Problem 1

Advantages of SVM:

- Guarantees optimal separation
- Usually gives better classification performance

Disadvantages of SVM:

- Need to decide on an appropriate kernel function
- Training is slow on really large training sets (SVM training is at least $O\left(n^2\right)$, while neural net training is $O\left(n\right)$ on the size of the training data)

Problems with both:

- Both have several parameters to tune which may have pretty unintuitive effects
- Both are prone to overfitting

Problem 2

A support vector machine kernel is a function that rolls the transformation of two points and subsequent inner product between the transformed points into one. Usually, the transformation maps the points into a higher dimensionality, and the inner product is the scalar dot product $\mathbf{x}\cdot\mathbf{y}=\sum_{i=1}^{n} x_i \cdot y_i$.

This is done mainly because it simplifies calculation, because the kernel in most cases is less computationally intensive than doing the same one step at a time. A convenient byproduct is that it enables transformations that are well-defined, but physically impossible to calculate. The most common kernels are:

- None: $\mathbf{x}\cdot\mathbf{y}$ - simply calculates the dot product, no transformation at all.
- Polynomial: $(1+\mathbf{x}\cdot\mathbf{y})^p$ - transformation into a polynomial space (e.g. $\left[1, x, x^2, x^3, \ldots, x^p\right]$)
- Radial basis function: $\exp\left(-\frac{(\mathbf{x}-\mathbf{y})^2}{2\sigma^2}\right)$ or $\exp\left(-\gamma\left(\mathbf{x}-\mathbf{y}\right)^2\right)$ - transformation into an infinite-dimensional space that has a "smoothness" that is controlled by σ.

Problem 3

An SVM with soft margin allows some of the training data to be misclassified. This is done to avoid over-fitting to the (perhaps faulty) training data, but we would still like to get as much of the training data as possible outside of the margin. This gives us two conflicting factors to worry about: Classifying as much of the training data right and avoiding over-fitting.

Problem 4

What cases where it would be wisest no to classify depends on how critical it is to have high precision: when in doubt, is it best to guess or to refuse to give an opinion.

Examples of cases where a guess should always be made includes cases where a faulty classification is harmless, like recommendations for movies or music. The opposite case would be something like making medical diagnosis, where it always be better to ask a real doctor when there is disagreement in the ensemble.

Problem 5

There are actually two main motivations:

- Make it easier to train a classifier because useless information is removed, or make the classifier simpler and more efficient because of reduced computational complexity.
- Make the data easier to visualize, to manually deduce patterns or to help tune some classifier.

Problem 6

The covariance is

$$\mathrm{cov}_{xy} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y)$$

so first we have to estimate the averages:

$$\mu_x = \frac{1}{n} \sum_{i=1}^{n} x_i = 23.8$$

$$\mu_y = \frac{1}{n} \sum_{i=1}^{n} y_i = 25.8$$

Calculating the covariance by the formula above then gives us $\mathrm{cov}_{xy} = -96.44$. This indicates that there are some negative correlation between x and y, that y decreases when x increases, but to determine how large it is one would have to calculate the variances for both x and y as well. Note that statistically speaking we have "used up" one of the data samples by calculating the averages, so we should really divide by $n - 1$ instead of n when calculating the covariance, giving $\mathrm{cov}_{xy} = -120.55$, which is what most statistics packages will report when you write `cov(x,y)` .