# Task A1 for Gender Recognition

# Tuning hyper-parameters for precision

Best parameters set found on training dataset:

{'C': 0.01, 'kernel': 'linear'}

Grid scores on training dataset:

0.253 (+/-0.001) for {'C': 0.01, 'kernel': 'rbf'}

0.253 (+/-0.001) for {'C': 0.05, 'kernel': 'rbf'}

0.453 (+/-0.490) for {'C': 0.1, 'kernel': 'rbf'}

0.820 (+/-0.035) for {'C': 0.5, 'kernel': 'rbf'}

0.859 (+/-0.039) for {'C': 1, 'kernel': 'rbf'}

0.894 (+/-0.027) for {'C': 5, 'kernel': 'rbf'}

0.901 (+/-0.030) for {'C': 10, 'kernel': 'rbf'}

0.922 (+/-0.020) for {'C': 0.01, 'kernel': 'linear'}

0.920 (+/-0.018) for {'C': 0.05, 'kernel': 'linear'}

0.918 (+/-0.019) for {'C': 0.1, 'kernel': 'linear'}

0.918 (+/-0.022) for {'C': 0.5, 'kernel': 'linear'}

0.916 (+/-0.024) for {'C': 1, 'kernel': 'linear'}

0.916 (+/-0.023) for {'C': 5, 'kernel': 'linear'}

0.916 (+/-0.021) for {'C': 10, 'kernel': 'linear'}

0.880 (+/-0.029) for {'C': 0.01, 'kernel': 'poly'}

0.901 (+/-0.029) for {'C': 0.05, 'kernel': 'poly'}

0.904 (+/-0.023) for {'C': 0.1, 'kernel': 'poly'}

0.915 (+/-0.024) for {'C': 0.5, 'kernel': 'poly'}

0.918 (+/-0.021) for {'C': 1, 'kernel': 'poly'}

0.922 (+/-0.020) for {'C': 5, 'kernel': 'poly'}

0.921 (+/-0.021) for {'C': 10, 'kernel': 'poly'}

Detailed classification report:

The model is trained on the full training dataset.

The scores are computed on the full testing dataset.

precision recall f1-score support

0.0 0.92 0.93 0.93 487

1.0 0.93 0.92 0.92 473

accuracy 0.93 960

macro avg 0.93 0.92 0.92 960

weighted avg 0.93 0.93 0.92 960

The confusion matrix is:

[[453 34]

[ 38 435]]

Best estimator found: SVC(C=0.01, break\_ties=False, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3, gamma='scale', kernel='linear',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

Best parameters set found: {'C': 0.01, 'kernel': 'linear'}

SVM with GridCV on testing data - Accuracy Score: 0.925 (+/- 0.000)

SVM with GridCV on training data - Accuracy Score: 0.934 (+/- 0.000)

A close up of a map

Description automatically generatedA close up of a map

Description automatically generatedA close up of a map

Description automatically generated

# Task A2 for Smiling Recognition

# Tuning hyper-parameters for precision

Best parameters set found on training dataset:

{'C': 0.5, 'kernel': 'poly'}

Grid scores on training dataset:

