Classification and Support Vector Machine

Yiyong Feng and Daniel P. Palomar

The Hong Kong University of Science and Technology (HKUST)

ELEC 5470 - Convex Optimization
Fall 2017-18, HKUST, Hong Kong

Outline of Lecture

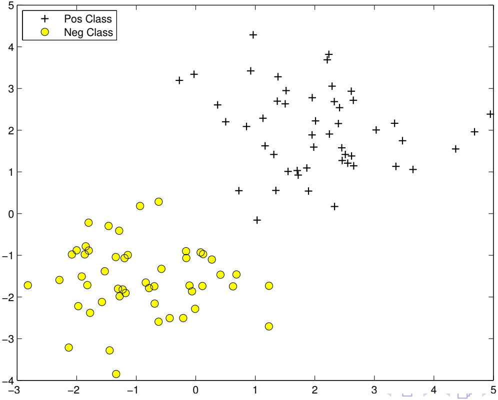
1. Problem Statement
2. Warm Up: Linear/Logistic Regression
   - Linear Regression
   - Regression with Huberized Loss
   - Logistic Regression
3. Support Vector Machine (SVM)
   - Linearly Separable SVM
   - Linearly Nonseparable SVM
   - Nonlinear SVM
   - Multiclass Learning
4. Application: Multiclass Image Classification

Outline of Lecture

1. Problem Statement
2. Warm Up: Linear/Logistic Regression
   - Linear Regression
   - Regression with Huberized Loss
   - Logistic Regression
3. Support Vector Machine (SVM)
   - Linearly Separable SVM
   - Linearly Nonseparable SVM
   - Nonlinear SVM
   - Multiclass Learning
4. Application: Multiclass Image Classification

Classification Problem

- A set of input points with binary labels: $\mathbf{x}_i \in \mathbb{R}^n \to y_i \in \{-1, 1\}$, $i = 1, \ldots, N$
- How to classify the $\mathbf{x}_i$'s?



Classification Problem

- Separate the two classes using a linear model:

$$\hat{y} = \boldsymbol{\beta}^T \mathbf{x} + \beta_0$$

- We can classify as follows:

$$\begin{cases} \text{predict “Pos Class”}, & \text{if } \operatorname{sign}(\hat{y}) = +1 \\ \text{predict “Neg Class”}, & \text{if } \operatorname{sign}(\hat{y}) = -1 \end{cases}$$

- Thus, misclassification happens if $\operatorname{sign}(\hat{y}) \neq y$!
- Note that then the decision boundary is the hyperplane $\boldsymbol{\beta}^T \mathbf{x} + \beta_0 = 0$

Classification Problem

- Then our goal is to minimize the number of misclassifications:

$$\begin{array}{ll} \underset{\beta_0, \boldsymbol{\beta}, \{\hat{y}_i\}}{\text{minimize}} & \sum_{i=1}^{N} \mathbf{1}_{\{\text{sign}(\hat{y}_i) \neq y_i\}} \\ \text{subject to} & \hat{y}_i = \boldsymbol{\beta}^T \mathbf{x}_i + \beta_0, \ \forall i \end{array}$$

- However, the objective loss function is nonconvex and nondifferentiable

- What could we do?

Outline of Lecture

Classification as Linear Regression

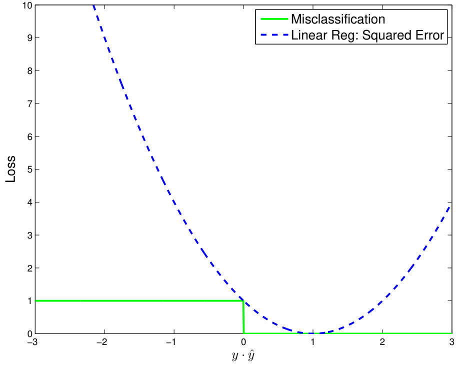
- A straightforward idea is to do linear regression, i.e., replacing the misclassification loss with the residual sum of squares loss:

$$\begin{array}{ll} \underset{\beta_0,\boldsymbol{\beta},\{\hat{y}_i\}}{\text{minimize}} & \sum_{i=1}^{N} |\hat{y}_i - y_i|^2 \\ \text{subject to} & \hat{y}_i = \boldsymbol{\beta}^T \mathbf{x}_i + \beta_0, \ \forall i \end{array}$$

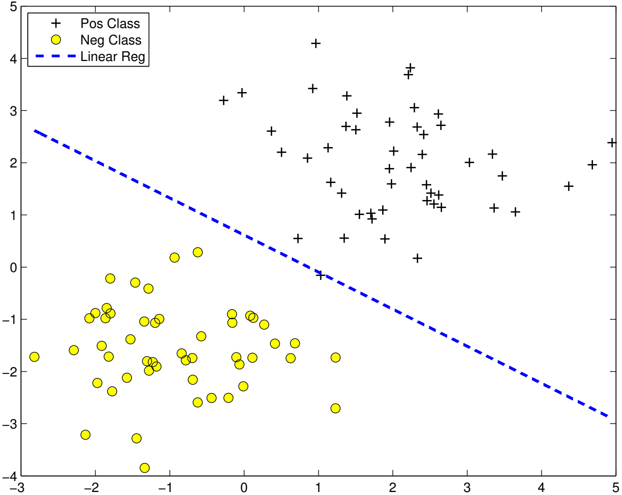
- Note that $|\hat{y}_i - y_i|^2 = |\hat{y}_i - y_i|^2 \cdot y_i^2 = (1 - y_i\hat{y}_i)^2$

From Loss Function Point of View

- Misclassification loss $\mathbf{1}_{\{\text{sign}(\hat{y})\neq y\}}$; squared error loss $(\hat{y}-y)^2 = (1-y_i\hat{y}_i)^2$

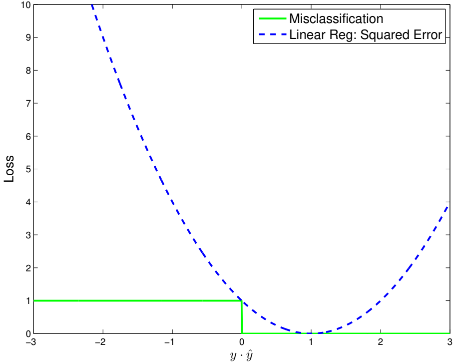


Decision Boundary



- Can we do better?

