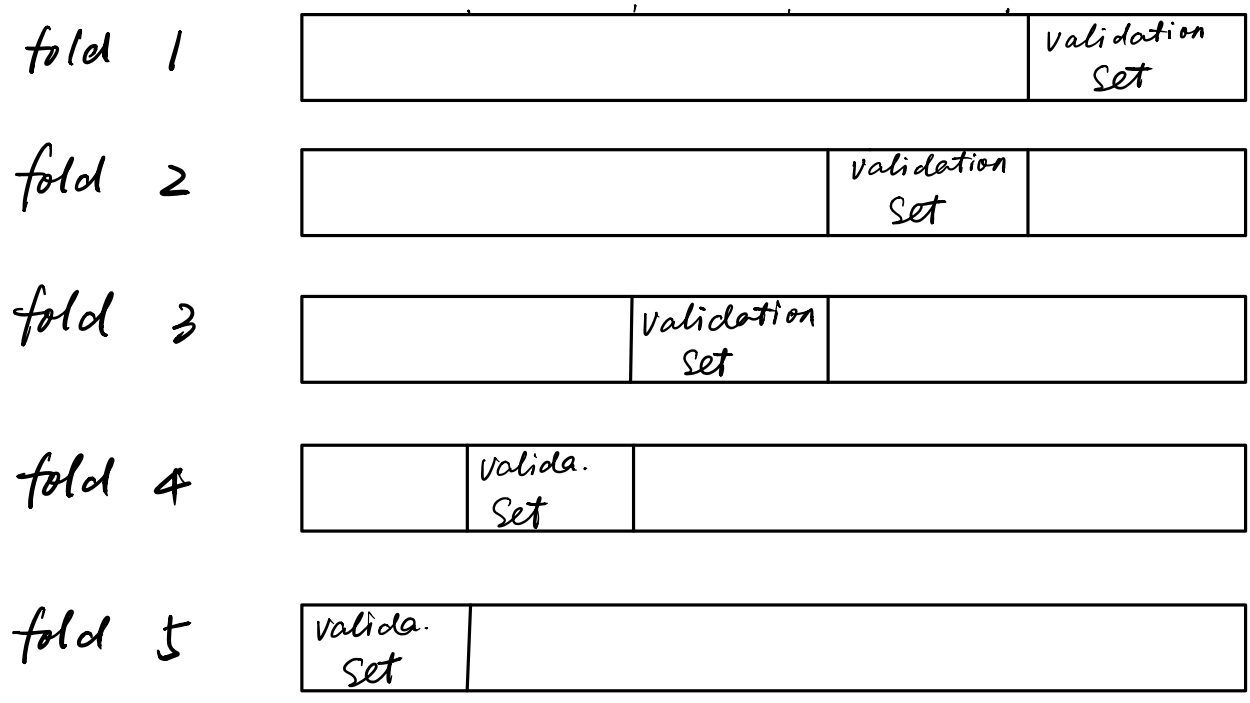
Q1. (a) Linear regression feature $\underline{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$

To construct polynomial model we could add more features that are powers of $\underline{x}$. i.e $\underline{x}' = [x_1, x_2, \dots x_n, x_1^2, x_2^2 \dots x_n^2]$.

And $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n + \theta_{n+1} x_1^2 \dots + \theta_{2n} x_n^2$, model remains linear in the parameter θ_i.

(b) In regression problem, high-dimensional polynomial is not always the good choice. We need to care about the tradeoff between bias and variance. When the degree of polynomial is high, problem of overfitting occurs. That is training data would fit model well and training error (bias) is small. But when model is applied to test data, the testing error (variance) would be large.

(c) Through technic called k-folded Cross Validation which split data into training, validation and test sets.



1. we split the data in 5 parts (of roughly equal sizes)
2. repeatedly train on 4 parts and test on the part "left out". i.e validation set.

3. average the errors of 5 validation sets to give so-called cross-validation error.

4. smaller k is less expensive but poorer estimate as size of training set is smaller and random fluctuations larger.

(d) MSE : $E[(y-h(x))^2] = Bias^2[h(x)] + Var[h(x)] + \sigma^2$

There are 3 components:

1. Bias : $Bias[h(x)] = E[h(x) - f(x)]$
2. Variance : $Var[h(x)] = E[h^2(x)] - E[h(x)]^2$.
3. irreducible error σ^2 due to noise in training data.

(e) We now additionally assume there exists some underlying function F such that

$$y = F(x) + \varepsilon$$

where ε is noise, i.e $E[\varepsilon] = 0$ and finite variance.

Thus the optimal prediction is $f^*(x) := E[y|x] = F(x)$ with square loss.

Our goal will be to understand the expected error at x

$$\mathcal{E}(h(x)) = E[(y - h(x))^2]$$

where y is a sample from the marginal $P(y|x)$

$$E[(y - h(x))^2] = E[y^2 + h(x)^2 - 2yh(x)]$$

$$= E[y^2] + E[h(x)^2] - 2E[y]E[h(x)]$$

next step is to find each component seperately.