0.254 (+/-0.001) for {'C': 0.01, 'kernel': 'rbf'}

0.846 (+/-0.020) for {'C': 0.05, 'kernel': 'rbf'}

0.863 (+/-0.013) for {'C': 0.1, 'kernel': 'rbf'}

0.878 (+/-0.015) for {'C': 0.5, 'kernel': 'rbf'}

0.881 (+/-0.017) for {'C': 1, 'kernel': 'rbf'}

0.890 (+/-0.013) for {'C': 5, 'kernel': 'rbf'}

0.891 (+/-0.014) for {'C': 10, 'kernel': 'rbf'}

0.891 (+/-0.013) for {'C': 0.01, 'kernel': 'linear'}

0.886 (+/-0.016) for {'C': 0.05, 'kernel': 'linear'}

0.882 (+/-0.024) for {'C': 0.1, 'kernel': 'linear'}

0.879 (+/-0.019) for {'C': 0.5, 'kernel': 'linear'}

0.879 (+/-0.024) for {'C': 1, 'kernel': 'linear'}

0.878 (+/-0.021) for {'C': 5, 'kernel': 'linear'}

0.878 (+/-0.022) for {'C': 10, 'kernel': 'linear'}

0.884 (+/-0.011) for {'C': 0.01, 'kernel': 'poly'}

0.890 (+/-0.017) for {'C': 0.05, 'kernel': 'poly'}

0.893 (+/-0.011) for {'C': 0.1, 'kernel': 'poly'}

0.894 (+/-0.019) for {'C': 0.5, 'kernel': 'poly'}

0.893 (+/-0.018) for {'C': 1, 'kernel': 'poly'}

0.891 (+/-0.015) for {'C': 5, 'kernel': 'poly'}

0.889 (+/-0.019) for {'C': 10, 'kernel': 'poly'}

Detailed classification report:

The model is trained on the full training dataset.

The scores are computed on the full testing dataset.

precision recall f1-score support

0.0 0.88 0.90 0.89 456

1.0 0.91 0.89 0.90 504

accuracy 0.89 960

macro avg 0.89 0.89 0.89 960

weighted avg 0.90 0.89 0.89 960

The confusion matrix is:

[[410 46]

[ 55 449]]

Best estimator found: SVC(C=0.5, break\_ties=False, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3, gamma='scale', kernel='poly',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

Best parameters set found: {'C': 0.5, 'kernel': 'poly'}

92.

SVM with GridCV on testing data - Accuracy Score: 0.895 (+/- 0.000)

SVM with GridCV on training data - Accuracy Score: 0.901 (+/- 0.000)

# Task B1 for Eye Colour Recognition

# Tuning hyper-parameters for precision

Best parameters set found on training dataset:

{'C': 0.01, 'kernel': 'poly'}

Grid scores on training dataset:

0.041 (+/-0.000) for {'C': 0.01, 'kernel': 'rbf'}

0.041 (+/-0.000) for {'C': 0.05, 'kernel': 'rbf'}

0.202 (+/-0.020) for {'C': 0.1, 'kernel': 'rbf'}

0.460 (+/-0.042) for {'C': 0.5, 'kernel': 'rbf'}

0.639 (+/-0.027) for {'C': 1, 'kernel': 'rbf'}

0.795 (+/-0.027) for {'C': 5, 'kernel': 'rbf'}

0.801 (+/-0.026) for {'C': 10, 'kernel': 'rbf'}

0.782 (+/-0.039) for {'C': 0.01, 'kernel': 'linear'}

0.782 (+/-0.039) for {'C': 0.05, 'kernel': 'linear'}

0.782 (+/-0.039) for {'C': 0.1, 'kernel': 'linear'}

0.782 (+/-0.039) for {'C': 0.5, 'kernel': 'linear'}

0.782 (+/-0.039) for {'C': 1, 'kernel': 'linear'}

0.782 (+/-0.039) for {'C': 5, 'kernel': 'linear'}

0.782 (+/-0.039) for {'C': 10, 'kernel': 'linear'}

0.840 (+/-0.033) for {'C': 0.01, 'kernel': 'poly'}

0.823 (+/-0.032) for {'C': 0.05, 'kernel': 'poly'}

0.813 (+/-0.046) for {'C': 0.1, 'kernel': 'poly'}

0.794 (+/-0.047) for {'C': 0.5, 'kernel': 'poly'}

0.794 (+/-0.046) for {'C': 1, 'kernel': 'poly'}

0.794 (+/-0.046) for {'C': 5, 'kernel': 'poly'}

0.794 (+/-0.046) for {'C': 10, 'kernel': 'poly'}

Detailed classification report:

The model is trained on the full training dataset.

The scores are computed on the full testing dataset.

precision recall f1-score support

0 0.82 0.79 0.81 176

1 0.89 0.79 0.84 147

2 0.98 0.73 0.84 166

3 0.79 0.85 0.82 157

4 0.65 0.88 0.75 154

accuracy 0.81 800

macro avg 0.83 0.81 0.81 800

weighted avg 0.83 0.81 0.81 800

The confusion matrix is:

[[139 6 0 12 19]

[ 6 116 1 8 16]

[ 8 5 121 12 20]

[ 3 1 1 133 19]

[ 13 2 0 3 136]]

Best estimator found: SVC(C=0.01, break\_ties=False, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3, gamma='scale', kernel='poly',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