- Some idea?



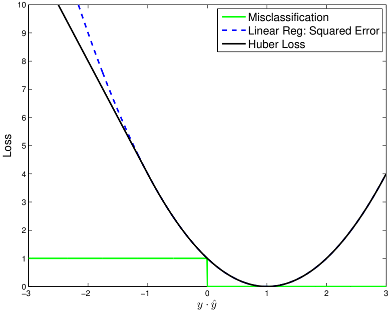
- Can we find some better loss function approximation?

Famous Huber Loss

- Huber Loss (with parameter M):

$$\phi_{hub}(x) = \begin{cases} |x|^2 & |x| < M \\ M(2|x| - M) & |x| \geq M \end{cases}$$

- Select $M = 2$, define loss as $\phi_{hub}(1 - y\hat{y})$:

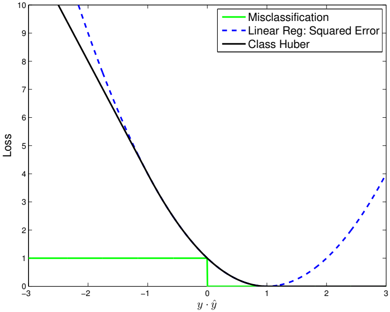


Famous Huber Loss

- Actually, we don't need to penalize when $y\hat{y} > 0$!
- Define

$$\phi_{hub\_pos}(x) = \begin{cases} \phi_{hub}(x) & x \geq 0 \\ 0 & x < 0 \end{cases}$$

- Select $M = 2$, define loss as $\phi_{hub\_pos}(1 - y\hat{y})$:

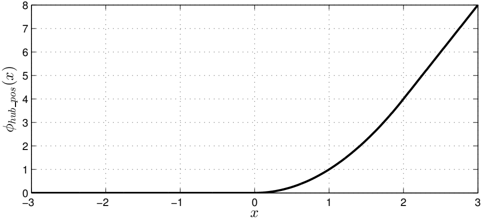


Minimize the "Huberized" Loss

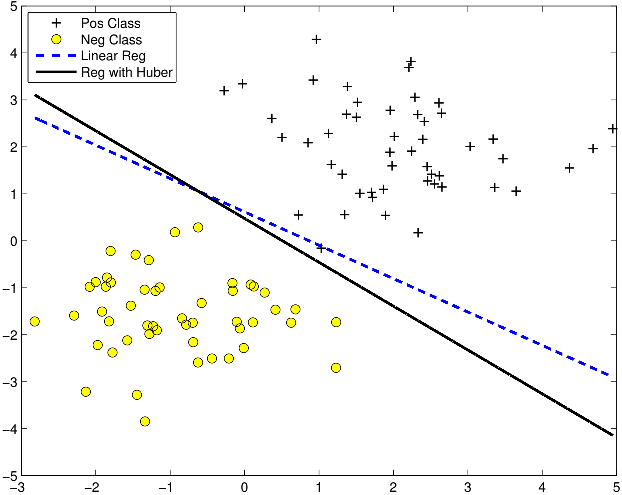
- Then, we take the "Huberized" loss as the approximation and intend to minimize it:

$$\begin{array}{ll} \underset{\beta_0, \boldsymbol{\beta}, \{\hat{y}_i\}}{\text{minimize}} & \sum_{i=1}^{N} \phi_{hub\_pos}\left(1 - y_i \hat{y}_i\right) \\ \text{subject to} & \hat{y}_i = \boldsymbol{\beta}^T \mathbf{x}_i + \beta_0, \ \forall i \end{array}$$

where $\phi_{hub\_pos}(x)$ with $M = 2$ looks like:



Decision Boundary



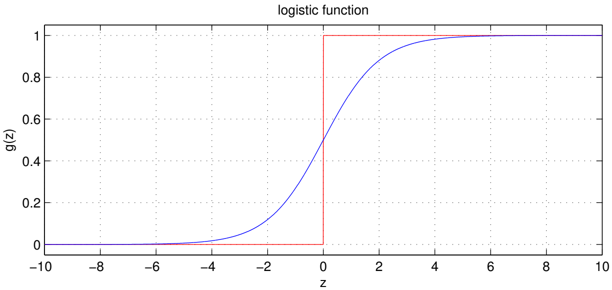
- Can we find more approximations?

Logistic Model

- Given a data point $\mathbf{x} \in \mathbb{R}^n$, model the probability of label $y \in \{-1, +1\}$ as (recall that $\hat{y} = \boldsymbol{\beta}^T \mathbf{x} + \beta_0$):

$$P(Y = y|\mathbf{x}) = \frac{1}{1 + e^{-y \cdot \hat{y}}}$$

- Here, the function $g(z) = \frac{1}{1+e^{-z}}$ is called logistic function:



Logistic Model

- The above probability formula amounts to modeling the log-odds ratio as linear model $\hat{y}$:

$$\log \frac{P\left(Y=1|\mathbf{x}\right)}{P\left(Y=-1|\mathbf{x}\right)} = \log \frac{1+e^{\hat{y}}}{1+e^{-\hat{y}}} = \hat{y} = \boldsymbol{\beta}^T\mathbf{x} + \beta_0$$

- Classification rule:

$$\begin{cases} \text{predict “Pos Class”}, & \text{if } \operatorname{sign}\left(\hat{y}\right) = +1 \\ \text{predict “Neg Class”}, & \text{if } \operatorname{sign}\left(\hat{y}\right) = -1 \end{cases}$$

- The above classification rule is exactly the same as we wanted before!

