Speech Emotion Recognition - Week 3-4

Model Training & Hyperparameter Tuning

Krishna Kukreja & Garvit Meena

November 2025

Week 3-4: Resources & Guide

1. Learning Objectives

- Understand SVM and Random Forest classifiers.
- Implement train-test-validation split.
- Perform hyperparameter tuning with GridSearchCV.
- Apply k-fold cross-validation.
- Compare multiple models.
- Interpret confusion matrices and metrics.

2. Reading Materials

- **SVM Tutorial**: DataCamp - Support Vector Machines with Scikit-learn
- **Random Forest**: Scikit-learn Random Forest Guide
- **Cross-Validation**: Understanding Cross-Validation
- **GridSearch**: Grid Search Complete Guide

3. Video Tutorials

- Machine Learning Tutorial - SVM in Scikit-learn
- Classification with SVM in Python - Practical Guide
- Sklearn GridSearchCV Tutorial - Cross-Validation

4. Python Code Templates

Train-Test Split and Normalization

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load features
df = pd.read_csv('features.csv')
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

# Split data (70% train, 15% val, 15% test)
X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y)

X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, random_state=42,
    stratify=y_temp)

# Normalize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)

print(f"Training: {X_train_scaled.shape[0]} samples")
print(f"Validation: {X_val_scaled.shape[0]} samples")
print(f"Test: {X_test_scaled.shape[0]} samples")
```

SVM Training with Grid Search

```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score

# Define parameter grid
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': ['scale', 'auto', 0.001, 0.01, 0.1],
    'kernel': ['rbf', 'linear']
}

# GridSearchCV
grid_search = GridSearchCV(
    estimator=SVC(probability=True, random_state=42),
    param_grid=param_grid,
    cv=5,
```

```
    scoring='accuracy',
    verbose=2,
    n_jobs=-1
)

# Train
grid_search.fit(X_train_scaled, y_train)

# Best model
best_svm = grid_search.best_estimator_
print(f"Best parameters: {grid_search.best_params_}")
print(f"Best CV score: {grid_search.best_score_:.4f}")

# Validate
y_val_pred = best_svm.predict(X_val_scaled)
val_acc = accuracy_score(y_val, y_val_pred)
print(f"Validation Accuracy: {val_acc:.4f}")
```

Random Forest Training

```
from sklearn.ensemble import RandomForestClassifier

param_grid_rf = {
    'n_estimators': [50, 100, 200],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10]
}

grid_search_rf = GridSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_grid=param_grid_rf,
    cv=5,
    scoring='accuracy',
    verbose=2,
    n_jobs=-1
)

grid_search_rf.fit(X_train_scaled, y_train)
best_rf = grid_search_rf.best_estimator_
print(f"Best RF params: {grid_search_rf.best_params_}")
```

Confusion Matrix Visualization

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

cm = confusion_matrix(y_val, y_val_pred)
emotion_labels = ['Neutral', 'Calm', 'Happy', 'Sad',
```

```
                'Angry', 'Fearful', 'Disgust', 'Surprised']

plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=emotion_labels,
            yticklabels=emotion_labels)
plt.title('Confusion Matrix - SVM Model')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

5. Assignments

1. Implement proper train-test-val split (70-15-15) with stratification.
2. Train baseline SVM and Random Forest with default parameters.
3. Perform grid search hyperparameter tuning for both models.
4. Generate confusion matrices and classification reports.
5. Create model comparison table with accuracy, precision, recall, F1-score.

6. Expected Outputs

- Trained SVM model (.pkl file) with validation accuracy $\geq 75\%$.
- Trained Random Forest model (.pkl file).
- Saved StandardScaler (.pkl file).
- Confusion matrices for both models.
- Classification reports (precision, recall, F1 per emotion).
- Model comparison document (2-3 pages).

7. Additional Resources

- GeeksforGeeks - Implementing SVM with Scikit-Learn
- W3Schools - Python Confusion Matrix
- GridSearchCV API Reference