Linear Classifier and Support Vector Machines

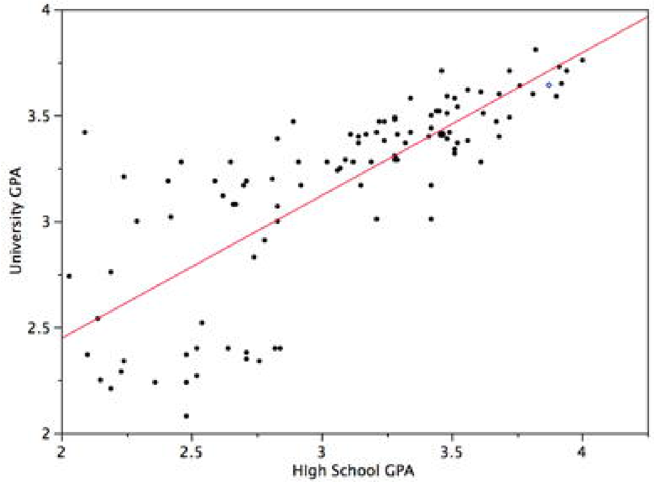
Outline

- Linear Classifier
- Support Vector Machines

Linear Classifier

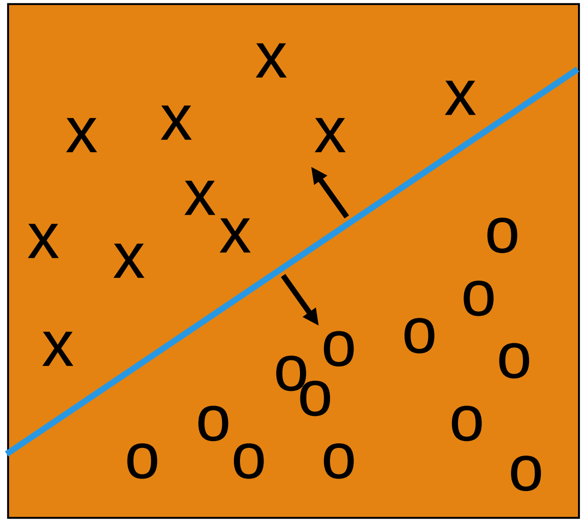
Linear Regression vs. Linear Classifier

- Linear regression
  - Data modeled to fit a straight line
    - *Linear equation: $Y = w X + b$*
  - Often uses the least-square method to fit the line
  - Used to predict continuous values



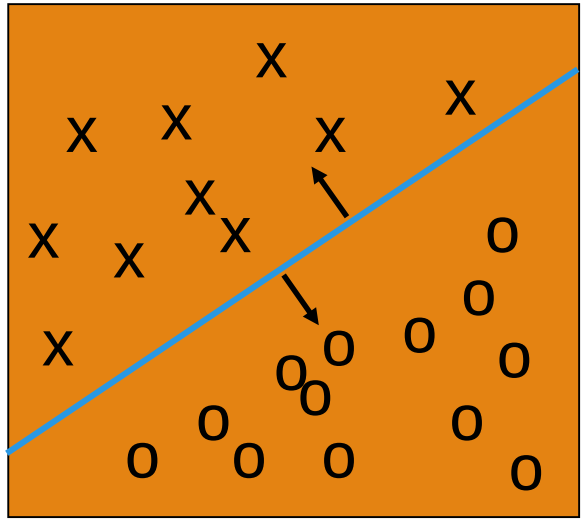
(Onlinestatbook.com, 2013)

- Linear Classifier
  - Built a classification model using a straight line
  - Used for (categorical data) binary classification



Linear Classifier: General Ideas

- Binary Classification
  - $f(x)$ is a linear function based on the example's attribute values
    - The prediction is based on the value of $f(x)$
    - Data above the blue line belongs to class 'x' (i.e., $f(x) > 0$)
    - Data below the blue line belongs to class 'o' (i.e., $f(x) < 0$)
- Classical Linear Classifiers
  - Linear Discriminant Analysis (LDA) (not covered)
  - Logistic Regression
  - Perceptron
  - SVM

