Lab 5
Support Vector Machines

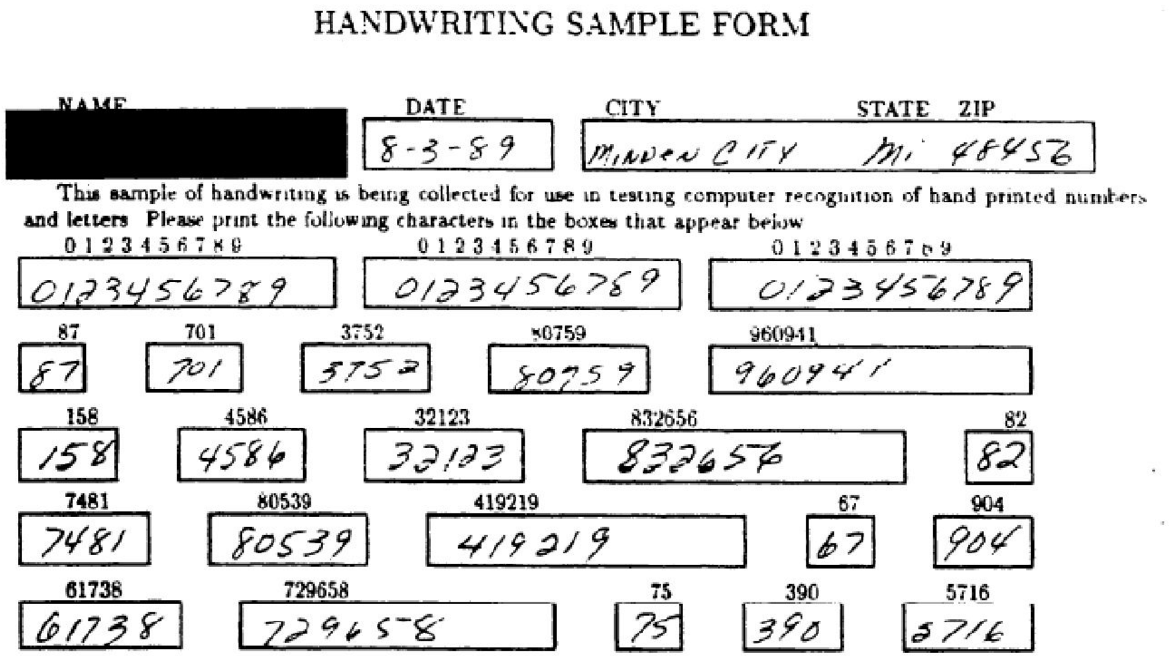
CREATED BY PROF. SUNDEEP RANGAN AND PROF. WANG FOR EE-UY 4563/EL-GY 9123

MODIFIED BY ARTIE SHEN

Outline

- Motivating example: Recognizing handwritten digits
- Maximum margin classifiers
- Support vector machines
- Multi-class classification problem
- Evaluation metrics
- Grid search
- Coding Exercise: apply SVM on the EMNIST dataset

MNIST Digit Classification


HANDWRITING SAMPLE FORM

NAME [redacted] DATE 8-3-89 CITY MINDEN CITY STATE MI ZIP 48456

This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below.

0123456789 0123456789 0123456789

87 701 3752 80759 960941

158 4586 32123 832656 82

7481 80539 419219 67 904

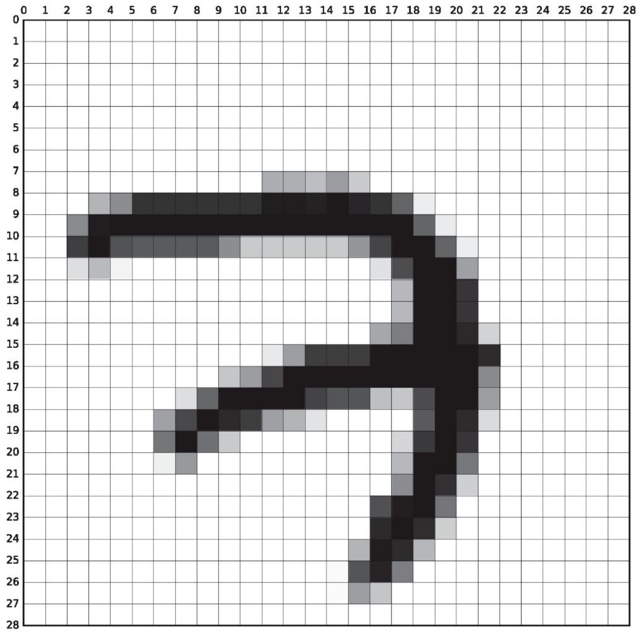
61738 729658 75 390 5716

From Patrick J. Grother, NIST Special Database, 1995

- Problem: Recognize handwritten digits
- Original problem:
  - Census forms
  - Automated processing
- Classic machine learning problem
- Benchmark

Problem Formulation

- Given an image of a handwritten digit
- Predict the number in the image
- {0,1,2,...,9}



