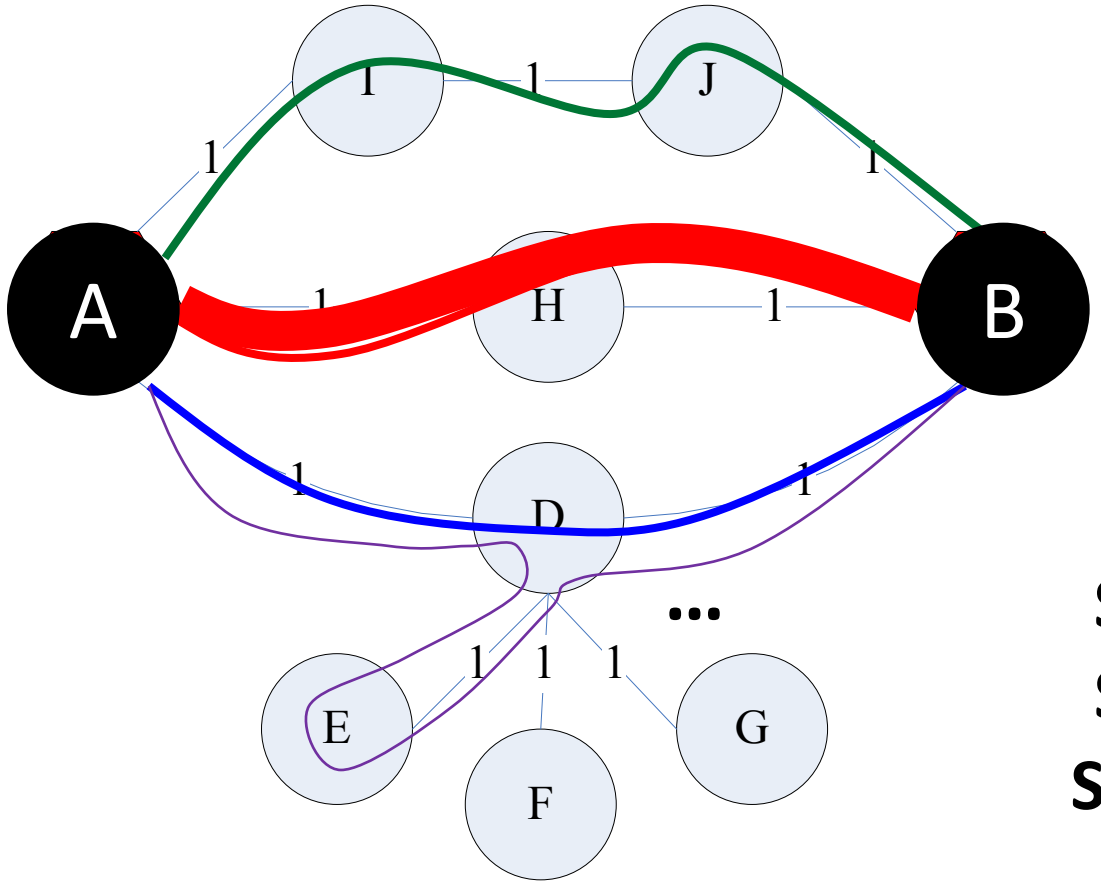
$$Q = c\tilde{W} + c^2\tilde{W}^2 + c^3\tilde{W}^3 + \cdots$$

all paths from *i* to *j* with length **1**

all paths from *i* to *j* with length 2

all paths from *i* to *j* with length 3

Why *RWR* is A Good Score?