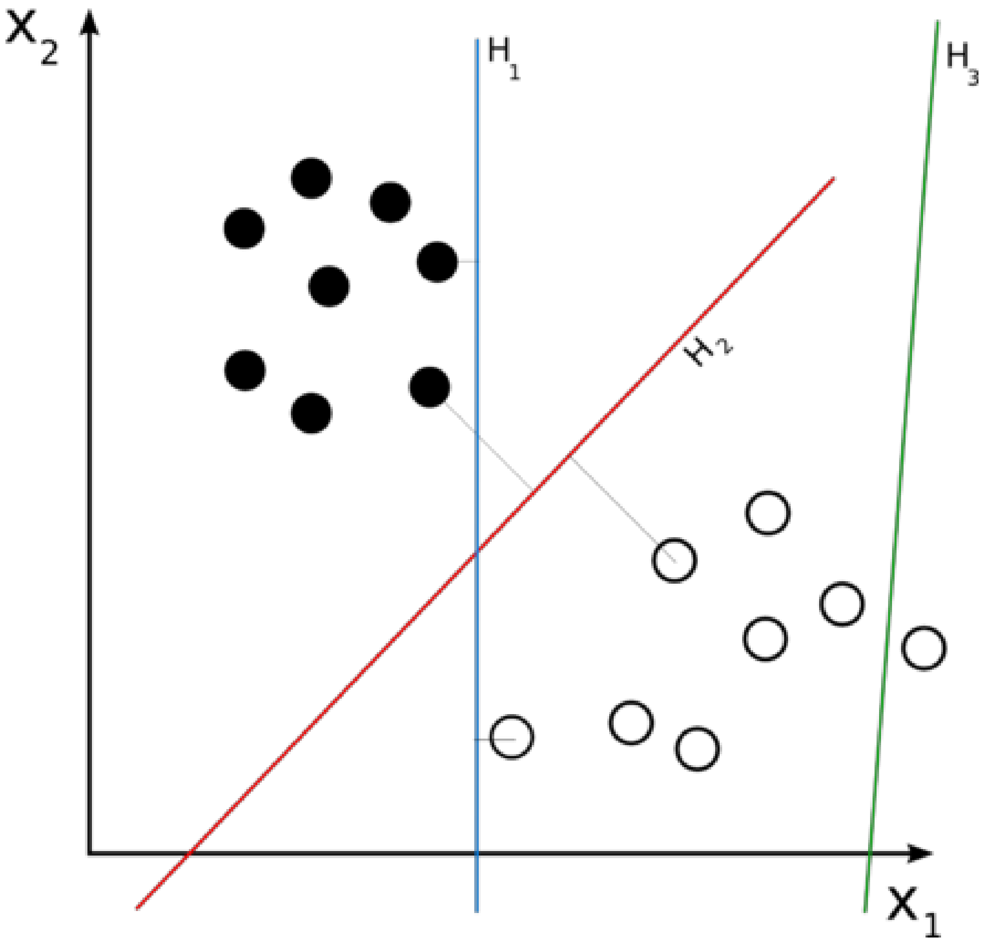


Linear Classifier: An Example

- A toy rule to determine whether a faculty member has tenure
  - Year >= 6 or Title = “Professor” ⇔ Tenure
- How to express the rule as a linear classifier?
  - Features
    - $\mathrm{x}_1 (x_1 \geq 0)$ is an integer denoting the year
    - x_2 is a Boolean denoting whether the title is “Professor”
  - A feasible linear classifier: $f(x) = (x_1 - 5) + 6 \cdot x_2$
    - When x_2 is True, because $x_1 \geq 0$, $f(x)$ is always greater than 0
    - When x_2 is False, because $f(x) > 0 \Leftrightarrow x_1 \geq 6$
  - There are many more feasible classifiers
    - $f(x) = (x_1 - 5.5) + 6 \cdot x_2$
    - $f(x) = 2 \cdot (x_1 - 5) + 11 \cdot x_2$
    - ……

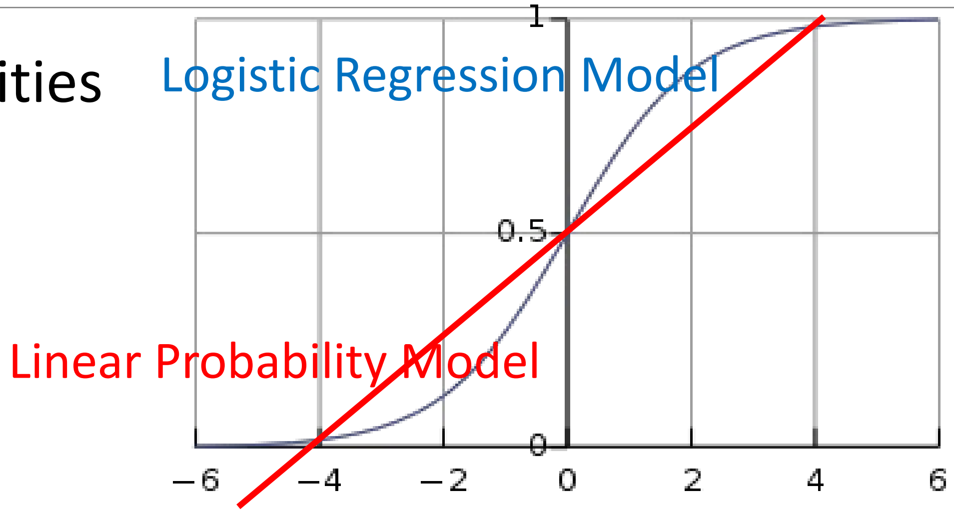
Key Question: Which Line Is Better?

