$$\frac{1}{2}\sum_i^N (x_i \cdot w - t_i)^2 \sim \|Xw - t\|^2 = \sum_i (x_i w - t_i)^2$$

L2: $$\frac{1}{2}\sum_i^N (y(x_i, w) - t_i)^2 - \underline{\frac{\lambda}{2}\|w\|^2}$$ L2 norm.

$$\hat{\mu} = \frac{1}{N}\sum_i x_i \qquad \hat{\sigma}^2 = \frac{1}{N}\sum_i (x_i - \hat{\mu})^2 \rightsquigarrow \mathbb{E}[\hat{\sigma}^2] = \left(1 - \frac{1}{n}\right)\cdot\sigma^2$$

↳ this is not unbiased, but with increasing N converges to

batch SGD: $$\nabla_w E(w) \approx \frac{1}{B}\sum_i^B \nabla_w L(y(x_i, w), t_i)$$ ⤳ for randomly chosen data

Perceptron class.:

$y = x^T w$

if $t \cdot y < 0$ ~ when wrong class

$w = w + tx$ ~ move towards the right

Categorical: $$\sum_i^n p_i = 1 \qquad p(x) = \prod_i^n p_i^x$$

$\mathbb{E}[x_i] = p_i$

$var(x) = p_i(1 - p_i)$

$$I(x) = -\log P(x) = \log \frac{1}{P(x)}$$

$$H(x) = \mathbb{E}[I(x)] = -\sum_i p(x_i) \cdot \log p(x_i)$$

$$H(P, Q) = \mathbb{E}_{x \sim p}[-\log Q(x)]$$

$$H(P, Q) \geq H(P)$$ ~ I can only add the level of surprise when going over another distribution

$$D_{KL}(P \| Q) = H(P, Q) - H(P) \qquad D_{KL}(P \| Q) \neq D_{KL}(Q \| P)$$

sigmoid : $\frac{1}{1+e^{-x}}$

softmax: $\frac{e^{z_i}}{\sum_j e^{z_j}}$

log. regr.: $p(C_1/x) = \sigma(x^T w + b)$

$p(C_0/x) = 1 - p(C_1|x)$

$\bar{y}(x,w) = \log\left(\frac{p(C_1/x)}{p(C_0/x)}\right)$

– the ratio between these two, called logit

Universal approximation theorem ~ I can map any $\mathbb{R} \to \mathbb{R}$ function into a set of ReLU gates (therefore MLP) if I have enough space...

precision $= \frac{TP}{TP+FP}$ recall $= \frac{TP}{TP+FN}$

f_β-score $= \frac{1+\beta^2}{\text{precision}^{-1} + \beta^2 \cdot \text{recall}^{-1}}$ $\overline{}$? $= \frac{TP + \beta^2 \cdot TP}{(TP+FP) + \beta^2 \cdot (TP+FN)}$

$\rightsquigarrow$ the β a „weight" how much we care about recall more than on precision

micro/macro F score

- micro first counts all
- macro counts individual and avgs.

k-nearest:

$$t = \sum_i \frac{w_i}{\sum_j w_j} \cdot t_i$$ $\rightsquigarrow$ for k nearest points

$\rightsquigarrow$ this could be a vector over multiple classes and we do argmax over it

kernels: $k(x,z) := \varphi(x)^T \varphi(z)$ $\rightsquigarrow$ but without explicitely computing $\varphi(x)$

homogeneous poly. ker:

$$k(x,z) = (\gamma \cdot x^T z)^d \sim x_1^2, x_1x_2, x_2^2 \quad (\text{for } d=2)$$

RBF: $k(x,z) = e^{-\gamma \cdot \|x-z\|^2}$ ~ like a soft k-nearest neighbour

– thanks to Taylor expansion considers all dimensions

↑ REVISE THIS ↑

$$e^{-\|x-z\|^2} = e^{-\|x\|^2 + 2x^Tz - \|z\|^2} = \sum_{i=0}^{\infty} \frac{(2x^Tz)^d}{d!} \cdot e^{-\|x\|^2 - \|z\|^2}$$