Best parameters set found: {'C': 0.01, 'kernel': 'poly'}

SVM with GridCV on testing data - Accuracy Score: 0.806 (+/- 0.000)

SVM with GridCV on training data - Accuracy Score: 0.869 (+/- 0.000)

# Task B2 for Face Shape Recognition

# Tuning hyper-parameters for precision

Best parameters set found on training dataset:

{'C': 0.5, 'kernel': 'poly'}

Grid scores on training dataset:

0.051 (+/-0.001) for {'C': 0.01, 'kernel': 'rbf'}

0.051 (+/-0.001) for {'C': 0.05, 'kernel': 'rbf'}

0.051 (+/-0.001) for {'C': 0.1, 'kernel': 'rbf'}

0.051 (+/-0.001) for {'C': 0.5, 'kernel': 'rbf'}

0.051 (+/-0.001) for {'C': 1, 'kernel': 'rbf'}

0.140 (+/-0.025) for {'C': 5, 'kernel': 'rbf'}

0.226 (+/-0.076) for {'C': 10, 'kernel': 'rbf'}

0.645 (+/-0.084) for {'C': 0.01, 'kernel': 'linear'}

0.606 (+/-0.114) for {'C': 0.05, 'kernel': 'linear'}

0.581 (+/-0.110) for {'C': 0.1, 'kernel': 'linear'}

0.524 (+/-0.129) for {'C': 0.5, 'kernel': 'linear'}

0.526 (+/-0.114) for {'C': 1, 'kernel': 'linear'}

0.534 (+/-0.105) for {'C': 5, 'kernel': 'linear'}

0.534 (+/-0.105) for {'C': 10, 'kernel': 'linear'}

0.371 (+/-0.184) for {'C': 0.01, 'kernel': 'poly'}

0.559 (+/-0.134) for {'C': 0.05, 'kernel': 'poly'}

0.633 (+/-0.104) for {'C': 0.1, 'kernel': 'poly'}

0.674 (+/-0.046) for {'C': 0.5, 'kernel': 'poly'}

0.650 (+/-0.082) for {'C': 1, 'kernel': 'poly'}

0.602 (+/-0.144) for {'C': 5, 'kernel': 'poly'}

0.603 (+/-0.115) for {'C': 10, 'kernel': 'poly'}

Detailed classification report:

The model is trained on the full training dataset.

The scores are computed on the full testing dataset.

precision recall f1-score support

0 0.75 0.65 0.70 485

1 0.53 0.70 0.61 471

2 0.71 0.74 0.73 559

3 0.66 0.62 0.64 478

4 0.86 0.72 0.79 511

accuracy 0.69 2504

macro avg 0.70 0.69 0.69 2504

weighted avg 0.71 0.69 0.69 2504

The confusion matrix is:

[[315 102 34 12 22]

[ 48 332 26 46 19]

[ 13 47 415 73 11]

[ 13 80 81 295 9]

[ 32 62 25 22 370]]

Best estimator found: SVC(C=0.5, break\_ties=False, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3, gamma='scale', kernel='poly',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

Best parameters set found: {'C': 0.5, 'kernel': 'poly'}

SVM with GridCV on testing data - Accuracy Score: 0.690 (+/- 0.000)

SVM with GridCV on training data - Accuracy Score: 0.793 (+/- 0.000)

Task A1: 0.922 (+/-0.020) for {'C': 0.01, 'kernel': 'linear'}

Task A2: 0.894 (+/-0.019) for {'C': 0.5, 'kernel': 'poly'}

Task B1: 0.840 (+/-0.033) for {'C': 0.01, 'kernel': 'poly'}

Task B2: 0.674 (+/-0.046) for {'C': 0.5, 'kernel': 'poly'}

TA1:0.9338541666666667,0.9347849599900989,0.925;

TA2:0.90078125,0.8972352164748262,0.8947916666666667;

TA1:0.9309176225234619,0.9313864227116373,0.9249217935349322;

TA2:0.9061522419186653,0.9121546994367723,0.8738269030239834;

TB2:0.763117754728493,0.7643026766006599,0.7169005491153142;

TB1:0.8565625,0.8399142263986015,0.80625;