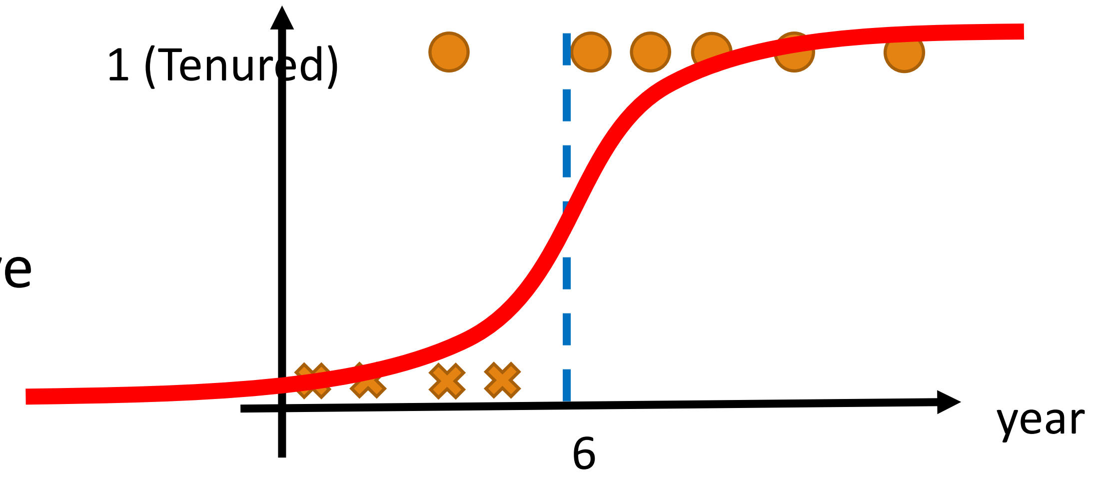
- There might be many feasible linear functions
  - Both H_1 and H_2 will work
- Key question: Which one is better?
  - H_2 looks "better" in the sense that it is also furthest from both groups
  - We will introduce more in the SVM section



Logistic Regression: General Ideas

- Key Idea: Turns linear predictions into probabilities
  - Sigmoid function:
    - $S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$
    - Projects $(-\infty, +\infty)$ to $[0, 1]$
  - Compare to linear probability model
    - Smoother transition
- Logistic regression on our example
  - Only consider year as feature, we have





Logistic Regression: Maximum Likelihood

- The prediction function to learn
  - $p(Y = 1\,|X = x; \boldsymbol{w}) \;=\; S(w_0 + \sum_{i=1}^{n} w_i \cdot x_i)$
  - $\boldsymbol{w} = (w_0, w_1, w_2, \ldots, w_n)$ are the parameters
- Maximum Likelihood
  - Log likelihood:

$$l(w) = \sum_{i=1}^{N} y_i \log p(Y = 1|X = x_i; \boldsymbol{w}) + (1 - y_i) \log\big(1 - p(Y = 1|X = x_i; \boldsymbol{w})\big)$$

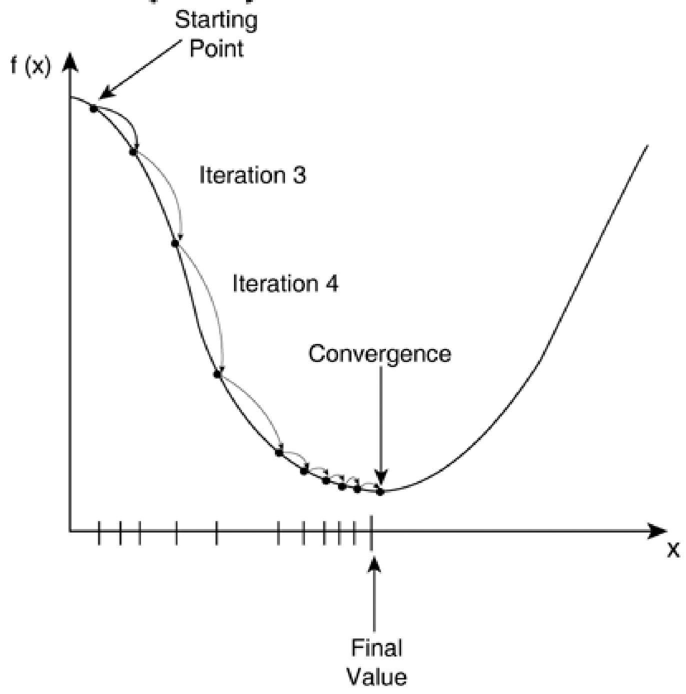
- How to solve it effectively?
  - Gradient Descent
  - Update *w* based on training data
    - Chain-rule for the gradient