Downloading MNIST

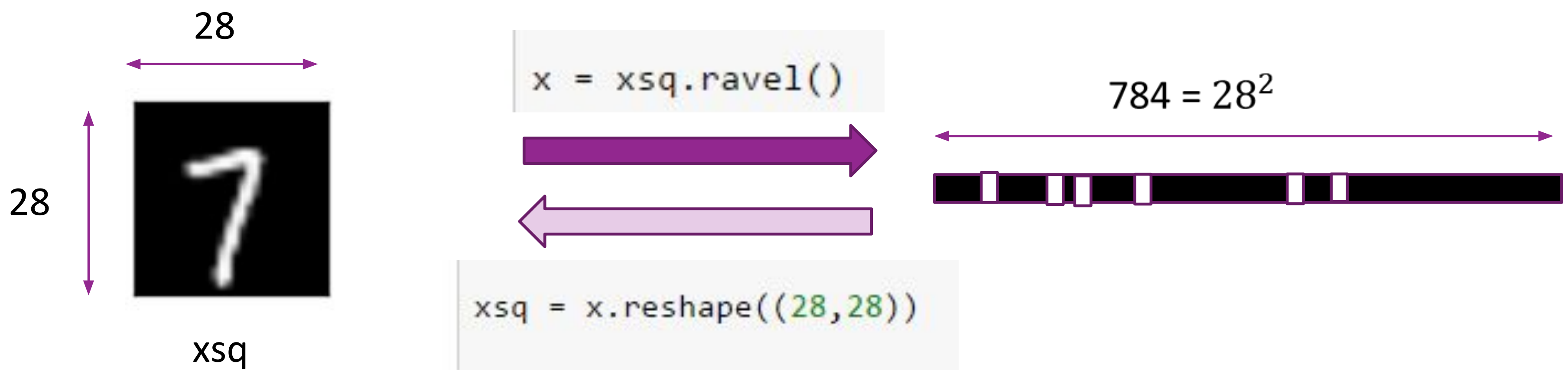
```python
import tensorflow as tf

(Xtr,ytr),(Xts,yts) = tf.keras.datasets.mnist.load_data()

print('Xtr shape: %s' % str(Xtr.shape))
print('Xts shape: %s' % str(Xts.shape))

ntr = Xtr.shape[0]
nts = Xts.shape[0]
nrow = Xtr.shape[1]
ncol = Xtr.shape[2]
```

```
Xtr shape: (60000, 28, 28)
Xts shape: (10000, 28, 28)
```

- MNIST data is available in many sources
- Note: It has been removed from sklearn
- Tensorflow version:
  - 60000 training samples
  - 10000 test samples
- Each sample is a 28 x 28 images
- Grayscale: Pixel values $\in \{0,1,\ldots,255\}$
  - 0 = Black and
  - 255 = White

Matrix and Vector Representation

- For this demo, we reshape data from $N \times 28 \times 28$ to $N \times 784$
- But, you can easily go back and forth
- Also, scale the pixel values from -1 to 1



$$S = \mathrm{Mat}(x) = \begin{bmatrix} s_{11} & \cdots & s_{1,28} \\ \vdots & \vdots & \vdots \\ s_{28,1} & \cdots & s_{28,28} \end{bmatrix} \qquad x = \mathrm{vec}(S) = [x_1 \quad \cdots \quad x_{784}]$$

Displaying Images in Python

4 random images in the dataset

A human can classify these easily

```
def plt_digit(x):
    nrow = 28
    ncol = 28
    xsq = x.reshape((nrow,ncol))
    plt.imshow(xsq,  cmap='Greys_r')
    plt.xticks([])
    plt.yticks([])

# Convert data to a matrix
X = mnist.data
y = mnist.target

# Select random digits
nplt = 4
nsamp = X.shape[0]
Iperm = np.random.permutation(nsamp)

# Plot the images using the subplot command
for i in range(nplt):
    ind = Iperm[i]
    plt.subplot(1,nplt,i+1)
    plt_digit(X[ind,:])
```

Key command

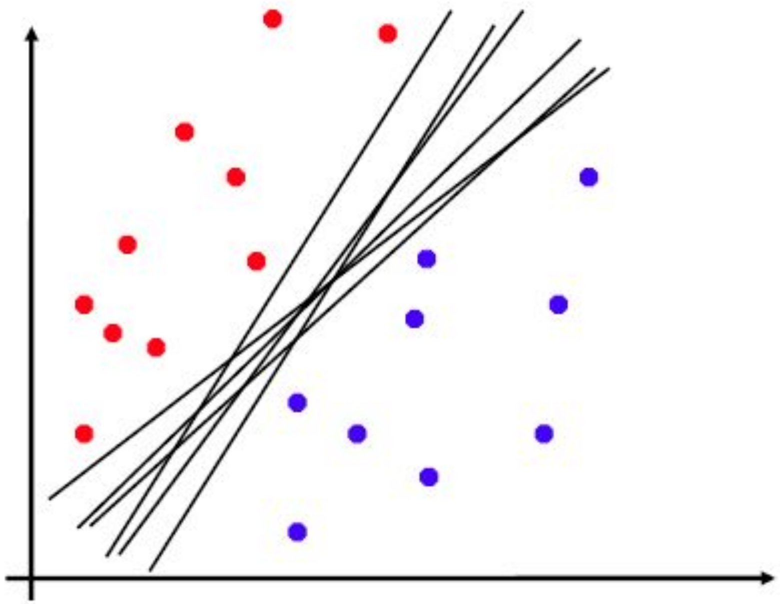
Sample permutation is necessary for this dataset, as the original data is ordered by digits

Outline

- ❑Motivating example: Recognizing handwritten digits
- ❑Maximum margin classifiers
- ❑Support vector machines
- ❑Multi-class classification problem
- ❑Evaluation metrics
- ❑Grid search
- ❑Coding Exercise: apply SVM on the EMNIST dataset

Linear Separability and Non-Uniqueness of Separating plane

- When the samples are linearly separable, one can find a separating hyper-plane as a linear classifier.
- Separating hyper-plane is not unique
- Fig. on right: Many separating planes
- Which one is optimal?
- Desired Properties:
  - Correctness
  - Robustness



Hyperplane Basics

- A hyperplane in d-dimensional space is defined by

$$b + w_1 x_1 + \cdots w_d x_d = 0 \;\; or \;\; b + \boldsymbol{w}^T \boldsymbol{x} = 0$$

- The parameters are unique only to a scaling factor:
  - $(b, \boldsymbol{w})$ and $(\alpha b, \alpha \boldsymbol{w})$ define the same plane.
  - For unique definition, we can require $\|\boldsymbol{w}\|=1$.
- The norm vector to the hyperplane is $\boldsymbol{w}/\|\boldsymbol{w}\|$.
- Distance of any point **x** to the hyperplane is $f(\boldsymbol{x})/\|\boldsymbol{w}\|$, where $f(\boldsymbol{x}) = b + \boldsymbol{w}^T \boldsymbol{x}$.
- See ESL Sec. 4.5.