From the formula of variance we have:

$$E[h(x)^2] = E[(h(x) - E[h(x)])^2] + E[h(x)]^2$$

Since $y = F(x) + \varepsilon$, then we can get:

$$E[y] = f^*(x)$$

$$E[y^2] = E[(y - f^*(x))^2] + f^*(x)^2$$

Thus,

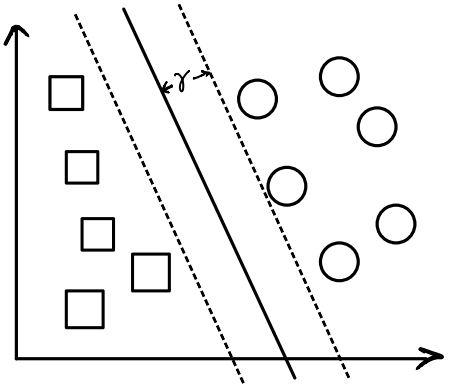
$$E[(y - h(x))^2] = E[y^2] + E[h(x)^2] - 2E[y]E[h(x)]$$

$$= E[(y - f^*(x))^2] + f^*(x)^2 + E[(h(x) - E[h(x)])^2] + (E[h(x)])^2 - 2f^*(x)E[h(x)]$$

$$= E[(y - f^*(x))^2] + (f^*(x) - E[h(x)])^2 + E[h(x) - E[h(x)]^2]$$

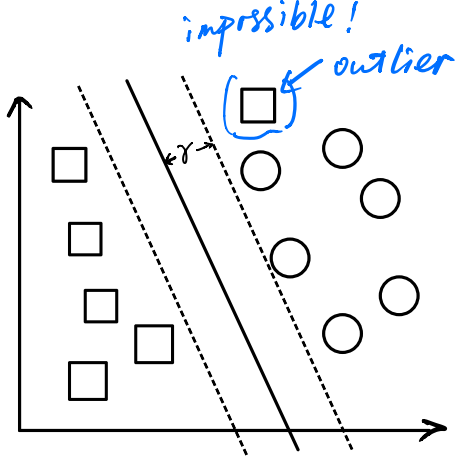
$$= Bias^2[h(x)] + Var[h(x)] + \sigma^2$$

Q2. (a) Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new instances $\underline{x}_i$ to one category or the other. SVM maps training examples to points in hyperplane so as to maximise the margins γ between 2 categories. New instance are then mapped into that same hyperplane and predicted to belong to a category based on which side of gap they fall.



Hard margins: (linear problem)

If we strictly impose that all instances to be classified on the right side. this is called hard margin classification. There are 2 main issues with hard margin classification. First, it only works if the data is linearly separable, and second it's quite sensitive to outliers.



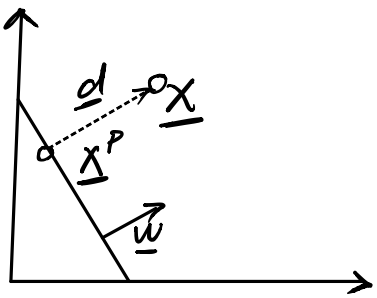
Soft margin: (non-linear problem)

To avoid these issues it's preferable to use a more flexible model. The objective is to find a good balance between keeping margin γ as large as possible and limiting the margine violations (i.e. instances that end up in the middle of model or even on the wrong side).

This is called soft margin classification. We can control this balance using the C hyperparameter: a smaller C values leads to a wider margin but more margin violations; vice versa for large C.

(b) SVMs are particularly well suited for classification of complex but small or medium sized datasets.

(c) Define a hyperplane $H = \{\underline{x} \mid \underline{w}^T\underline{x} + b = 0\}$. Let the distance m be defined as the distance from the hyperplane to the closest point across both classes.



Consider some point $\underline{x}$. Let $\underline{d}$ be the vector from H to $\underline{x}$ of minimum length. Let $\underline{x}^p$ be the projection of $\underline{x}$ onto H. It follows that:

$$\underline{x}^p = \underline{x} - \underline{d}$$

$\underline{d}$ is parallel to $\underline{w}$, so $\underline{d} = \alpha \underline{w}$ for some $\alpha \in \mathbb{R}$.

$\underline{x}^p \in H$ which implies $\underline{w}^T\underline{x}^p + b = 0$

therefore

$$\underline{w}^T\underline{x}^p + b = \underline{w}^T(\underline{x} - \underline{d}) + b = \underline{w}^T(\underline{x} - \alpha\underline{w}) + b = 0$$

which implies $\alpha = \frac{\underline{w}^T\underline{x} + b}{\underline{w}^T\underline{w}}$

The length of $\underline{d}$:

$$\|d\|_2 = \sqrt{\underline{d}^T \underline{d}} = \sqrt{\alpha^2 \underline{w}^T \underline{w}} = \frac{|\underline{w}^T \underline{x} + b|}{\sqrt{\underline{w}^T \underline{w}}} = \frac{|\underline{w}^T \underline{x} + b|}{\|w\|_2}$$

distance m with respect to D: $m(\underline{w}, b) = \min_{\underline{x} \in D} \frac{|\underline{w}^T \underline{x} + b|}{\|w\|_2}$

as we also defined $|\underline{w}^T \underline{x} + b| = 1$, so the distance m is:

$$m(\underline{w}, b) = \frac{1}{\|w\|_2}$$

therefore margin $\gamma(\underline{w}, b) = 2m(\underline{w}, b) = \frac{2}{\|w\|_2}$.

Q3. (a) Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. And Max Pooling calculate the maximum value for each patch of the feature map. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

Goal of pooling layer is to subsample (i.e shrink) the input image in order to reduce the computational load, the memory usage, and the number of parameters (thereby limiting the risk of overfitting)

(b) The distance between two consecutive receptive field is called the stride.

(c) padding = "VALID": without zero padding and ignore some rows and columns at the bottom and right of the input image depending on the stride.

padding = "SAME": using zero padding to force layers to have same height and width. More specificly, the number of output neurons is equal to the number of input neurons divided by the stride, rounded up.

(d) (i)

9

(ii)

6	9
8	9

(iii)

5	7	9
6	8	9
6	9	8

(iv)

6	7	8	9
7	8	9	9
8	9	9	9
8	9	9	9

(v)

8

(vi)

6	9
8	9