Gradient Descent

- Gradient Descent is an iterative optimization algorithm for finding the minimum of a function (e.g., the negative log likelihood)
- For a function F(x) at a point **a**, F(x) *decreases fastest* if we go in the direction of the negative gradient of **a**

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla F(\mathbf{a}_n)$$

When the gradient is zero, we arrive at the local minimum



(McDonald, 2017)

Generative vs. Discriminative Classifiers

- X: Observed variables (features)   Y: Target variables (class labels)
- A generative classifier models p(Y, X)
  - It models how the data was "generated,“ and what is the likelihood this or that class generated this instance, and pick the one with higher probability
  - Naïve Bayes
  - Bayesian Networks
- A discriminative classifier models p(Y|X)
  - It uses the data to create a decision boundary
  - Logistic Regression
  - Support Vector Machines

Further Comments on Discriminative Classifiers

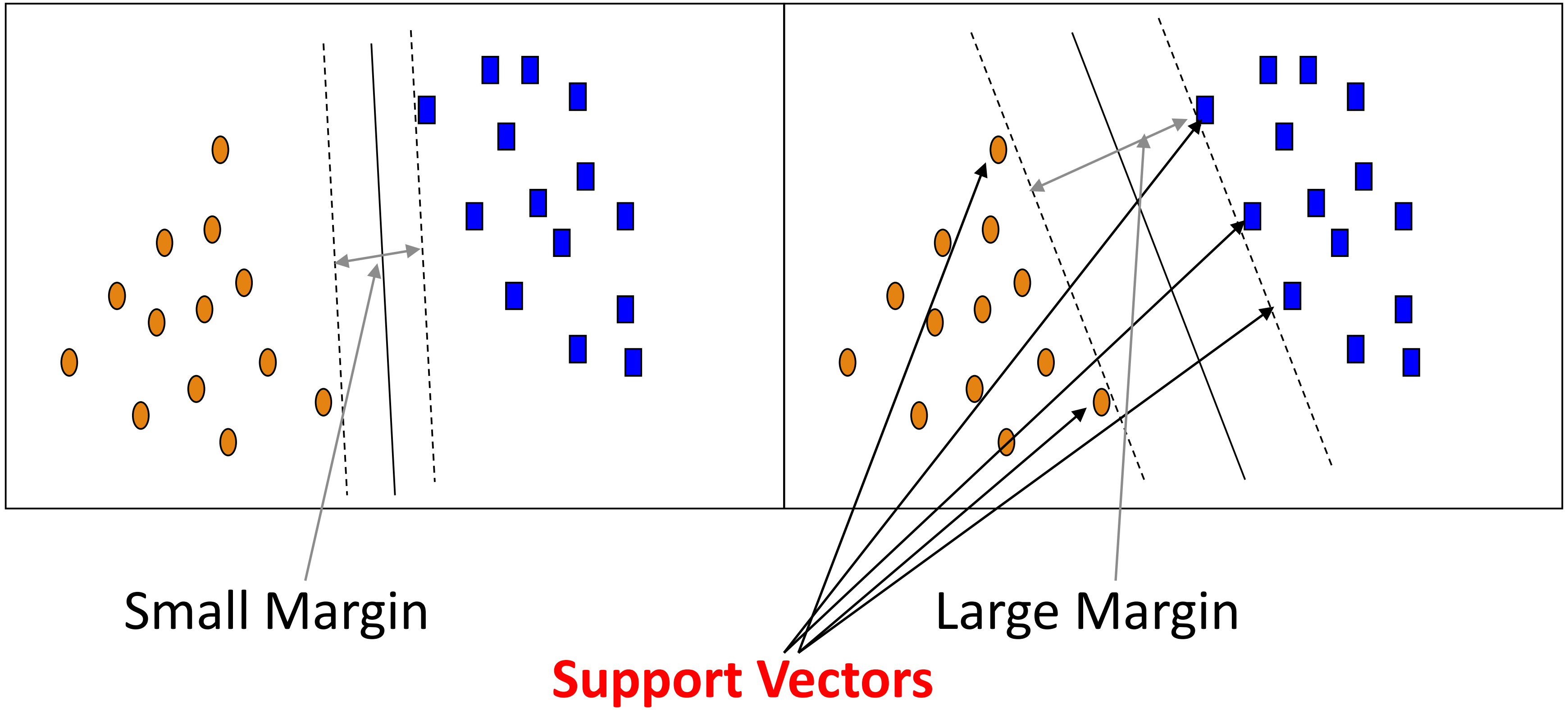
- Strength
  - Prediction accuracy is generally high
    - As compared to generative models
    - Robust, works when training examples contain errors
  - Fast evaluation of the learned target function
    - Comparing to Bayesian networks (which are normally slow)
- Criticism
  - Long training time
  - Difficult to understand the learned function (weights)
    - Bayesian networks can be used easily for pattern discovery
  - Not easy to incorporate domain knowledge
    - Easy in the form of priors on the data or distributions