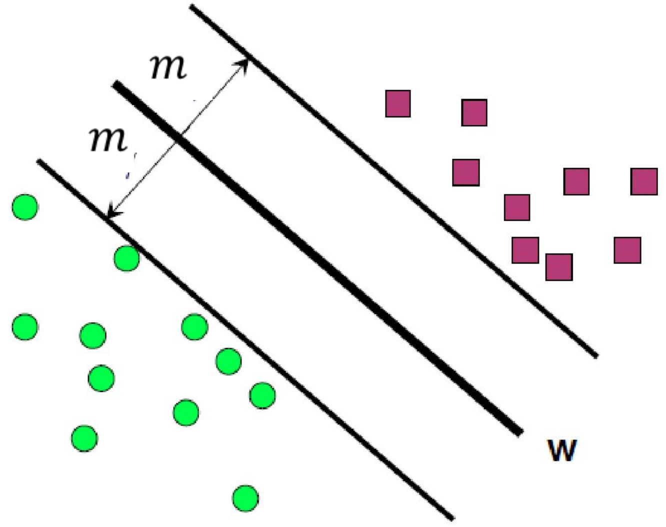
- ESL: Hastie, Tibshirani, Friedman, “The Elements of Statistical Learning”. 2nd Ed. Springer.

Recap: Linear Separability and Margin

- Given training data $(\boldsymbol{x}_i, y_i), i = 1, \dots, N$
- Binary class label: $y_i = \pm 1$
- Perfectly linearly separable if there exists a $\boldsymbol{\theta} = (b, w_1, \dots, w_d)$ and $\gamma > 0$ s.t.:
  - $b + w_1 x_{i1} + \cdots w_d x_{id} > \gamma$ when $y_i = 1$
  - $b + w_1 x_{i1} + \cdots w_d x_{id} < -\gamma$ when $y_i = -1$
- $(\boldsymbol{w}, b)$ defines the separating hyperplane
- m is the margin: the minimal distance of a sample to the plane
- Single equation form:

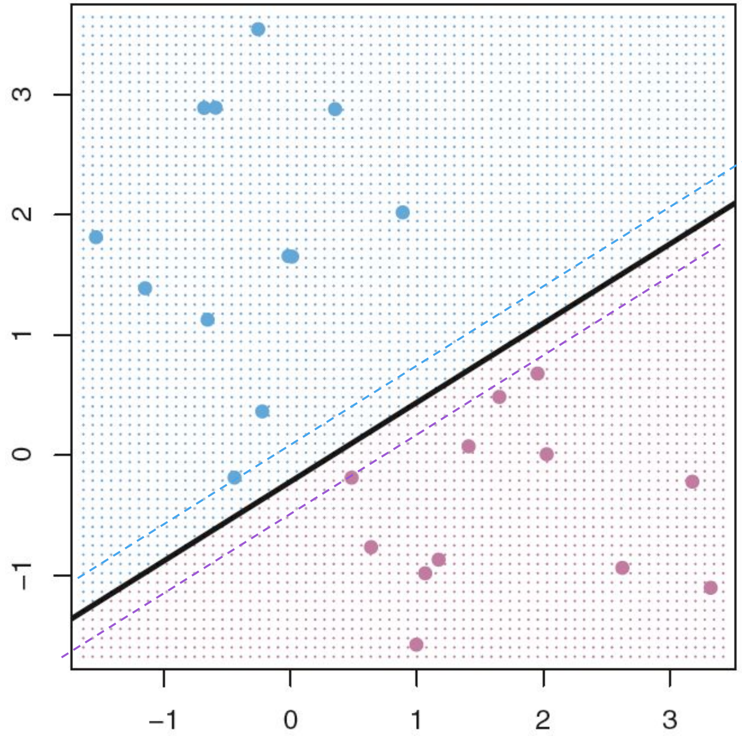
$$y_i(b + w_1 x_{i1} + \cdots w_d x_{id}) > \gamma \text{ for all } i = 1, \dots, N$$

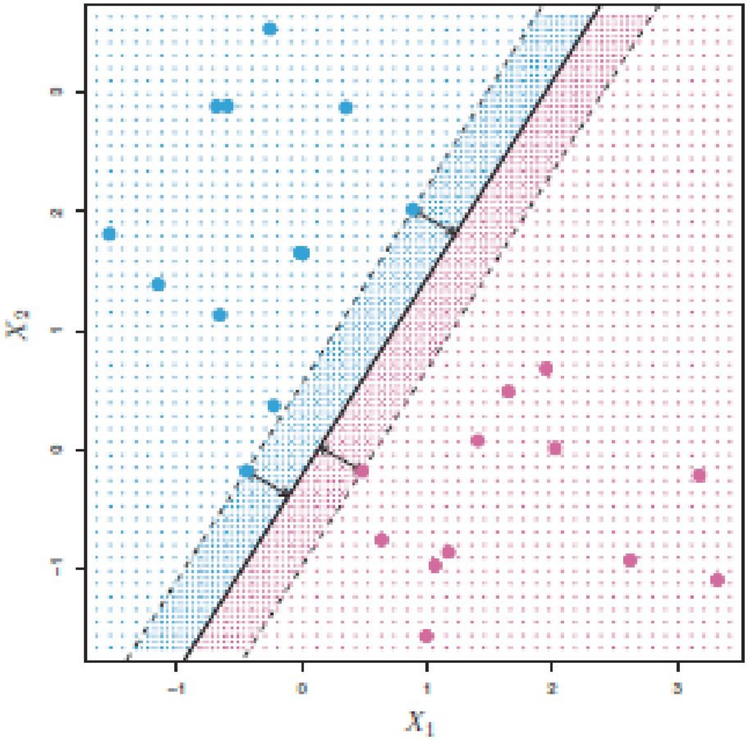
$$\mathrm{m} = \frac{\gamma}{\|\boldsymbol{w}\|}$$



Recall that the distance of a point x to the line is $(b + w^T \boldsymbol{x})/\|\boldsymbol{w}\|$.
For points on the margin line, $b + w^T \boldsymbol{x} = \gamma$, distance m= $\gamma / \|\boldsymbol{w}\|$.