Negative Log-likelihood as Loss

- Armed with logistic model, we can have the likelihood function:

$$LH\left(\boldsymbol{\beta}, \beta_0\right) = \prod_{i=1}^{N} P\left(Y = y_i | \mathbf{x}_i\right) = \prod_{i=1}^{N} \frac{1}{1 + e^{-y_i \cdot \hat{y}_i}}$$

- The loss function can be defined as negative log-likelihood:

$$Loss\left(\boldsymbol{\beta}, \beta_0\right) = -\log LH\left(\boldsymbol{\beta}, \beta_0\right) = \sum_{i=1}^{N} \log\left(1 + e^{-y_i \cdot \hat{y}_i}\right)$$

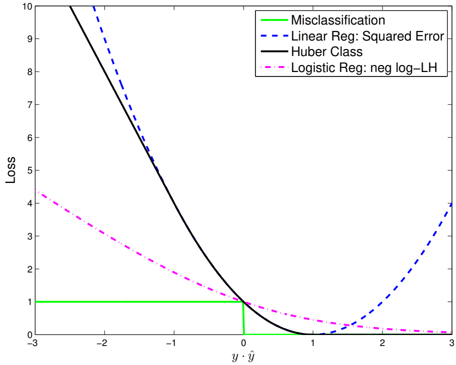
Logistic Regression

- Now, minimize the loss function (i.e., maximize the likelihood):

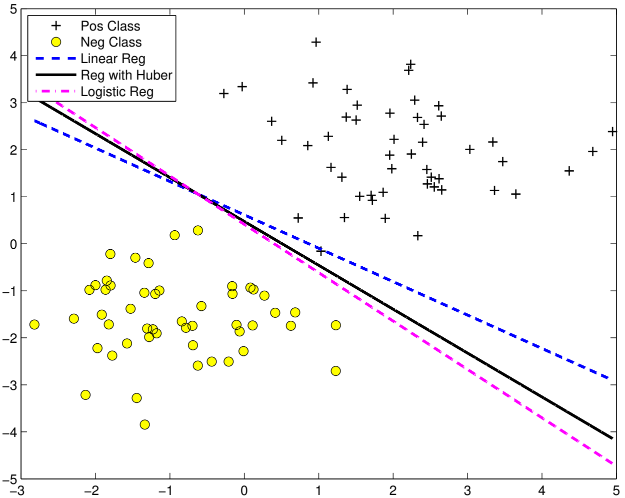
$$\begin{array}{ll}\underset{\beta_0, \boldsymbol{\beta}, \{\hat{y}_i\}}{\text{minimize}} & \sum_{i=1}^{N} \log\left(1 + e^{-y_i \cdot \hat{y}_i}\right) \\ \text{subject to} & \hat{y}_i = \boldsymbol{\beta}^T \mathbf{x}_i + \beta_0, \ \forall i \end{array}$$

From Loss Function Point of View

- Misclassification loss $\mathbf{1}_{\{\text{sign}(\hat{y})\neq y\}}$; squared error loss $(\hat{y}-y)^2$; Class Huber $\phi_{hub\_pos}(1-y\hat{y})$; Negative log-likelihood $\log\left(1+e^{-y\cdot\hat{y}}\right)$



Decision Boundary

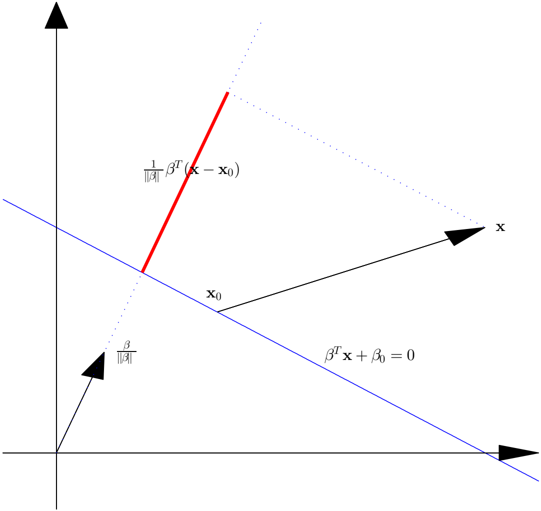


Outline of Lecture

Optimal Separating Hyperplane

- So many hyperplanes can separate the two classes
- BUT, what is the optimal separating hyperplane?
- Optimal separating hyperplane should:
  - separate the two classes, and
  - maximize the distance to the closest point from either class.

Signed Distance



- decision boundary:

$$\boldsymbol{\beta}^T\mathbf{x}+\beta_0=0$$

- signed distance

$$\frac{1}{\|\boldsymbol{\beta}\|}\boldsymbol{\beta}^T(\mathbf{x}-\mathbf{x}_0) = \frac{1}{\|\boldsymbol{\beta}\|}\left(\boldsymbol{\beta}^T\mathbf{x}+\beta_0\right)$$

Maximize the Margin

- Then the "margin" is defined as

$$\frac{1}{\|\boldsymbol{\beta}\|} y \left(\boldsymbol{\beta}^T \mathbf{x} + \beta_0\right)$$

- Now, maximize the distance to the closest point from either class

$$\begin{array}{ll} \underset{\beta_0, \boldsymbol{\beta}, M}{\text{maximize}} & M \\ \text{subject to} & \frac{1}{\|\boldsymbol{\beta}\|} y_i \left(\boldsymbol{\beta}^T \mathbf{x}_i + \beta_0\right) \geq M, \ \forall i \end{array}$$

Support Vector Machine

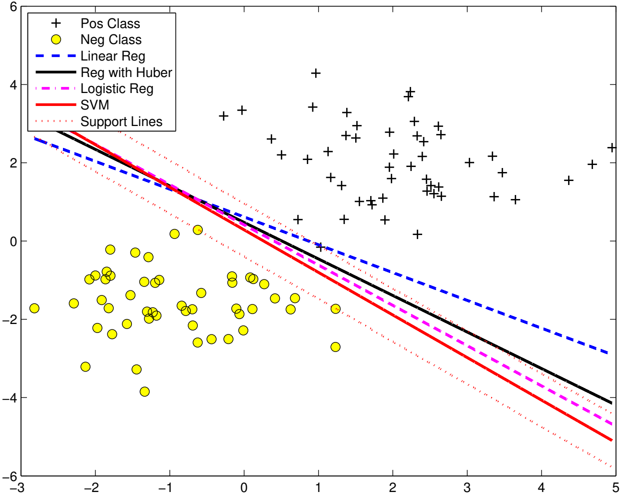
- For any $\boldsymbol{\beta}$ and β_0 satisfying these inequalities, any positively scaled multiple satisfies them too, then arbitrarily set