Support Vector Machines

SVM - Support Vector Machines

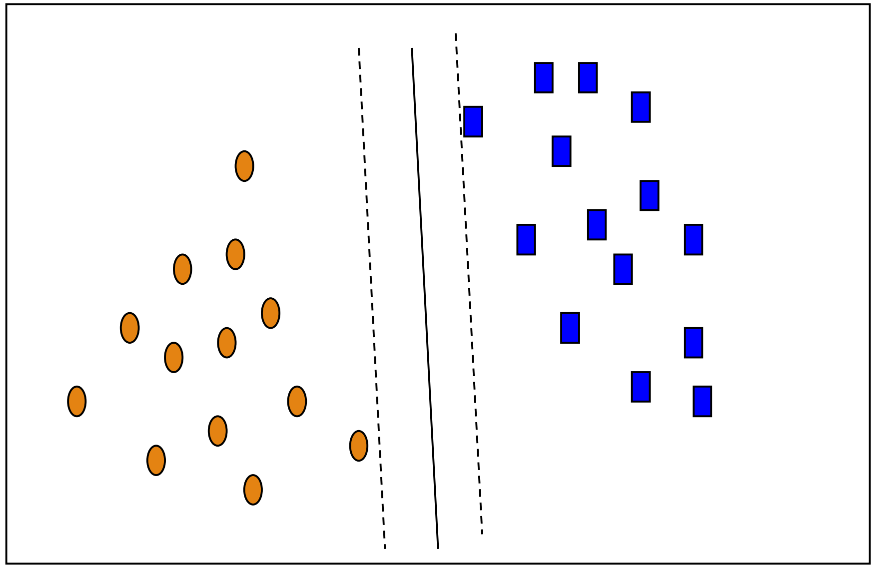
- A popularly adopted classification method for both linear and nonlinear data
  - Proposed by Vapnik and colleagues (1992) - groundwork from Vapnik & Chervonenkis' statistical learning theory in 1960s
- For nonlinear classifier, it uses a nonlinear mapping to transform the original training data into a higher dimension space
- With the new dimension, it searches for the linear optimal separating **hyperplane** (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using **support vectors** ("essential" training instances) and **margins** (defined by the support vectors)

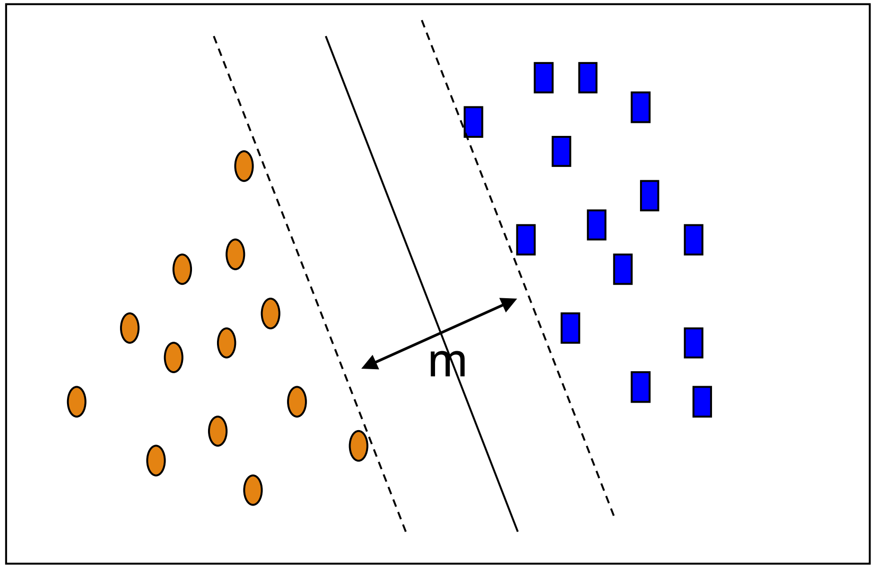
SVM - General Philosophy



- The best hyperplane is the one that represents the largest separation, or **margin**, between the two classes
- Samples on the margin are called the **support vectors**

SVM - When Data Is Linearly Separable





Let data D be ($\mathbf{X}_1$, y_1), …, ($\mathbf{X}_{|D|}$, $y_{|D|}$), where $\mathbf{X}_i$ is the set of training instances associated with the class labels y_i

There are infinite lines (<u>hyperplanes</u>) separating the two classes but we want to <u>find the best one</u> (the one that minimizes classification error on unseen data)

SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)

SVM - Linearly Separable

- A separating hyperplane can be written as

  W ● **X** + b = 0

  where **W**={w_1, w_2, …, w_n} is a weight vector and b a scalar (bias)
- For 2-D, it can be written as: $w_0 + w_1 x_1 + w_2 x_2 = 0$
  - The hyperplane defining the sides of the margin:

    $H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

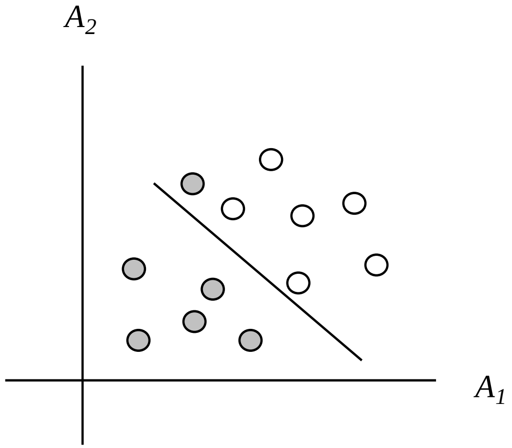
    $H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**
- This becomes a **constrained (convex) quadratic optimization** problem:
  - Quadratic objective function and linear constraints → *Quadratic Programming (QP)* → Lagrangian multipliers

SVM - Linearly Inseparable

- Transform the original input data into a higher dimensional space

Example 6.8 Nonlinear transformation of original input data into a higher dimensional space. Consider the following example. A 3D input vector $\mathbf{X} = (x_1, x_2, x_3)$ is mapped into a 6D space Z using the mappings $\phi_1(X) = x_1, \phi_2(X) = x_2, \phi_3(X) = x_3$, $\phi_4(X) = (x_1)^2, \phi_5(X) = x_1x_2$, and $\phi_6(X) = x_1x_3$. A decision hyperplane in the new space is $d(\mathbf{Z}) = \mathbf{W}\mathbf{Z} + b$, where $\mathbf{W}$ and $\mathbf{Z}$ are vectors. This is linear. We solve for $\mathbf{W}$ and b and then substitute back so that we see that the linear decision hyperplane in the new ($\mathbf{Z}$) space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$\begin{aligned} d(Z) &= w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b \\ &= w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + b \end{aligned} \quad \blacksquare$$



- Search for a linear separating hyperplane in the new space

Kernel Functions for Nonlinear Classification

- SVMs can efficiently perform a non-linear classification using kernel functions, implicitly mapping their inputs into high-dimensional feature spaces
- Instead of computing the dot product on the transformed data, it is mathematically equivalent to applying a kernel function $K(\mathbf{X}_i, \mathbf{X}_j)$ to the original data, i.e.,
  - $K(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i)\ \Phi(\mathbf{X}_j)$
- Typical Kernel Functions

$$\text{Polynomial kernel of degree } h: \quad K(X_i, X_j) = (X_i \cdot X_j + 1)^h$$

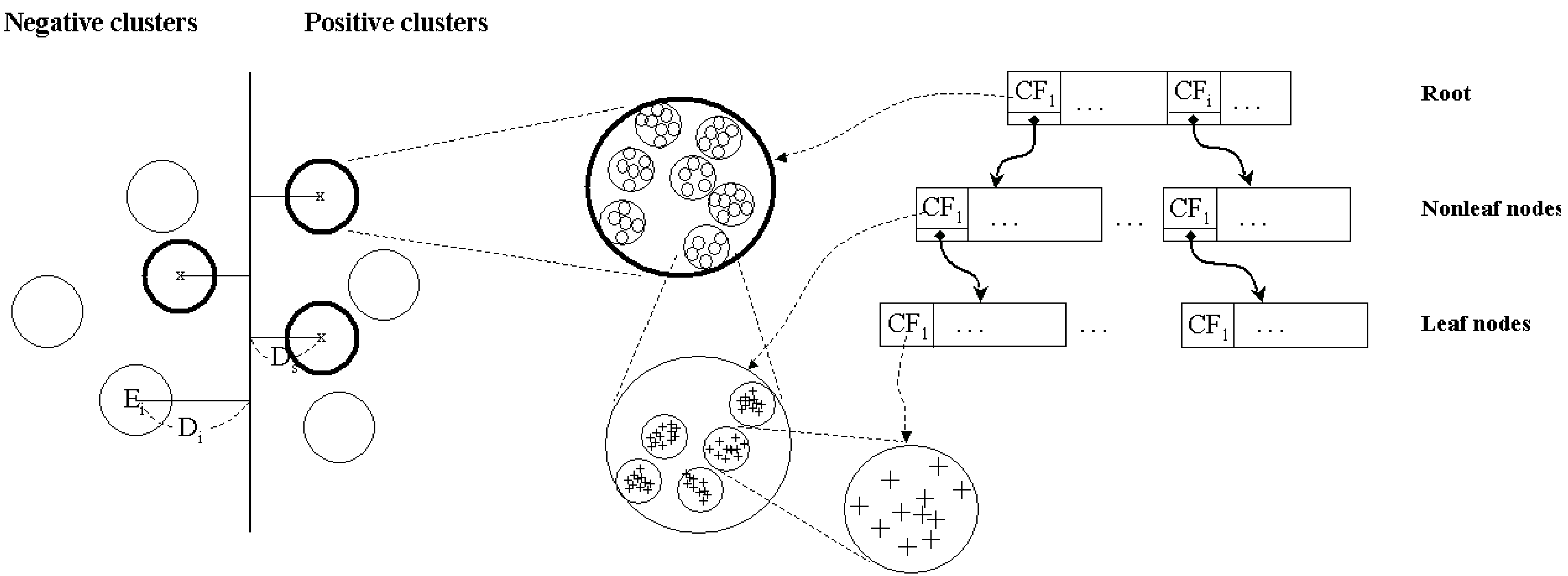
$$\text{Gaussian radial basis function kernel}: \quad K(X_i, X_j) = e^{-\|X_i - X_j\|^2/2\sigma^2}$$

