**Other notes**

a classifier should probably also just give back a probability value not a final solution to the classification

gradient descend helps us to find the loss for regression and linear models better because instead of making hella unnecessary models we take the derivative and mode downwards strictly and repeat till we arrive at the minimum we find the direction in which the loss decreses the most just calulating the min setting the derivitave of the loss function to zero like we did in highschool stops working with more complicated models sadly so we have to do this iterativly

The linear classifier draw a line in the feature space and everything above that is one class everything below is the other class this cutting in half can be done even if the feature space has a lot more dimensions to find which line would be the best to draw we put all possible lines in another feature space with a b and c and check our loss function to see which works best in this case. So baisically we try all lines and see which of them has the lowest loss function

Tree classifier for each feature we have a threshold for which we make a descicion in the example we first check weather or not the instance is taller than 170 and then if the person is smaller that would have to have broader shoulders for us to actually assume they are male because they are shorter and you can model this decision tree to more features as well of course

K nearest neighbor very intuitively checks an odd number of the nearest neighbors (say seven) and then makes a decision to classify based on that result (if 4 or more of the neighbours are in class a we will classiy the current model as a)

Regression the output of a regression model is not a class but a number so instead of drawing a line through the feature space to show the border between the classes we are trying to make a line that best describes the (hopefully somewhat linear) relation between the input and output space a loss function for this would be the mean squared error <- this was linear regression there is also knn regression and tree regression models. Knn predicts now does not output a class but a number by using the average values of the nearest neighbours. For linear regression finding the perfect line is difficult and we try to do it by maybe using a weight vecotor and an offset and then we use random search on the losss model space to find the weights and that have the lowest loss

K means is a clustering algorithm that first takes three random points in the feature space and then colour every point according to the nearest mean. Then we recompute the actual mean of all three classes and use those as the new true means in our next iteration of the algorithm

**Lecture 1**

Some about the first work sheet and the first homework

Deductive vs inductive reasoning like the sun risen in the east every day of my life so it will do so again tomorrow is inductive

Machine learning uses inductive reasoning

Machine learning only provides approximate solutions so do not use it if we canno make a choice

Abstract tasks of the actual problems: classification and regression for example

Supervised has input and output as training data and in unsupervised learning only input is provided

Linear classifier with a hyperplane ( drawing a line between the data in a higher dimension plane)

Loss function tells us how good the algorithm is

Decision tree classifier start at the root and then on each node make a decision based on a single feature (height bigger or smaller than 160)

K nearest neighbors pick the k number of nearest neighbours in the data set and then make a decision based on the number of neighbours in a class

Binary classification or multiclass classification and multilabel classification

Class propability gives the probabitly of each class

Regression has as output a number not a class

Mean squared error loss I think I remember

We square to make sure negative loss Is just added and does not cancel out and to penalize the higher error but squaring it

Talking about overfitting never judge you model on the training data

Unsupervised learning like clustering

Density estimation outputs a probability of your instance to occure based on training data

Generative modelling

**Lecture 2**

Linear models and search is the theme

Start with data set which gives instances each instance has some features than you feed that to a learner which then outputs a model that can predict stuff

Talking about y axis intercept and the coefficient the slope in a linear function

For one feature F(x ) = wx +b

For two features F(x1, x2) = w1x1 +w2x2 +b

For multiple features an instance is a vectore and a weight vectore and then you take the dot product

Reminder dot product is multiplying wi \* xi for each I and then you add all the results getting to a dsingle number you can als do the dot product by taking the length of the x vector times the length of the w vector times the cosin of the alpha between them

Reminder the x vector the feature vectore is data and the weight vector means how important the feature is for the overall evaluation

Talked about regression

Random search loop pick a random point if the model is bettermove to that point otherwise find a different point

Local vs global minimum in the loss hyperplane

Model space has b as y axis and w as x axis and the rest is

We can also do the search in parallel on the loss plane

Can use partical swarm and evolutionary computing algorithms

Population methods are very powerfull and are still black blox you do not need to know anything about the data