Which separating plane is better ?





From Fig. 9.2 and Fig. 9.3 in ISL.

Maximum Margin Classifier

- For the classifier to be more robust to noise, we want to maximize the margin!

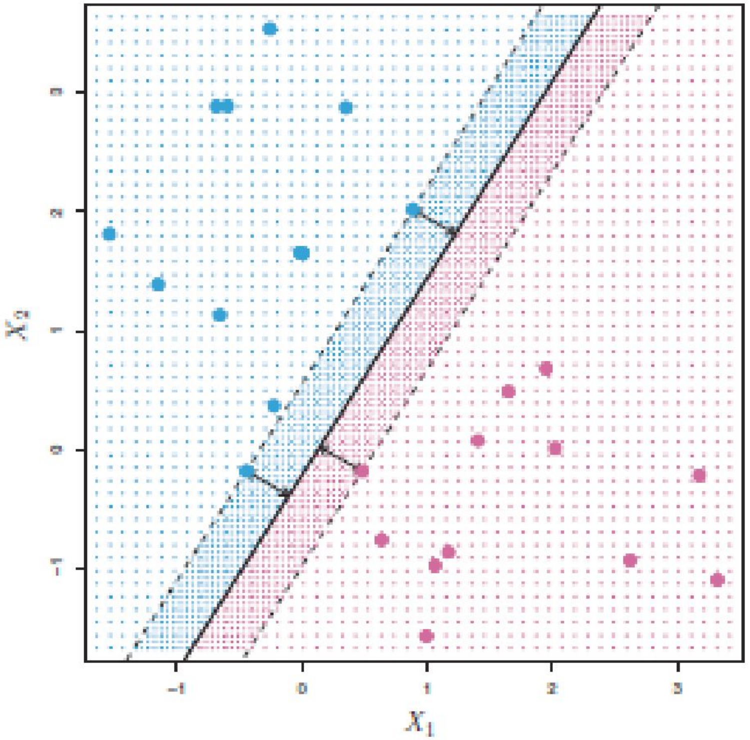
- Define maximum margin classifier

  $\max_{w,\gamma} \gamma$ — Maximizes the margin
  - Such that $y_i(b + \boldsymbol{w}^T \boldsymbol{x}) \geq \gamma$ for all i — Ensures all points are correctly classified
  - $\sum_{j=1}^{d} w_j^2 \leq 1$ — Scaling on weights

- Called a constrained optimization
  - Objective function and constraints
  - More on this later.

- See closed form solution in Sec. 4.5.2 in ESL. Note notation difference.

Visualizing Maximum Margin Classifier



- Margin determined by closest points to the line
  - The maximal margin hyperplane represents the mid-line of the widest “slab” that we can insert between two classes
- In this figure, there are 3 points at the margin

ISL: James, Witten, Hastie, Tibshirani, An Introduction to Statistical Learning, Springer. 2013.

Problems with MM classifier

- Data is often not perfectly separable
  - Only want to correctly separate most points

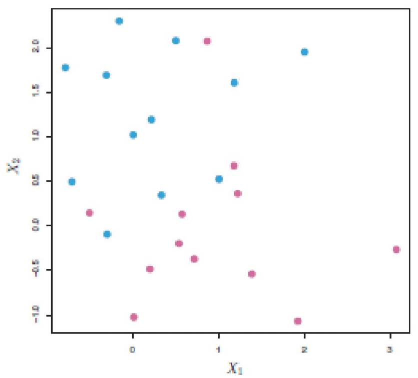


Fig. 9.4

- MM classifier is not robust
  - A single sample can radically change line

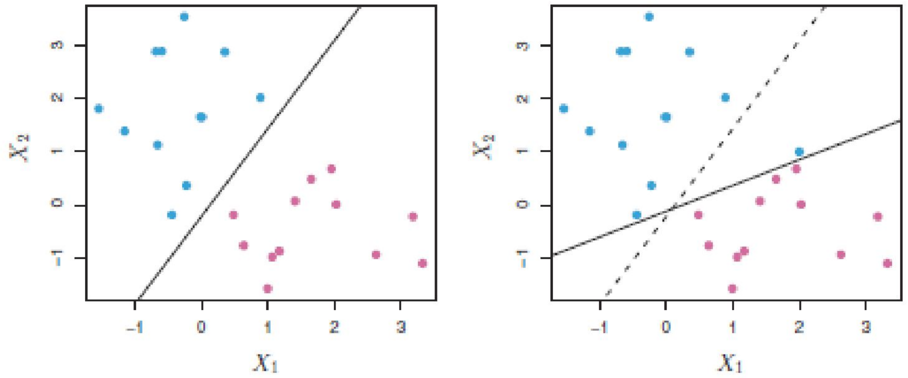


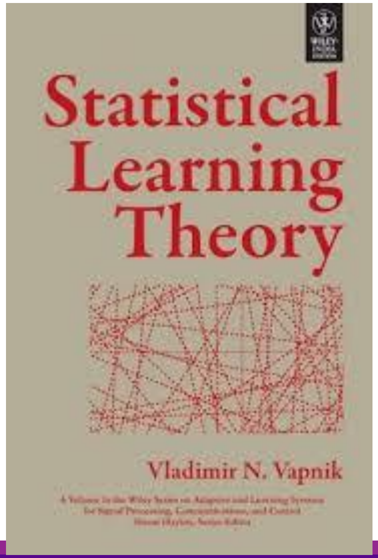
Fig. 9.5

Outline

- Motivating example: Recognizing handwritten digits
- Maximum margin classifiers
- Support vector machines
- Multi-class classification problem
- Evaluation metrics
- Grid search
- Coding Exercise: apply SVM on the EMNIST dataset