$$\|\boldsymbol{\beta}\| = \frac{1}{M}$$

- The above problem amounts to

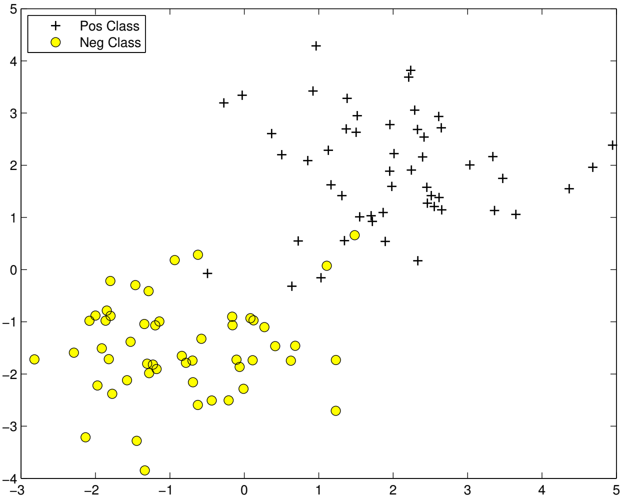
$$\begin{array}{ll} \underset{\beta_0, \boldsymbol{\beta}}{\text{minimize}} & \|\boldsymbol{\beta}\| \\ \text{subject to} & y_i \left(\boldsymbol{\beta}^T \mathbf{x}_i + \beta_0 \right) \geq 1, \ \forall i \end{array}$$

Boundary and Support Vectors



- The case we considered before is linearly separable
- What if it's linearly nonseparable?

Linearly Nonseparable Case



Linearly Nonseparable Case

- Relax the constraints by Introducing positive slack variables ξ_i's:

$$y_i\left(\boldsymbol{\beta}^T\mathbf{x}_i+\beta_0\right)\geq 1-\xi_i$$

- And then penalize $\sum_i \xi_i$ in the objective function:

$$\begin{array}{ll}\underset{\beta_0,\boldsymbol{\beta},\{\xi_i\}}{\text{minimize}} & \frac{1}{2}\left\|\boldsymbol{\beta}\right\|^2+C\sum_{i=1}^{N}\xi_i\\ \text{subject to} & \xi_i\geq 0,\; y_i\left(\boldsymbol{\beta}^T\mathbf{x}_i+\beta_0\right)\geq 1-\xi_i,\;\forall i\end{array}$$

- Linearly separable case corresponds to $C=\infty$

The SVM as a Penalization Method

- Revisit the primal problem:

$$\begin{array}{ll} \underset{\beta_0,\boldsymbol{\beta},\{\xi_i\}}{\text{minimize}} & \frac{1}{2}\left\|\boldsymbol{\beta}\right\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} & \xi_i \geq 0,\ y_i\left(\boldsymbol{\beta}^T\mathbf{x}_i + \beta_0\right) \geq 1-\xi_i,\ \forall i \end{array}$$

- Consider the optimization problem:

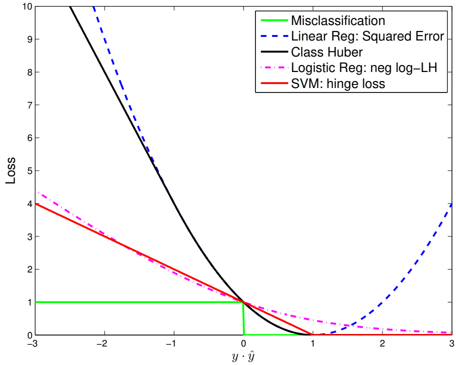
$$\begin{array}{ll} \underset{\beta_0,\boldsymbol{\beta},\{\hat{y}_i\}}{\text{minimize}} & \sum_{i=1}^{N}\left[1-y_i\hat{y}_i\right]^+ + \frac{\lambda}{2}\left\|\boldsymbol{\beta}\right\|^2 \\ \text{subject to} & \hat{y}_i = \boldsymbol{\beta}^T\mathbf{x}_i + \beta_0,\ \forall i \end{array}$$

  where loss is $\sum_{i=1}^{N}\left[1-y_i\hat{y}_i\right]^+$ (called hinge loss)

- Two problems are equivalent with $\lambda = 1/C$ (linearly separable case corresponds to $\lambda = 0$)

From Loss Function Point of View

- Misclassification loss $\mathbf{1}_{\{\text{sign}(\hat{y})\neq y\}}$; squared error loss $(\hat{y}-y)^2$; Class Huber $\phi_{hub\_pos}(1-y\hat{y})$; Negative log-likelihood loss $\log\left(1+e^{-y\cdot\hat{y}}\right)$; Hinge loss $[1-y_i\hat{y}_i]^+$



Solving SVM I

- The Lagrange function is

$$L(\boldsymbol{\beta}, \beta_0, \xi_i, \alpha_i, \mu_i) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\mu_i\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i\left(\boldsymbol{\beta}^T\mathbf{x}_i + \beta_0\right) - (1-\xi_i)\right]$$

where $\alpha_i \geq 0$ and $\mu_i \geq 0$, $\forall i$, are dual variables

- Setting derivatives w.r.t. $\boldsymbol{\beta}, \beta_0, \{\xi_i\}$ to zero

$$\boldsymbol{\beta} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i, \quad 0 = \sum_{i=1}^{N}\alpha_i y_i, \quad \alpha_i = C - \mu_i \quad \forall i$$

Solving SVM II

- Dual function:

$$
\begin{aligned}
g\left(\alpha_i, \mu_i\right) = & \inf_{\boldsymbol{\beta}, \beta_0, \{\xi_i\}} L\left(\boldsymbol{\beta}, \beta_0, \xi_i, \alpha_i, \mu_i\right) \\
= & \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \left(\alpha_i + \mu_i\right) \xi_i \\
& - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \beta_0 \sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i \\
= & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j
\end{aligned}
$$