TA1:0.9309176225234619,0.9313864227116373,0.9249217935349322;TA2:0.9061522419186653,0.9121546994367723,0.8738269030239834;TB1:0.8565625,0.8399142263986015,0.80625;TB2:0.763117754728493,0.7643026766006599,0.7169005491153142;

# Abstract

*This section provides a brief overview of the methodology/results presented in the report.*

# Chapter 1: Introduction

*This section introduces the problem, a brief bird’s-eye view of the methodologies you adopted and the organization of this report.*

# Chapter 2: Literature Survey

*This section should focus on an overview of potential approaches to solve the tasks. You can introduce some classical and state-of-the-art machine learning algorithms.*

Support Vector Machines (SVMs) are a type of supervised learning model that aim to

# Chapter 3: Description of Models

*In this section, you should brieﬂy describe the model you are using for each task, along with the rationale. You may opt to use a single learning algorithm to solve the problem or multiple ones, but bear in mind there are page limitations and that you should explain your rationale behind your choices. That is, the algorithmic description must detail your reasons for selecting a particular model.*

*You can clarify them with ﬂow charts, ﬁgures or equations. An example of how to draw an image is demonstrated in Fig. 1.*

## 3.1 Task A1: Gender Recognition

The gender recognition task is a binary classification problem where SVMs can be highly accurate and appropriate for this case.

## 3.2 Task A2: Smiling Recognition

## 3.3 Task B1: Eye Colour Recognition

## 3.4 Task B2: Face Shape Recognition

# Chapter 4: Implementation

*This section must provide the detailed implementation of your models. In particular, you must provide the name and use of external libraries, explain hyper-parameter selection, training pipeline (if any) and key modules/classes/functions/algorithms.*

*You also must provide a detailed description of the dataset (content, size, format, etc.), any data pre-processing that was applied and how you separate your dataset into training, validation and test sets.*

*The execution of your models also should be reported here. In particular, this section should include a thorough discussion on the training convergence and stopping criterion (it is recommended that learning curves graphs be used to this effect).*

**Data pre-processing**

Data pre-processing plays a big role in speeding up computational time and reducing computational complexity when training a ML model especially SVMs. SVMs are sensitive to the feature scales hence it is a good idea to scale the training data

**Regularization by hyper-parameter tuning**

To reduce the risk of overfitting the training data, the hyper-parameters such as the kernel and regularization parameter (C) of Support Vector Machine (SVM) models are tuned with optimal values to ensure that the model will be able to generalize the testing data well in the expense of suffering some degree of penalty in fitting the training data. This is the bias-variance trade-off where we aim to trade-off some the variance to the bias of the model in ensuring that we allow some degree of freedom for the model to simplify itself to generalize testing data well but complex enough to reasonably fit the training data.

By undergoing the exhaustive grid search method that loops through all possible values of regularisation parameter (C), we would be able to determine the best hyper-parameters for the model to perform the best in terms of accuracy and validation scores over testing data.

## 4.1 Task A1: Gender Recognition

### Hyper-parameter tuning

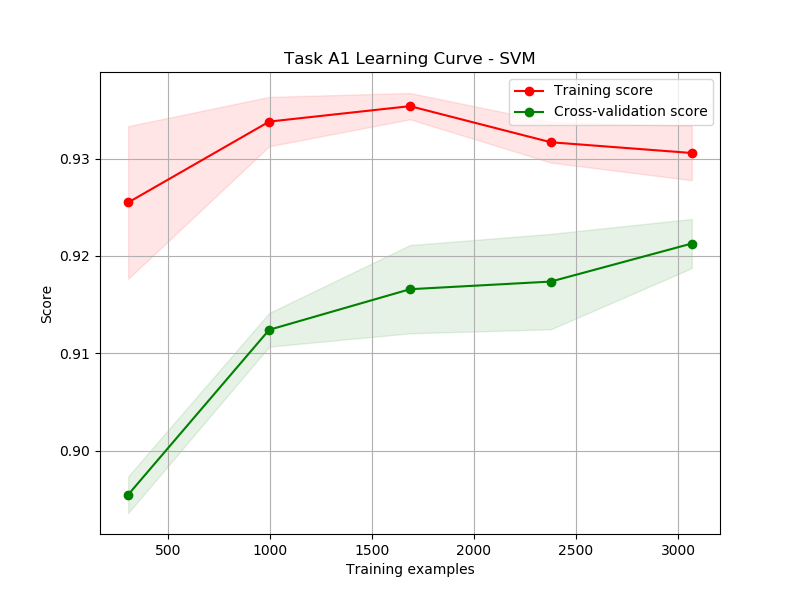
For gender recognition, the Support Vector Machine (SVM) model was used in conjunction with the Exhaustive Grid Search (GridCV) to determine the optimal kernel type and regularisation penalty parameter (C) values. This involves looping through all possible hyper-parameter values and kernel types to determine the best set of parameters for the model to achieve the highest model accuracy score for the problem. It was determined that for this task, the best parameters were the linear kernel and C = 0.01 using facial landmarks data as the training data. This has achieved a test score of 92.5% and a cross-validation score of 91.9%.