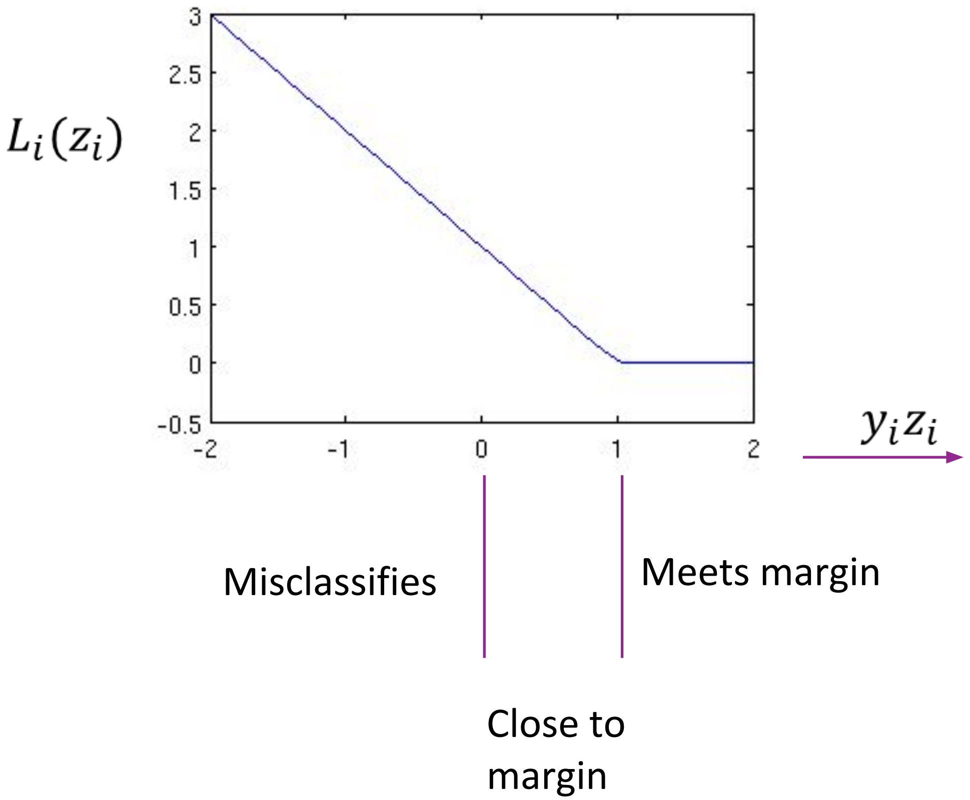
Support Vector Machine

- Key idea: Allow “slack” in the classification
  - Support vector classifier (SVC): Directly use raw features. Good when the original feature space is roughly linearly separable
  - Support vector machine (SVM): Map the raw features to some other domain through a kernel function



Hinge Loss

- Fix $\gamma = 1$
- Want ideally: $y_i(\boldsymbol{w}^T\boldsymbol{x} + b) \geq 1$ for all samples i
  - Equivalently, $y_i z_i \geq 1, \quad z_i = b + \boldsymbol{w}^T\boldsymbol{x}$
- But, perfect separation may not be possible
- Define hinge loss or soft margin:
  - $L_i(\boldsymbol{w}, b) = \max(0, 1 - y_i z_i)$
- Starts to increase as sample is misclassified:
  - $y_i z_i \geq 1 \;\Rightarrow$ Sample meets margin target, $L_i(w) = 0$
  - $y_i z_i \in [0,1) \;\Rightarrow$ Sample margin too small, small loss
  - $y_i z_i \leq 0 \;\Rightarrow$ Sample misclassified, large loss



SVM Optimization

- Given data $(\boldsymbol{x}_i, y_i)$
- Optimization $\min_{w,b} J(\mathbf{w}, b)$

$$J(\mathbf{w}, b) = C \sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^T \boldsymbol{x}_i + b)) + \frac{1}{2} \|\mathbf{w}\|^2$$

Hinge loss term
Attempts to reduce
Misclassifications

C controls final margin

margin=$1/\|\boldsymbol{w}\|$

- Constant $C > 0$ will be discussed below
- Note: ISL book uses different naming conventions.
  - We have followed convention in sklearn

Alternate Form of SVM Optimization

- Equivalent optimization:

$$\min J_1(\boldsymbol{w}, b, \boldsymbol{\epsilon}), \qquad J_1(\boldsymbol{w}, b, \boldsymbol{\epsilon}) = C \sum_{i=1}^{N} \epsilon_i + \frac{1}{2} \|\boldsymbol{w}\|^2$$

- Subject to constraints:

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1 - \epsilon_i \text{ for all } i = 1, \ldots, N$$

  - ϵ_i = amount sample i misses margin target

- Sometimes write as $J_1(\boldsymbol{w}, b, \boldsymbol{\epsilon}) = C\|\boldsymbol{\epsilon}\|_1 + \frac{1}{2}\|\boldsymbol{w}\|^2$