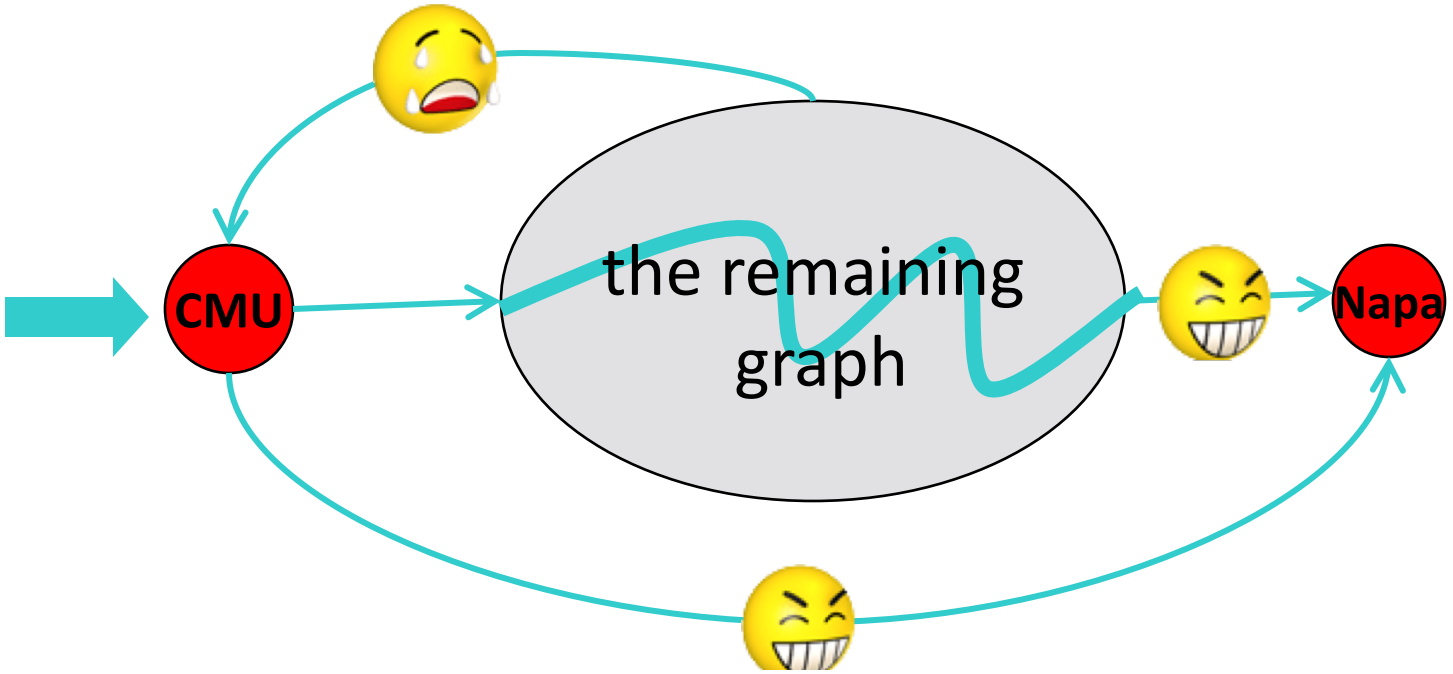


Prox (A, B) =
Score (Red Path) +
Score (Green Path) +
Score (Blue Path) +
Score (Purple Path) + …

High proximity ⟺ many, short, heavy-weighted paths

Variant: escape probability

- Define Random Walk (RW) on the graph
- Esc_Prob(CMU→Napa)
  - Prob (starting at CMU, reaches Napa before returning to CMU)



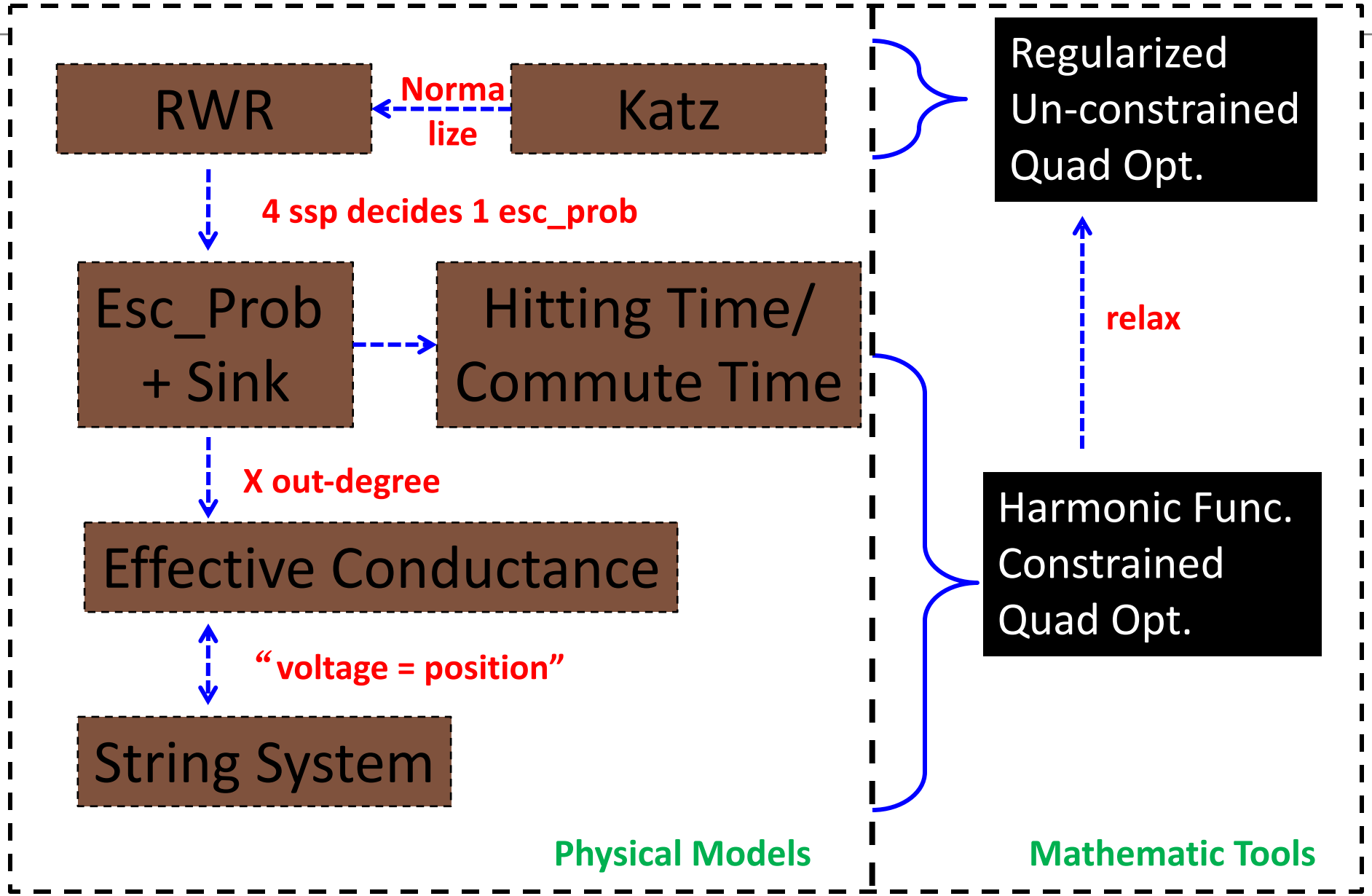
Esc_Prob = Pr (smile before cry)