Solving SVM III

- Dual problem:

$$
\begin{array}{ll}
\underset{\{\alpha_i\}}{\text{maximize}} & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\
\text{subject to} & 0 \leq \alpha_i \leq C, \ \forall i \\
& \sum_{i=1}^{N} \alpha_i y_i = 0.
\end{array}
$$

- Dual problem is a QP!
- Vector-matrix representation of the objective:

$$
\mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^T \left(\text{Diag}\left(\mathbf{y}\right) \mathbf{X}^T \mathbf{X} \text{Diag}\left(\mathbf{y}\right) \right) \boldsymbol{\alpha}
$$

KKT Conditions

- KKT equations characterize the solution to the primal and dual problems

$$\xi_i \geq 0, \; y_i \left(\boldsymbol{\beta}^T \mathbf{x}_i + \beta_0 \right) \geq 1 - \xi_i, \; \forall i$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i, \; 0 \leq \alpha_i \leq C, \; \forall i$$

$$\boldsymbol{\beta} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i, \; \alpha_i = C - \mu_i, \; \forall i$$

$$0 = \mu_i \xi_i, \; \alpha_i \left[y_i \left(\boldsymbol{\beta}^T \mathbf{x}_i + \beta_0 \right) - (1 - \xi_i) \right] = 0, \; \forall i$$

Finding Decision Boundary

- Given optimal dual $\alpha_i^\star$'s, we have:

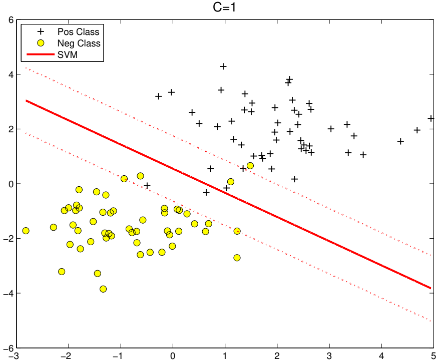
$$\boldsymbol{\beta}^\star = \sum_{i=1}^{N} \alpha_i^\star y_i \mathbf{x}_i.$$

- Support Vectors: those observations i with nonzero $\alpha_i^\star$
- For any $0 < \alpha_j^\star < C$, we have $\xi_j = 0$, and $y_j \left(\sum_{i=1}^{N} \alpha_i^\star y_i \mathbf{x}_i^T \mathbf{x}_j + \beta_0 \right) = 1$, hence we can solve for $\beta_0^\star$
- Decision function becomes:

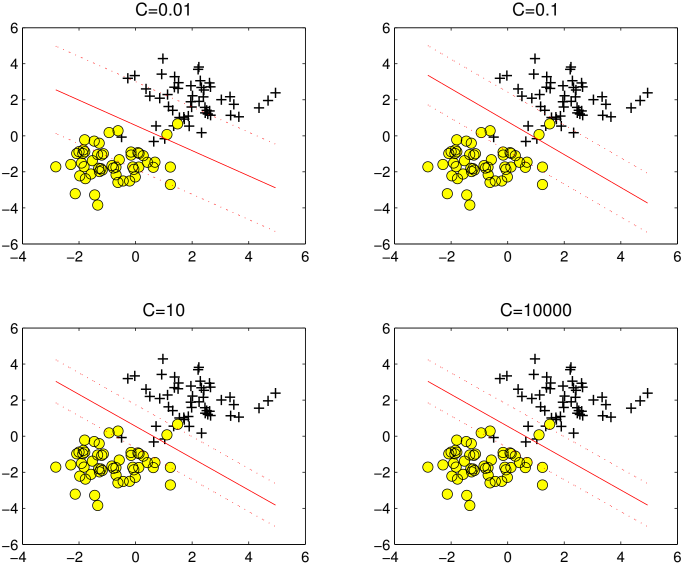
$$D(\mathbf{x}) = \text{sign}(\hat{y}) = \text{sign}\left(\boldsymbol{\beta}^{\star T} \mathbf{x} + \beta_0^\star \right) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i^\star y_i \mathbf{x}_i^T \mathbf{x} + \beta_0^\star \right)$$

- Observation: in the dual problem and the above decision function, the only operation on $\mathbf{x}_i$'s is the inner product!

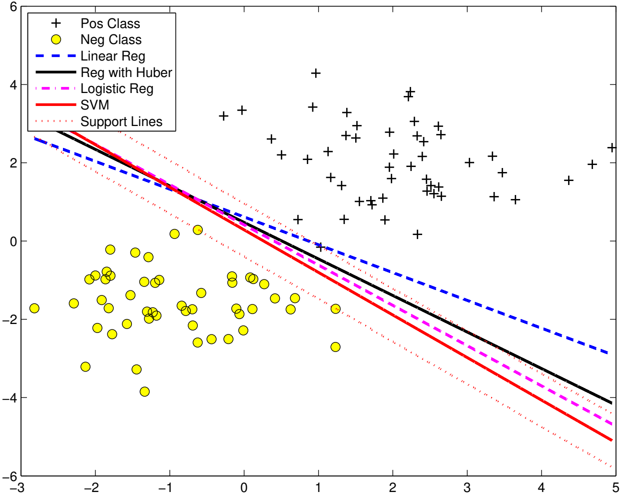
Decision Boundary



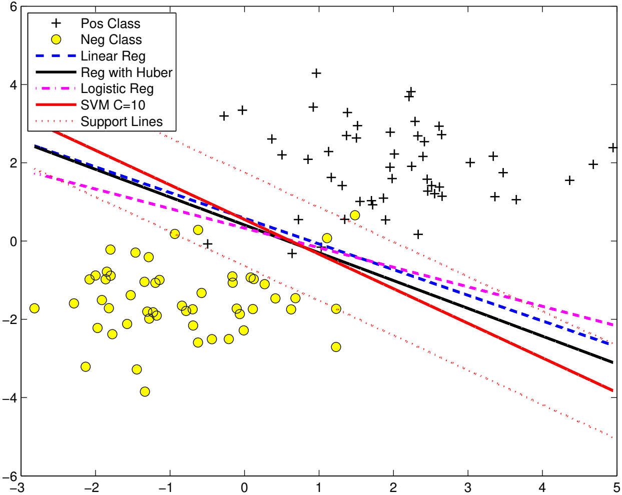
- Large C: focus attention more on points near the boundary ⇒ small margin
- Small C: involves data further away ⇒ large margin


