Week 7

Why is higher dimensions good?

A complex patter-classification problem, cast in a high-dimensional space non-linearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated.

Model complexity

Sometimes, the models become too complex

Overfitting You model is too specialized on you training data. It works great on training, but not so much in test data

Underfitting You model is too bad in general. The minimal error is too great for both training data and testing data.

Cross-validation

We don't want our model to be trained too specifically on the training data. What we really care for is the ability of our model to predict for new data. To achieve this, we use **cross-validation**, a way to measure how our model performs for new data.

This is a way to test if our method is overfitting or underfitting, not any actual training algorithms

Test validation set method

1. Randomly choose 30% of the data to form a validation set
2. The remainder then acts as your new training set
3. Perform regression on this new training set only
4. Estimate the performance on the test data using the validation set

Leave one out cross validation (LOOCV)

1. Do, for every $i = 1 \dots N$
    1. Let (x_i, y_i) be the i^{th} sample
    2. Temporarily remove (x_i, y_i) from the training set
    3. Perform the regression on the remaining $N - 1$ samples
    4. Compute the error on (x_i, y_i)
2. Report the average error

Validation set vs Leave-one-out

	Downside	Upside
Validation set	Unreliable estimate	Cheap
Leave-one-out	Expensive	Doesn't waste data

We would like a mix of both

k-fold cross validation

1. Randomly split the dataset into k partitions
2. For every partition P, do
    1. Train the model on the points not in partition P
    2. Compute the error on the points of P
3. Report the average error

The higher the k, the more expensive the computations become, but the more reliable they become.

Uses of cross-validation Cross validation let's you measure how good your model performs given new data. This can be useful to: * Determine between similar models which one is the best * Find the best value for one hyperparameter

Penalizing overfitting

We add a regularizer to the empirical risk, which leads to a new training objective of the form

$$E(w) = R(w) + \lambda E_w(w)$$

where:

- $R(w)$ is the empirical risk
- $E_w(w)$ is the regularizer
- λ is a hyperparameter which defines the strength of the regularizer

Regularized linear regression We use the sum of squares of the weights regularizer

$$E_w(w) = w^T w$$

Then the regularized linear regression is

$$\min_w \sum_{i=1}^{N} (\phi(\overline{x}_i)^T w - y_i)^2 + \lambda w^T w$$

which remains a convex function.

The gradient is now $\nabla E = \nabla R + \nabla E_w$, which when setting to 0 gives

$$\left(\sum_{i=1}^{N} \phi(\overline{x}_i)\phi(\overline{x}_i)^T + \lambda I_F \right) w^* = \sum_{i=1}^{N} \phi(\overline{x}_i) y_i$$

which let's use analytically compute w^* as $(\Phi^T\Phi + \lambda I_F)^{-1}\Phi^T y$

For multioutput, the results are the same, but using the Frobenius norm squared over a matrix when needed, instead of just squaring the results. We define the Frobenius norm squared as

$$\|(a_{ij})_{i\in\mathbf{N}}^{j\in\mathbf{M}}\|_F^2 = \sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}^2$$

Regularized logistic regression We can use the sum of squares as well to the cross-entropy loss in logistic regression while maintaining a convex function.

Regularized SVM Because the SVM problem is expressed as

$$\underset{w,\{\xi_i\}}{min}\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i$$

subject to $y_i \cdot (w^T\overline{x}_i) \geq 1 - \xi_i$, $\forall i$, we can see that it's already regularized by the margin maximization term, so we don't need to to change anything. However, note that it does not involve $w^{(0)}$ (This can also be translated to the other cases, actually)

markdown