To escape local minum we add reandomness add multiple models

And to converge faster we combine known good models like breeding

Using calulus we can find the direction in which the loss drops most quickly

This direction is the opposite to the gradient

To find the slope of the tangent line take the deriitave of the function and plug in the point you are trying to find the slope of

For more dimension the tangent becomes a tangent hyperplane

You construct that tangent hyperplane by taking a tangent in each axis direction

You end up with a vecotor of the coefficients of the tangents building the tangent hyperplane

The direction of steepest ascent is the gradient so we take the minus to do the direction of the steepest descent

Sometimes the loss function should not be the same as the evaluation function

**Lecture 3**

Evaluation of the model

Binary classification with positive and negative and testing for the positive

Error is how many you classified correctly: accuracy, how many you misclassified is error

On which data do we compute the error

Because of overfitting never judge your model on the performance of the training data

We should at least have 500 of the test data

You use a training and validiation set and a testing set so you train on the training test and validate which method to use on the validation set then retrain on the training and validation set and then testing on the testing set

Als cross validation as before with k folds

If you have temporal sensitive data you should make sure to test only on data that comes exclusively after the training data in time

Which hyperparamters to use ? trial and error to find out (or grid search or random search)

Grid search is not always the best option ( look at the slides)

Class balance is important because if we have 80% positive and our model always says positive our model is 80% accurate

Cost ombalance also important is your alpha or beta error worse does it matter

Make sure your class instanes is the same for test and training set in total number if you have large class imbalance also make sure that the training method has to be bigger if you have class imbalance

So use more performacce metrics

Confusion matrix Is the thing with the true positive and false positive and so on in a grid

Precision is true positive over true and false positives and recall is true positives over true positives and false negatives

Building a ranking classifier ranks instances from most negative to most positive ( distance to the decision boundry

Ranking error is explained on the slides boi

Now talking about regression

Loss function mean squared errors or root mean squared error

There is a measured MSE and an optimal MSE

Distance from the true MSE to the optimal MSE is the bias and distance form the true MSE to the measured MSE

High bias model doesn’t fir the generating distribution also called underfitting

High variance high model capacity sensitivity to random fluctuations aka overfitting

Reducing bias increase model capacity increase features

Reducing variance reduce model capacity add regulazation reduce tree depth

For kn neighbours increase k to increase bias decrease variance

Week 5 we will look at ensembling

Now talking about statistics

Error bars could mean anything so you have to tell the readers what it is like the standard error

No free lunch theorem we can always find a dataset that fucks everything we did before so there is no perfect machine learning model

**Rereading lectures**

Dealing with missing data and labels and dealing with classimbalance and outliers

We make the feature space more dimensional to make our classifiers better so instead of a line we might have a parable as our line for regressional models we do this by adding features derived from the first feature

**Lecture 4**

Probability basics and then Bayesian classifier

Logistical regression and information theory

Objective probability the probability for x in a demographic of y

One form of this is frequentism which is under the assumption that we repeat the experiment and then the probability refers to the chances that in this repeat expitriment we get the outcome we specified

We should also note that probability is not perfect we do expiriments to estimate the actual probability of an event

Logistical loss

Find the classifier q that maximises the probability of the true classes

**Some lecture idk I lost fucking count**

Deep learning and generative modeling

Transfer learning use a preexisting model and then retrain only the last couple of layers on your own data

Deep learning for generative models generative because it it supposed to generate random data like a face we use any feedforward neural network.

Using a generator neural network

Turning a neural network into a probability distribution

Option one is turning the output values into the parameters of a probability distribution

Still on how to turn a NN into a probability distribution

Mode collapse

So one idea would be to generate data and then check with a random instance how close we where to this random instance and then backpropagate this causes mode collapse doe dawg

Adversarial example given a network you can create a picture that fools the model

Use the adversarial example and then retrain your model with that

Repeat that process and the adversarial examples are getting closer and closer