$$\text{Sigmoid kernel}: \quad K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)$$

Is SVM Scalable on Massive Data?

- SVM is effective on high dimensional data
  - The **complexity** of trained classifier is characterized by the <u>number of support vectors</u> rather than the dimensionality of the data
  - The **support vectors** are the <u>essential or critical training examples</u> - they lie closest to the decision boundary (MMH)
  - Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high
- SVM is not scalable to the number of data objects in terms of training time and memory usage
  - Scaling SVM by a hierarchical micro-clustering approach
    - Classifying Large Data Sets Using SVM with Hierarchical Clusters (Yu, Yang, & Han, 2003)

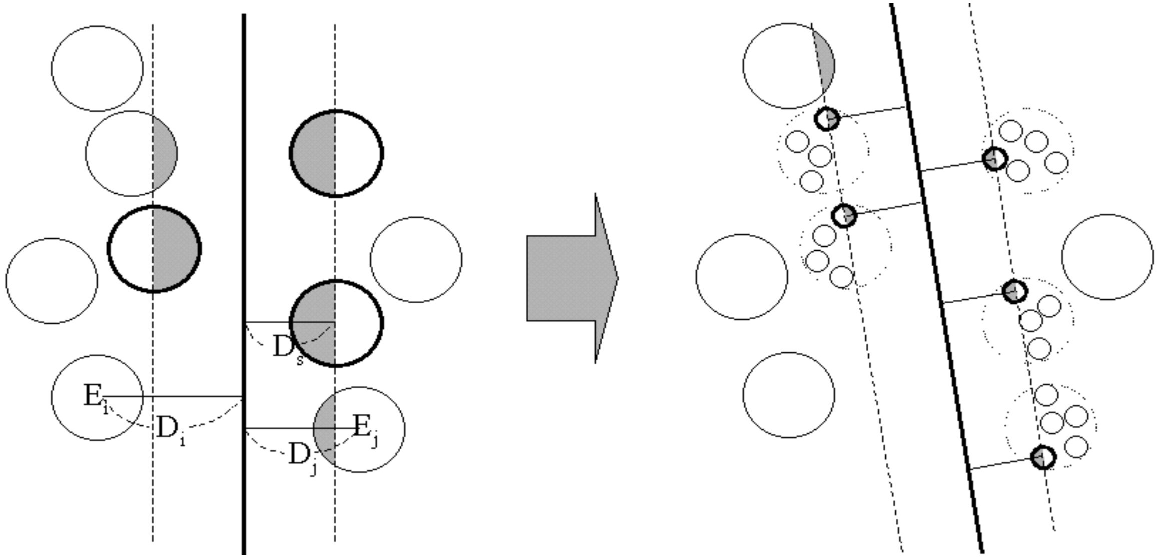
Scaling SVM by Hierarchical Micro-Clustering



- Construct two CF-trees (i.e., statistical summary of the data) from positive and negative data sets independently (with one scan of the data set)
- Micro-clustering: Hierarchical indexing structure
  - Provide finer samples closer to the boundary and coarser samples farther from the boundary

Selective Declustering: Ensure High Accuracy

- Decluster only the cluster whose subclusters have possibilities to be the support cluster of the boundary
  - "Support cluster": The cluster whose centroid is a support vector
- De-cluster only the cluster E_i such that
  - $D_i - R_i < D_s$, where D_i is the distance from the boundary to the center point of E_i and R_i is the radius of E_i