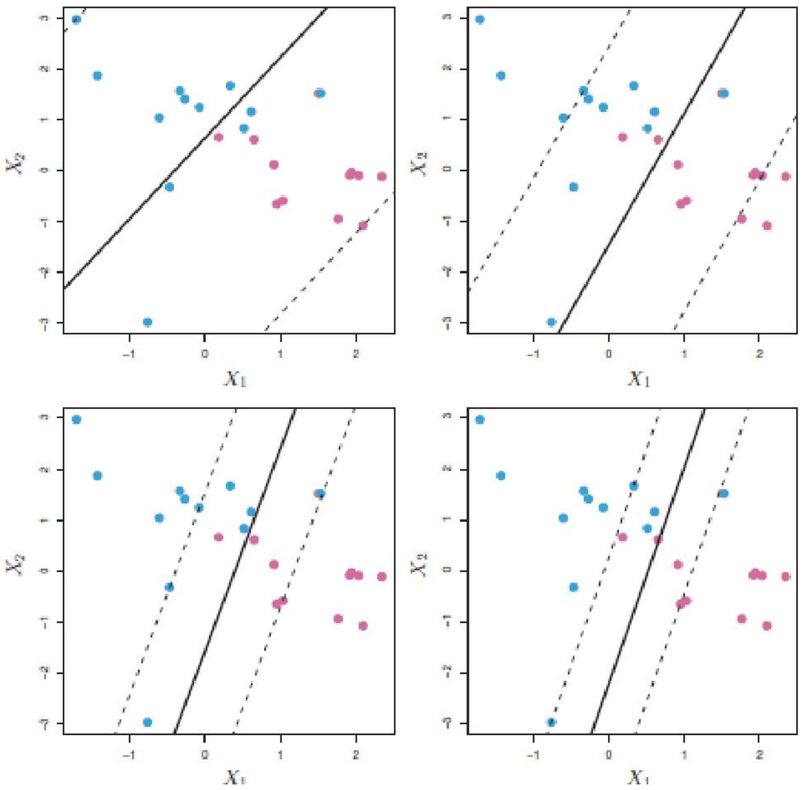
  - $\|\boldsymbol{\epsilon}\|_1 = \sum_{i=1}^{N} \epsilon_i$ called the "one-norm"
  - Generally one-norm would have absolute sign over ϵ_i. But in this case, when the constraint is met, ϵ_i>=0.

Interpreting Parameters

- Margin is $1/\|\boldsymbol{w}\|$
- Parameter ϵ_i called the slack variable
  - $\epsilon_i = 0 \Rightarrow$ Sample on correct side of margin
  - $0 \leq \epsilon_i < 1 \Rightarrow$ Sample violates the margin (are inside the margin)
  - $\epsilon_i \geq 1 \Rightarrow$ Sample misclassified (wrong side of hyperplane)
- Parameter C:
  - Balance between first term (violations) and second term (inverse of margin)
  - C large: Forces minimum number of violations, but small margin.
    - Highly fit to data. Low bias, higher variance
  - C small: Enables more samples violations, but large margin.
    - Higher bias, lower variance
  - Found by cross-validation

Illustrating Effect of C

Low $C = C_1$ $\quad$ C_2

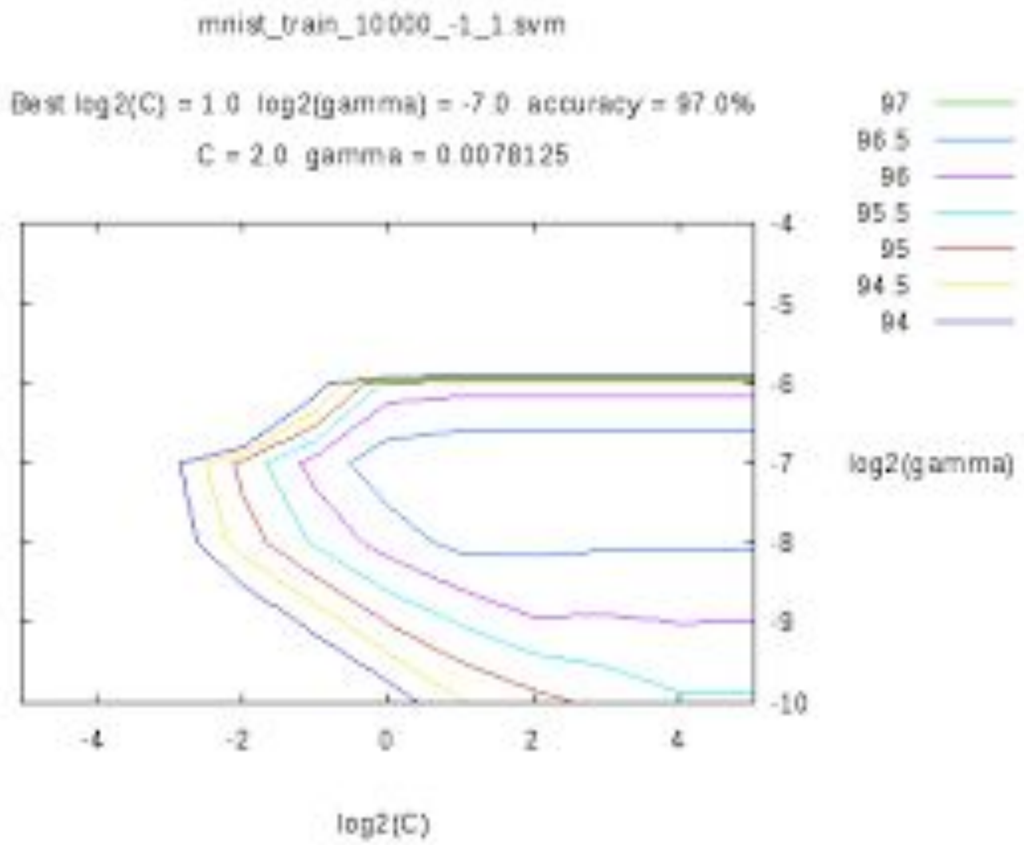


C_3 $\quad$ High C_4

- Fig. 9.7 of ISL
  - Note: C has opposite meaning in ISL than python
  - Here, we use python meaning
- Low C:
  - Leads to large margin
  - But allow many violations of margin.
  - Many more SVs
  - Reduces variance by using more samples
- Large C:
  - Leads to small margin
  - Reduce number of violations, and fewer SVs.
  - Highly fit to data. Low bias, higher variance
  - More chance to overfit