C=0.01
C=0.1
C=10
C=10000
6
4
2
0
−2
−4
−6
−4
−2
0
2
4
6

Decision Boundary: Revisit Linearly Separable Case

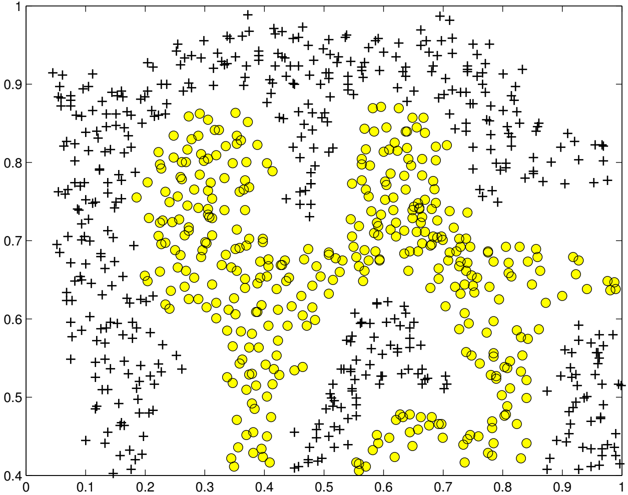


Decision Boundary: Linearly Nonseparable Case



- For the linearly separable/nonseparable cases, so far SVM works well!
- But, what if the linear decision boundary doesn't work any more?

Nonlinear Decision Surface



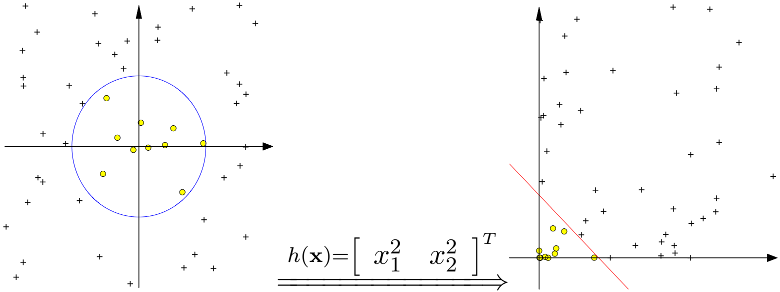
Feature Transformation

Naive method:

- use a curve instead of a line $\Rightarrow$ not efficient

Feature Transformation:

- pre-process the data with $h : \mathbb{R}^n \mapsto \mathcal{H}$, $\mathbf{x} \mapsto h(\mathbf{x})$
- $\mathbb{R}^n$: input space; $\mathcal{H}$: feature space



After Feature Transformation

- Using the features $h(\mathbf{x})$ as inputs:

$$\hat{y} = \boldsymbol{\beta}^T h(\mathbf{x}) + \beta_0$$

- Dual problem:

$$\begin{array}{ll} \underset{\{\alpha_i\}}{\text{maximize}} & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j h(\mathbf{x}_i)^T h(\mathbf{x}_j) \\ \text{subject to} & \sum_{i=1}^N \alpha_i y_i = 0, \ 0 \leq \alpha_i \leq C, \ \forall i \end{array}$$

- The decision function:

$$D(\mathbf{x}) = \text{sign}\left(\boldsymbol{\beta}^T h(\mathbf{x}) + \beta_0\right) = \text{sign}\left(\sum_{i=1}^N \alpha_i y_i h(\mathbf{x}_i)^T h(\mathbf{x}) + \beta_0\right)$$

where β_0 can be determined, as before, by solving $y_i \hat{y}_i = 1$ for any $\mathbf{x}_i$ for which $0 < \alpha_i < C$

Kernelized SVM

- In fact, we need not specify the transformation $h(\mathbf{x})$ at all, but require only knowledge of the kernel function that computes inner products in the transformed space, i.e.:

$$K(\mathbf{x}_i, \mathbf{x}_j) \triangleq h(\mathbf{x}_i)^T h(\mathbf{x}_j)$$

- Dual problem:

$$\begin{array}{ll} \underset{\{\alpha_i\}}{\text{maximize}} & \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} & \sum_{i=1}^N \alpha_i y_i = 0,\ 0 \leq \alpha_i \leq C,\ \forall i \end{array}$$

- The decision function:

$$D(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + \beta_0\right)$$

where β_0 can be determined, as before, by solving $y_i \hat{y}_i = 1$ for any $\mathbf{x}_i$ for which $0 < \alpha_i < C$

Kernel

- K should be a symmetric positive (semi-) definite function
- The previous linearly SVM corresponds to $K(\mathbf{x}_i, \mathbf{x}_j) \triangleq \mathbf{x}_i^T \mathbf{x}_j$
- Popular kernels:

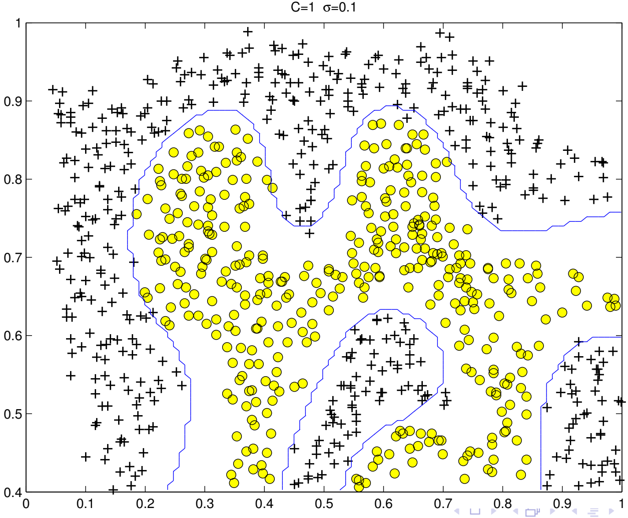
$$
\begin{aligned}
p\text{th Degree polynomial:} \quad & K(\mathbf{x}_i, \mathbf{x}_j) = \left(1 + \mathbf{x}_i^T \mathbf{x}_j\right)^p \\
\text{Gaussian radial kernel:} \quad & K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / \sigma^2} \\
\text{Neural network:} \quad & K(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\kappa_1 \mathbf{x}_i^T \mathbf{x}_j + \kappa_2\right)
\end{aligned}
$$