Accuracy and Scalability on Synthetic Dataset

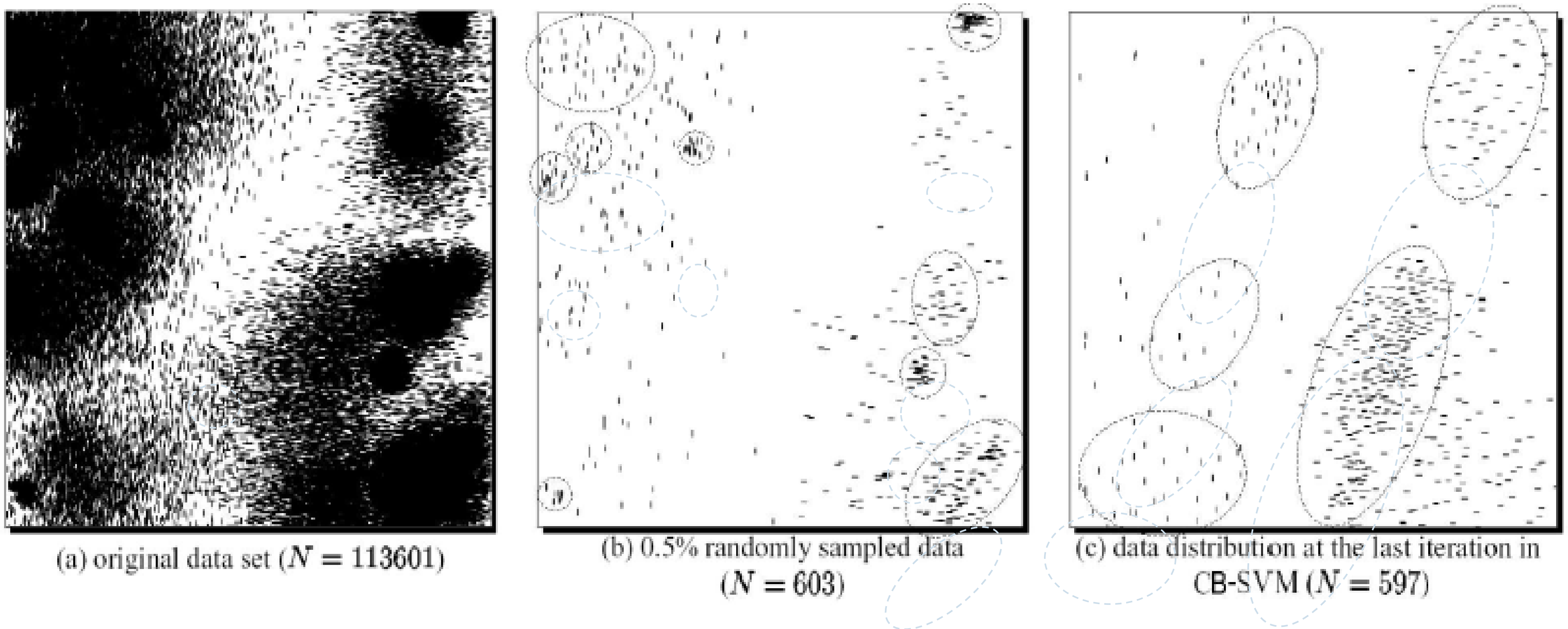


Figure 6: Synthetic data set in a two-dimensional space. '|': positive data; '—': negative data

- Experiments on large synthetic data sets show better accuracy than random sampling approaches and far more scalable than the original SVM algorithm

11

SVM: Features and Applications

- Features and challenges
  - Training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
  - However, the parameters of a solved model are difficult to interpret
- SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional parameters)
- SVM has been used in many applications
  - Handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests
  - Document classification, biomedical data analysis

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

Summary

Recommended Readings

- Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician, 46*(3), 175-185. Retrieved from https://www.jstor.org/stable/2685209?seq=1#page_scan_tab_contents
- Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273-297. Retrieved from https://link.springer.com/article/10.1023/A:1022627411411
- Hosmer Jr., D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression.* Indianapolis, IN: John Wiley.
- Kleinbaum, D. G. & Klein, M. (2010). *Logistic regression: A self-learning text.* New York, NY: Springer.
- Ng, A. Y. & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naïve Bayes. *NIPS.* Retrieved from https://papers.nips.cc/paper/2020-on-discriminative-vs-generative-classifiers-a-comparison-of-logistic-regression-and-naive-bayes
- Scholkopf, B. & Smola, A.J. (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond.* London, UK: The MIT Press.
- Tong, S. & Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research, 2,* 45-66. Retrieved from https://dl.acm.org/citation.cfm?id=944793
- Yu, H., Yang, J., & Han, J. (2003). Classifying large data sets using SVM with hierarchical clusters. *KDD*, 306-315. Retrieved from https://dl.acm.org/citation.cfm?doid=956750.956786
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). Recent advances of large-scale linear classification. *Proc. IEEE, 100*(9), 2584-2603. Retrieved from http://ieeexplore.ieee.org/document/6177645/

References

- McDonald, Conor. (2017). *Machine learning fundamentals (I): Cost functions and gradient descent* [Online image]. Retrieved Feb 16, 2018 from https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220
- Onlinestatbook.com. (2013). *Gpa.jpg* [Online image]. Retrieved Feb 16, 2018 from http://onlinestatbook.com/2/regression/graphics/gpa.jpg