Parameter Selection

- ❑Consider SVM with:
  - Parameter $C > 0$, RBF with $\gamma > 0$
- ❑Higher C or γ
  - Fewer SVs
  - Classifiers averages over smaller set
  - Lower bias, but higher variance
- ❑Typically select via cross-validation
  - Try out different (C, γ)
  - Find which one provides highest accuracy on test set
- ❑Python can automatically do grid search



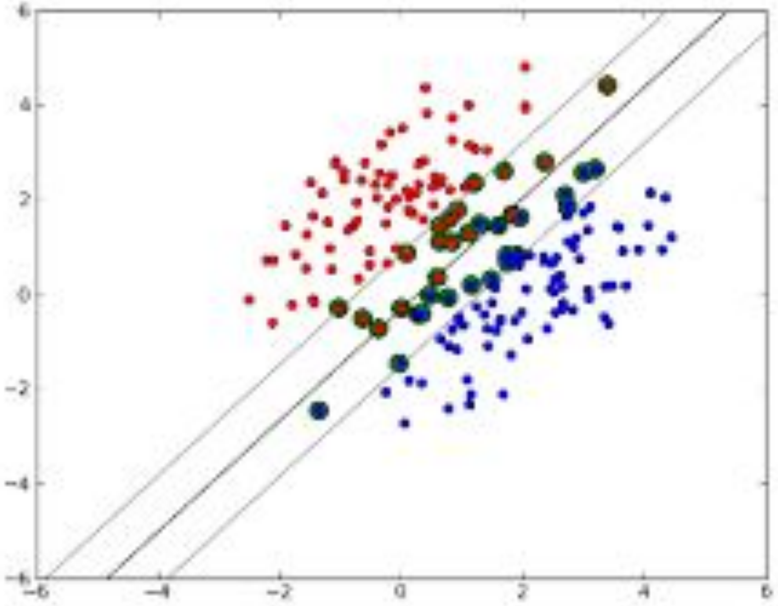
http://peekaboo-vision.blogspot.com/2010/09/mnist-for-ever.html

Hyperparameter Search

- **Grid Search + Cross Validation**
  - for C in [c1, c2, …, ck]:
    - for gamma in [g1, g2, …, gk]:
      - do 10-fold CV
  - report best performance
- **Random search**
  - for i in range(n_iter):
    - sample hyper_param_i from a generative distribution
      - train a model using hyper_param_i
      - evaluate the model on the validation set
  - report best performance
- **Bayesian Optimization**
  - Build a surrogate probability model of the objective function
  - Find the hyperparameters that perform best on the surrogate
  - Apply these hyperparameters to the true objective function
  - Update the surrogate model incorporating the new results
  - Repeat steps above
- More to read:
  https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f

Support Vectors

- **Support vectors:** Samples that either:
  - Are exactly on margin: $y_{\text{i}}(\boldsymbol{w}^T\boldsymbol{x}_i + b) = 1$
  - Or, on wrong side of margin: $y_{\text{i}}(\boldsymbol{w}^T\boldsymbol{x}_i + b) \le 1$
- Changing samples that are not SVs
  - Does not change solution
  - Provides robustness



Outline

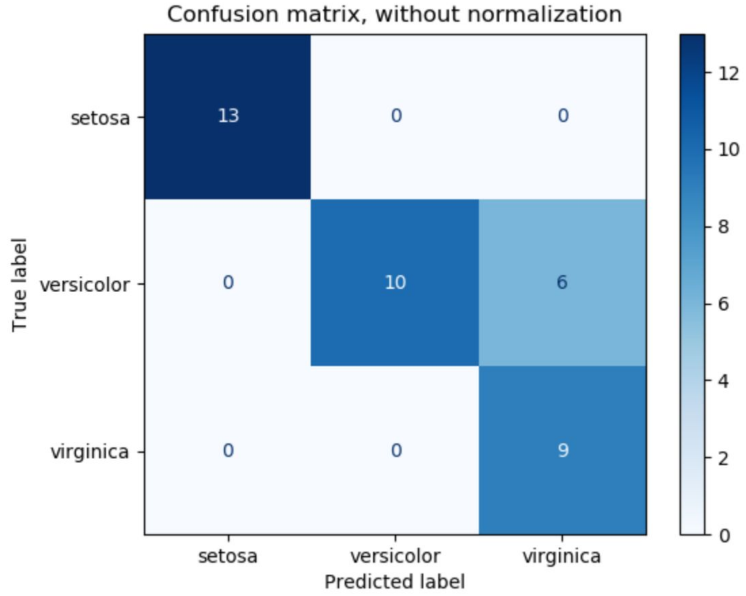
- ❑Motivating example: Recognizing handwritten digits
- ❑Maximum margin classifiers
- ❑Support vector machines
- ❑Multi-class classification problem
- ❑Coding Exercise: apply SVM on the EMNIST dataset

Multi-Class SVMs

- Suppose there are K classes
- One-vs-one:
  - Train $\binom{K}{2}$ SVMs for each pair of classes
  - Test sample assigned to class that wins “majority of votes”
  - Best results but very slow
- One-vs-rest:
  - Train K SVMs: train each class k against all other classes
  - Pick class with highest z_k
- Sklearn has both options

Evaluation Metrics

- Accuracy
- Confusion Matrix
  - By definition a confusion matrix C is such that C_{i,j} is equal to the number of observations known to be in group i and predicted to be in group j.