Steps for Applying SVM

Steps:

1. In-sample training:
    1. Select the parameter C
    2. Select the kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ and related parameters
    3. Solve the dual problem to obtain $\alpha_i^\star$
    4. Compute $\beta_0^\star$, and then classify according to $\text{sign}\left(\sum_{i=1}^{N} \alpha_i^\star y_i K(\mathbf{x}_i, \mathbf{x}) + \beta_0^\star\right)$
2. Out-of-sample testing:
    1. Use the trained model to test out-of-samples
    2. If the out-of-sample test is not good, adjust parameter C, or kernels, and re-train the model until the out-of-sample result is good enough

Decision Boundary by Gaussian Radial Kernel

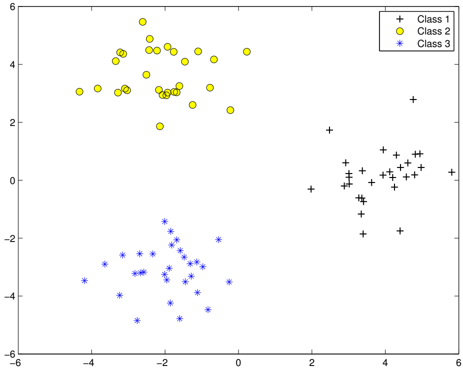


More than two classes?

- What if there are more than TWO classes?

Multiclass Classification Setup

- Labels: $\{-1, +1\} \rightarrow \{1, 2, \ldots, K\}$
- Classification decision rule: $f : \mathbf{x} \in \mathbb{R}^n \mapsto \{1, 2, \ldots, K\}$



Multiclass Classification Methods

- Main ideas:
  - Decompose the multiclass classification problem into multiple binary classification problems
  - Use the majority voting principle or a combined decision from a committee to predict the label
- Common approaches:
  - One-vs-Rest (One-vs-All) approach
  - One-vs-One (Pairwise, All-vs-All) approach

One-vs-Rest Approach

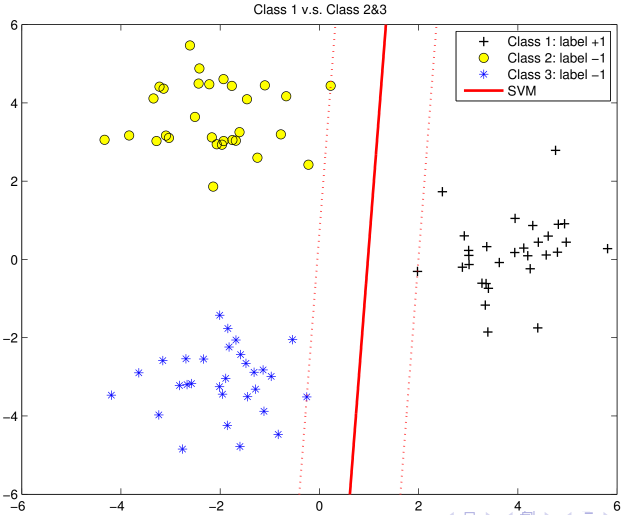
Steps:

1. Solve K different binary problems: classify class i as $+1$ versus the rest classes for all $j \in \{1, \ldots, K\} \backslash i$ as -1
2. Assign a test sample to the class $\arg\max_i f_i(\mathbf{x})$, where $f_i(\mathbf{x})$ is the solution from the i-th problem

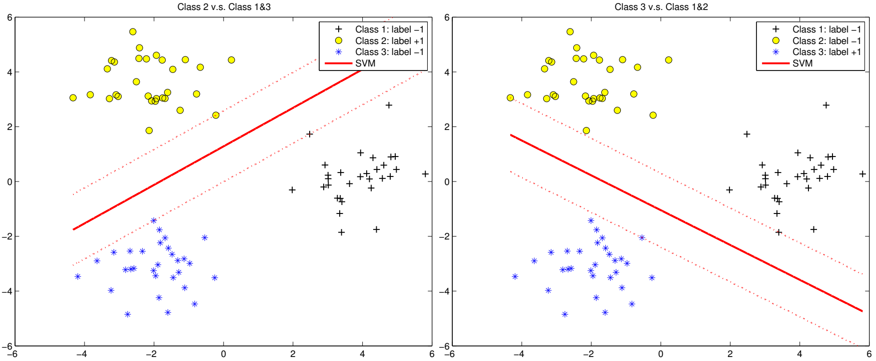
Properties:

- Simple to implement, performs well in practice
- Not optimal (asymptotically)

One-vs-Rest Example: Step 1, training

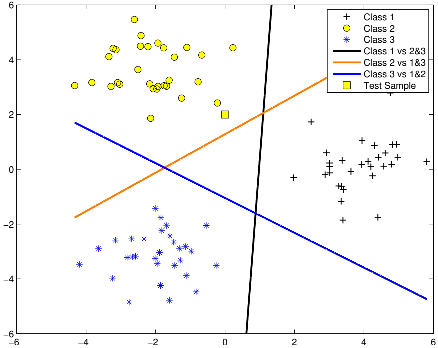


One-vs-Rest Example: Step 1, training...



One-vs-Rest Example: Step 2, test

- Test point $\mathbf{x} = \begin{bmatrix} 0 & 2 \end{bmatrix}^T$, by $f_i(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\beta}^i + \beta_0^i$, we have $f_1(\mathbf{x}) = -1.0783$, $f_2(\mathbf{x}) = 0.5545$, and $f_3(\mathbf{x}) = -2.2560$
- Assign $\begin{bmatrix} 0 & 2 \end{bmatrix}^T$ to class 2!