This would be the preferred choice because this task is a binary classification problem where a linear kernelized SVM model with linear decision boundary that separates the two classes would suffice in performing well. Furthermore, the small regularization parameter of C = 0.01 produces a wider margin between the decision boundary that separates the two classes and the training instances in the expense of more margin violations – the allowance of certain points to be in the margin. This *soft margin classification* enables the model to generalize better to the testing data hence boosting the accuracy scores on testing data.

Lastly, the model was trained using landmarks instead of RGB as the pre-processed dataset which is a less complex and lower dimensional dataset reducing the risk of overfitting and computational complexity.

Comparing the other grid parameters such as poly and rbf kernels and C values larger than 0.01 to the chosen best parameters, we see that testing accuracy scores decrease by around -1.2% because using other kernels and C values could lead to overﬁtting to the data and poor performances in generalisation.

### Training Pipeline



## 4.2 Task A2: Smiling Recognition

## For smiling recognition, a similar approach to that of Task A1 can be used by utilising SVM model with GridCV search methods to obtain the classiﬁer with the optimal hyper-parameters values for this problem. It was determined that for this task, the best parameters were the polynomial kernel and C = 0.5 using facial landmarks data as the training data. This has achieved a test score of 89.5% and a cross-validation score of 89.3%.

## This would be the preferred choice because a SVM model with a polynomial kernel

## 4.3 Task B1: Eye Colour Recognition

This would be the preferred choice because this is a non-linear classification problem that requires the SVM with a polynomial kernel to classify the complex training data in a higher dimensional feature space.

The training data is comprised of high dimensional pixel information of images. A dataset of high complexity requires kernelized SVMs

## 4.4 Task B2: Face Shape Recognition

*You also must provide a detailed description of the dataset (content, size, format, etc.), any data pre-processing that was applied and how you separate your dataset into training, validation and test sets.*

## Dataset

The aim of this assignment is to train machine learning models and perform binary and multi-class classification on 5000 (JPG) image files from the “CelebFaces Attributes” Dataset (CelebA) and 10000 Portable Network Graphic (PNG) image files from the “Cartoon Set” dataset respectively. The binary and multi-class labels for each image are provided in comma-separated values (CSV) files, in which each column corresponds to different sets of labels for each classification tasks. All images include their respective binary labels and multi-class labels from the CSV labels file provided for each dataset for the following classification tasks. The images are to be classified according to the labels in the CSV files. The classification tasks with their labels are as follows:

For Task A1, the binary classification task is for gender recognition with labels 0 being male and 1 being female.

For Task A2, the binary classification task is smiling recognition with labels 0 being not smiling and 1 being smiling.

For Task B1, the multi-class classification task is eye colour recognition with 5 labels for 5 different colours for eyes.

For Task B2, the multi-class classification task is face shape recognition with 5 labels for 5 different types of face shapes.

In order to train a suitable model for the required classification tasks, several data pre-processing methods were taken into consideration to provide appropriate features for the process.

For classification tasks A1, A2 and B2, the facial landmarks were extracted from each image from the dataset and used to train models using supervised learning algorithms such as Support Vector Machines (SVM). The landmark features provide a set of x and y coordinates that describe specific points of the face and lay out the contours connecting those points as shown in the figure. This was done using the frontal face detector function from the dlib library that uses a Histogram of Oriented Gradients (HOG) feature with a linear classifier.