Evaluation Metrics

- **Macro v.s. Micro**
  - Macro-averaging metrics give equal weight to each class.
  - Micro-averaging evaluation metrics, on the other hand, weight all items equally.
- Example:
  - Precision = TP / (TP + FP)
  - Suppose that we have 4 classes: {A,B,C,D}
  - Class A: 1 TP and 1 FP, Precision_A = 1/2
  - Class B: 10 TP and 90 FP, , Precision_B = 1/10
  - Class C: 1 TP and 1 FP, Precision_C = 1/2
  - Class D: 1 TP and 1 FP, Precision_D = 1/2
  - Precision_Macro = (Precision_A+Precision_B+Precision_C+Precision_D)/4 = 0.4
  - Precision_Micro = (TP_A + TP_B + TP_C + TP_D) / (2+100+2+2) = 0.123

Evaluation Metrics

Metric	Formula	Evaluation focus
Average Accuracy	$\frac{\sum_{i=1}^{k} \frac{tp_i+tn_i}{tp_i+tn_i+fp_i+tn_i}}{k}$	The average per-class effectiveness of the classifier
Error Rate	$\frac{\sum_{i=1}^{k} \frac{fp_i+fn_i}{tp_i+tn_i+fp_i+tn_i}}{k}$	The average per-class classification error
Precision$_\mu$	$\frac{\sum_{i=1}^{k} tp_i}{\sum_{i=1}^{k}(tp_i + fp_i)}$	Agreement of the true class labels with those of the classifier's, calculated by summing all TPs and and FPs in the system, across all classes
Recall$_\mu$	$\frac{\sum_{i=1}^{k} tp_i}{\sum_{i=1}^{k}(tp_i + fn_i)}$	Effectiveness of a classifier to identify class labels, calculated by summing all TPs and and FNs in the system, across all classes
F1-score$_\mu$	$\frac{2 * Precision_\mu * Recall_\mu}{Precision_\mu + Recall_\mu}$	The harmonic mean of the micro-average precision and recall
Precision$_M$	$\frac{\sum_{i=1}^{k} \frac{tp_i}{tp_i+fp_i}}{k}$	Average per-class agreement of the true class labels with those of the classifier's
Recall$_M$	$\frac{\sum_{i=1}^{k} \frac{tp_i}{tp_i+fn_i}}{k}$	Average per-class effectiveness of a classifier to identify class labels
F1-score$_M$	$\frac{2 * Precision_M * Recall_M}{Precision_M + Recall_M}$	The harmonic mean of the macro-average precision and recall

Evaluation Metrics

- Macro v.s. Micro
  - Macro-level metrics give equal weight to each class.
  - Micro-level evaluation metrics, on the other hand, weight all items equally.
- Macro-AUC
  - Example: 3-class classification problem, prediction = (p1, p2, p3)
  - Calculate AUC for class 1: score = p1, label = 1 if y = 1 0 if y = 2 or 3
  - Average AUC across class 1, 2, 3

MNIST Results

- ❑Run classifier
- ❑Very slow
  - Several minutes for 40,000 samples
  - Slow in training and test
  - Major drawback of SVM
- ❑Accuracy ≈ 0.984
  - Much better than logistic regression
- ❑Can get better with:
  - pre-processing
  - More training data
  - Optimal parameter selection

```
from sklearn import svm

# Create a classifier: a support vector classifier
svc = svm.SVC(probability=False,  kernel="rbf", C=2.8, gamma=.0073,verbose=10)
```

```
svc.fit(Xtr,ytr)
```

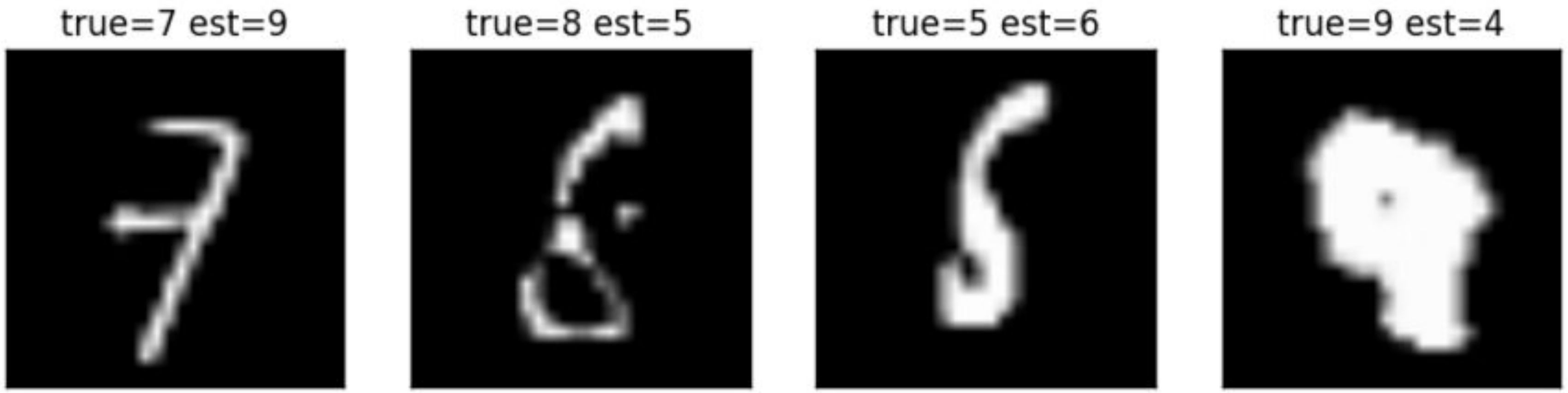
```
[LibSVM]

SVC(C=2.8, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.0073, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=10)
```

```
yhat1 = svc.predict(Xts)
acc = np.mean(yhat1 == yts)
print('Accuaracy = {0:f}'.format(acc))
```

```
Accuaracy = 0.984000
```

MNIST Errors

- Some of the error are hard even for a human



What you should know

- ❑Understand the margin in linear classification and maximum margin classifier
- ❑SVM classifier: Allow violation of margin by introducing slack variables (More robust than linear classifier)
- ❑Select SVM parameters from cross-validation
- ❑Adapt a binary classification model for multi-class problems
- ❑Evaluate the effectiveness of multi-class classifiers

Coding Exercise

- Now let's apply SVM to the EMNIST dataset
- Also contains letters