One-vs-One Approach

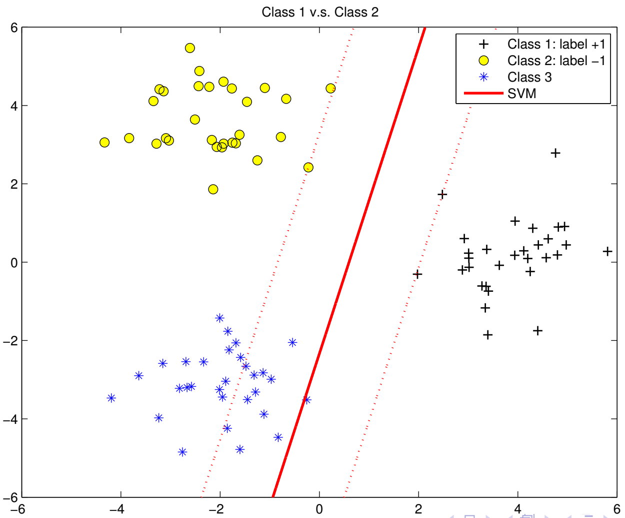
Steps:

1. Solve $\binom{K}{2}$ different binary problems: classify class i as $+1$ versus each class $j \neq i$ as -1. Each classifier is called f_{ij}
2. For prediction at a point, each classifier is queried once and issues a vote. The class with the maximum number of votes is the winner

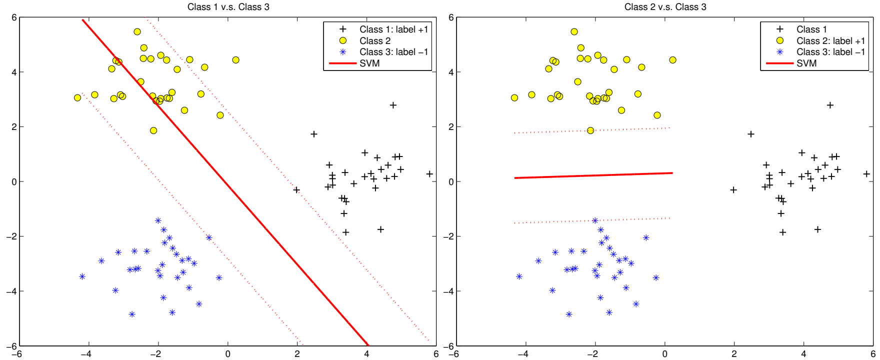
Properties:

- Training process is efficient: small binary problems
- There are too many problems when K is large (If $K = 10$, we need to train 45 binary classifiers)
- Simple to implement, performs competitively in practice

One-vs-One Example: Step 1, training

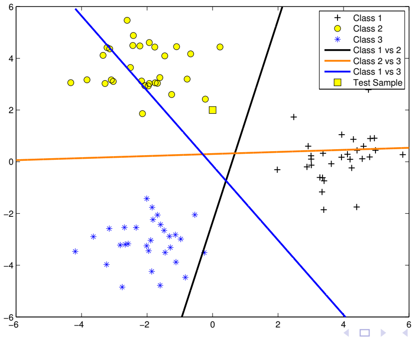


One-vs-One Example: Step 1, training...



One-vs-One Example: Step 2, test

- The same test point $\mathbf{x} = \begin{bmatrix} 0 & 2 \end{bmatrix}^T$:
  - $f_{12}(\mathbf{x}) = -0.7710 < 0$, vote to class 2
  - $f_{23}(\mathbf{x}) = 1.0336 > 0$, vote to class 2
  - $f_{13}(\mathbf{x}) = 0.7957 > 0$, vote to class 1
- Conclusion: class 2 wins!



Outline of Lecture

Application: Histogram-based Image Classification

- Multiclass: air shows, bears, Arabian horses, night scenes, and several more classes not shown here

App: Histogram-based Image Multi-Classification, Modeling

Why not put image pixels into a vector? Drawbacks:

- large size
- lack of invariance with respect to translations

Histogram-based Image representation

- color space: Hue-Saturation-Value (HSV)
- $\#$ of bins per color component $= 16$
- $\mathbf{x} \in \mathbb{R}^{4096}$: histogram of the picture, dimension $= 16^3 = 4096$
- $y \in \{airshows, \ bears, \dots\}$: the class labels

App: Histogram-based Image Multi-Classification, SVM

Applied SVMs:

- linear SVM
- Poly 2: $K(\mathbf{x}_i, \mathbf{x}_j) = \left(1 + \mathbf{x}_i^T \mathbf{x}_j\right)^2$
- Radial basis function

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\rho d(\mathbf{x}_i, \mathbf{x}_j)}$$

where the distance measure $d(\mathbf{x}_i, \mathbf{x}_j)$ can be

- Gaussian: $d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|^2$
- Laplacian (ℓ_1 distance): $d(\mathbf{x}_i, \mathbf{x}_j) = |\mathbf{x}_i - \mathbf{x}_j|$
- χ^2: $d(\mathbf{x}_i, \mathbf{x}_j) = \sum_k \frac{\left(\mathbf{x}_i^{(k)} - \mathbf{x}_j^{(k)}\right)^2}{\mathbf{x}_i^{(k)} + \mathbf{x}_j^{(k)}}$

App: Histogram-based Image Multi-Classification, Result

Criteria: error rate in percentage

- Benchmark, K-nearest neighbor (KNN) method

Database	KNN ℓ_2	KNN χ^2
Corel14	47.7	26.5
Corel7	51.4	35.4

- SVM

Database	linear	Poly 2	Radial basis function		
			Gaussian	Laplacian	χ^2
Corel14	36.3	33.6	28.8	14.7	14.7
Corel7	42.7	38.9	32.2	20.5	21.6

For Further Reading

Trevor Hastie, Robert Tibshirani, and Jerome Friedman,
The elements of statistical learning.
Springer New York, 2009.

Olivier Chapelle, Patrick Haffner, and Vladimir N. Vapnik,
"Support vector machines for histogram-based image classification,"
IEEE Transactions on Neural Networks. 10(5):1055–1044, 1999.

Thanks

For more information visit:

http://www.danielppalomar.com