For classification tasks B1, the images from the “Cartoon Set” dataset were downsampled from 500 x 500 pixels to 128 x 128 pixels and the raw RGB values for each image were read using the OpenCV library. Downsampling was carried out to reduce computational complexity and training times while preserving enough colour information to fairly distinguish each image during training. The RGB data is then finally reshaped into a one-dimensional array before feeding into the ML model.

# Chapter 5: Experimental Results and Analysis

*This section describes and discusses your results. Additionally, this section should include accuracy prediction scores on a separate test dataset, provided by the module organizers, but not used during your training and validation process.*

*We recommend you use a table to list the tasks, models and results before analysis.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Model** | **Train Accuracy** | **Validation Accuracy** | **Test Accuracy** |
| **A1** | SVM – Linear kernel, C = 0.01 | *93.1%* | *93.1%* | *92.5%* |
| **A2** | SVM – Poly kernel, C = 0.5 | *90.6%* | *91.2%* | *87.4%* |
| **B1** | SVM – Poly kernel, C = 0.01 | *85.7%* | *84.0%* | *80.6%* |
| **B2** | SVM – Poly kernel, C = 0.5 | *76.3%* | *76.4%* | *71.7%* |

Task A1: 0.922 (+/-0.020) for {'C': 0.01, 'kernel': 'linear'}

Task A2: 0.894 (+/-0.019) for {'C': 0.5, 'kernel': 'poly'}

Task B1: 0.840 (+/-0.033) for {'C': 0.01, 'kernel': 'poly'}

Task B2: 0.674 (+/-0.046) for {'C': 0.5, 'kernel': 'poly'}

Train/Validation/Test

TA1:0.9309176225234619,0.9313864227116373,0.9249217935349322;

TA2:0.9061522419186653,0.9121546994367723,0.8738269030239834;

TB1:0.8565625,0.8399142263986015,0.80625;

TB2:0.763117754728493,0.7643026766006599,0.7169005491153142;

# Chapter 6: Conclusion

*This last section summarizes the ﬁndings and suggests directions for future improvements.*

**Chapter 5: Support Vector Machines**

**1. What is the fundamental idea behind Support Vector Machines?**

The fundamental idea behind Support Vector Machines is to fit the widest possible “street” between the classes. In other words, the goal is to have the largest possible margin between the decision boundary that separates the two classes and the training instances. When performing soft margin classification, the SVM searches for a compromise between perfectly separating the two classes and having the widest possible street (i.e., a few instances may end up on the street). Another key idea is to use kernels when training on nonlinear datasets.

**2. What is a support vector?**

After training an SVM, a support vector is any instance located on the “street” (see the previous answer), including its border. The decision boundary is entirely determined by the support vectors. Any instance that is not a support vector (i.e., off the street) has no influence whatsoever; you could remove them, add more instances, or move them around, and as long as they stay off the street they won’t affect the decision boundary. Computing the predictions only involves the support vectors, not the whole training set.

**3. Why is it important to scale the inputs when using SVMs?**

SVMs try to fit the largest possible “street” between the classes (see the first answer), so if the training set is not scaled, the SVM will tend to neglect small features (see Figure 5-2).

**4. Can an SVM classifier output a confidence score when it classifies an instance?**

**What about a probability?**

An SVM classifier can output the distance between the test instance and the decision boundary, and you can use this as a confidence score. However, this score cannot be directly converted into an estimation of the class probability. If you set probability=True when creating an SVM in Scikit-Learn, then after training it will calibrate the probabilities using Logistic Regression on the SVM’s scores (trained by an additional five-fold cross-validation on the training data). This will add the predict\_proba() and predict\_log\_proba() methods to the SVM.

**5. Should you use the primal or the dual form of the SVM problem to train a model on a training set with millions of instances and hundreds of features?**

This question applies only to linear SVMs since kernelized can only use the dual form. The computational complexity of the primal form of the SVM problem is proportional to the number of training instances m, while the computational complexity of the dual form is proportional to a number between m 2 and m 3 . So if there are millions of instances, you should definitely use the primal form, because the dual form will be much too slow.

**6. Say you trained an SVM classifier with an RBF kernel. It seems to underfit the training set: should you increase or decrease γ (gamma)? What about C?**

If an SVM classifier trained with an RBF kernel underfits the training set, there might be too much regularization. To decrease it, you need to increase gamma or C (or both).