Other Variants

- Other measure by RWs
  - Commute Time/Hitting Time [Fouss+]
  - SimRank [Jeh+]
- Equivalence of Random Walks
  - Electri
    - EC [D
  - String
- Katz [Katz], [Huang+], [Scholkopf+]
- Matrix-Forest-based Alg [Chobotarev+]

All are “related to” or “similar to” random walk with restart!

Chaptering different proximity measurements



Summary

- Feature Selection
  - Filter methods, wrapper methods and embedded methods
- Support Vector Machines
  - Large margin approach, Kernel tricks
- Weakly Supervised Learning
  - Semi-supervised learning, Active learning, Transfer learning, Distant supervision, Zero-shot learning
- Classification with Rich Data Type
  - Stream, sequence, and graph

References (1)

- C. M. Bishop, Neural Networks for Pattern Recognition. Oxford University Press, 1995
- C. M. Bishop, Pattern Recognition and Machine Learning. Springer, 2006
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth International Group, 1984
- C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2): 121-168, 1998
- N. Cristianini and J. Shawe-Taylor, Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000
- H. Yu, J. Yang, and J. Han. Classifying large data sets using SVM with hierarchical clusters. KDD'03
- A. J. Dobson. An Introduction to Generalized Linear Models. Chapman & Hall, 1990
- R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification, 2ed. John Wiley, 2001
- T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer-Verlag, 2001
- S. Haykin, Neural Networks and Learning Machines, Prentice Hall, 2008
- D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning, 1995

References (2)

- Chandrashekar, Girish, and Ferat Sahin. "A survey on feature selection methods." Computers & Electrical Engineering 40.1 (2014): 16-28.
- He, Xiaofei, Deng Cai, and Partha Niyogi. "Laplacian score for feature selection." Advances in neural information processing systems. 2006.
- Weston, Jason, et al. "Feature selection for SVMs." Advances in neural information processing systems. 2001.
- Zhu, Xiaojin, and Andrew B. Goldberg. "Introduction to semi-supervised learning." Synthesis lectures on artificial intelligence and machine learning 3.1 (2009): 1-130.
- Meyers, Chet, and Thomas B. Jones. Promoting Active Learning. Strategies for the College Classroom. Jossey-Bass Inc., Publishers, 350 Sansome Street, San Francisco, CA 94104, 1993.
- Pan, Sinno Jialin, and Qiang Yang. "A survey on transfer learning." IEEE Transactions on knowledge and data engineering 22.10 (2009): 1345-1359.
- Go, Alec, Richa Bhayani, and Lei Huang. "Twitter sentiment classification using distant supervision." CS224N project report, Stanford 1.12 (2009): 2009.
- Socher, Richard, et al. "Zero-shot learning through cross-modal transfer." Advances in neural information processing systems. 2013.