Dual (kernel) formulation:

$$w = \sum_i^N \beta_i \varphi(x_i) \quad , \quad y(z) = \sum_i^N \beta_i k(z, x_i) + b$$

↳ linear combination of all values

↳ we compute k_{ij} and train with SGD

SVM:

$$\underset{w,b}{\arg\min} \frac{1}{2}\|w\|^2 , \quad \text{constrainted to} \quad t_i y(x_i) \geq 1$$

$$d_i = \frac{t_i y(x_i)}{\|w\|}$$

↳ this we want to maximize

↳ minimizing $\|w\|^2$

$$L = \frac{1}{2}\|w\|^2 - \sum_i \underline{a_i} (t_i y(x_i) - 1)$$

$$w = \sum_i a_i t_i \varphi(x_i)$$

$$0 = \sum_i a_i t_i$$

KKT: values strictly on one side have $a_i = 0$

$$\Rightarrow a_i \cdot (t_i y(x_i) - 1) = 0$$

$$L = \sum_i a_i - \frac{1}{2} \sum_i \sum_j a_i a_j t_i t_j k(x_i, x_j)$$ → this we want to maximize

perception would never find optimal separating hyperplane.

- support vectors: boundary is a subset of closest data-points

Soft SVM:

$$L = \frac{1}{2}\|w\|^2 + C \cdot \sum_i \xi_i - \sum_i a_i (t_i y(x_i) - 1 + \xi_i) - \sum_i b_i \xi_i$$ → minimizing

$$\sum_i \xi_i \geq 0 \quad , \quad b_i = C - a_i$$

the softening correction.

$$\xi_i = \begin{cases} 0 & t_i y(x_i) \geq 1 \\ |y(x_i) - t_i| & \text{otherwise} \end{cases}$$

↳ how much I am offset

Multiclass SVM:

→ one-versus-rest: k classifiers, the strongest voice wins

→ one-versus-one: $\binom{k}{2}$ classifiers, majority votes class wins

TF-IDF:

$$TF := \frac{\# t_i \text{ in } d}{\sum_i \# t_i \text{ in } d} \qquad IDF := \log \frac{\# d}{\# d \text{ with } t}$$

$x = TF \cdot IDF$ ~ reflects how important the document is.

Mutual Information

$$I(X,Y) = H(Y) - H(Y|X)$$
$$= E[-\log P(y)] - E[-\log P(x/y)]$$
$$= E\left[-\log \frac{P(x,y)}{P(x)\cdot P(y)}\right]$$ $\rightsquigarrow$ how much I see x,y together more/less than individually

Covariance

$$Cov(X,Y) = E\left[(X - E[X])\cdot(Y - E[Y])\right]$$
$$= E[XY] - E[X]\cdot E[Y]$$

ρ corr. coef.

$$\rho = \frac{cov(X,Y)}{\sqrt{var(X)}\cdot\sqrt{var(Y)}}$$

Ensembling

they are uncorrelated

$$E\left[\left(\frac{1}{M}\sum_i \varepsilon_i(x)\right)^2\right] = E\left[\frac{1}{M^2}\cdot\sum_{i,j}\varepsilon_i(x)\varepsilon_j(x)\right] = \underline{\frac{1}{M}}\cdot E\left[\frac{1}{M}\cdot\sum_i \varepsilon_i^2(x)\right]$$

so I have lowered the expected MSE by factor $\frac{1}{M}$

Bagging

- bootstrap aggregation
- every model receives different dataset, randomly drawn with repetition

Decision trees

minimizing $C_{T_L} + C_{T_R} - C_T$ for splitting node T to T_L and T_R.

$$C_T := \sum_i (t_i - \hat{t}_T)^2, \quad \hat{t}_T = \frac{1}{|T|}\sum_i t_i$$

- usually we also incorporate some rules to manage the tree shape
- can also prune the tree after training, will increase loss, might decrease costs radically
- Gini: $|T| \cdot \sum_k p_T(k)\cdot(1 - p_T(k))$ Entropy: $|T|\cdot H(p_T)$

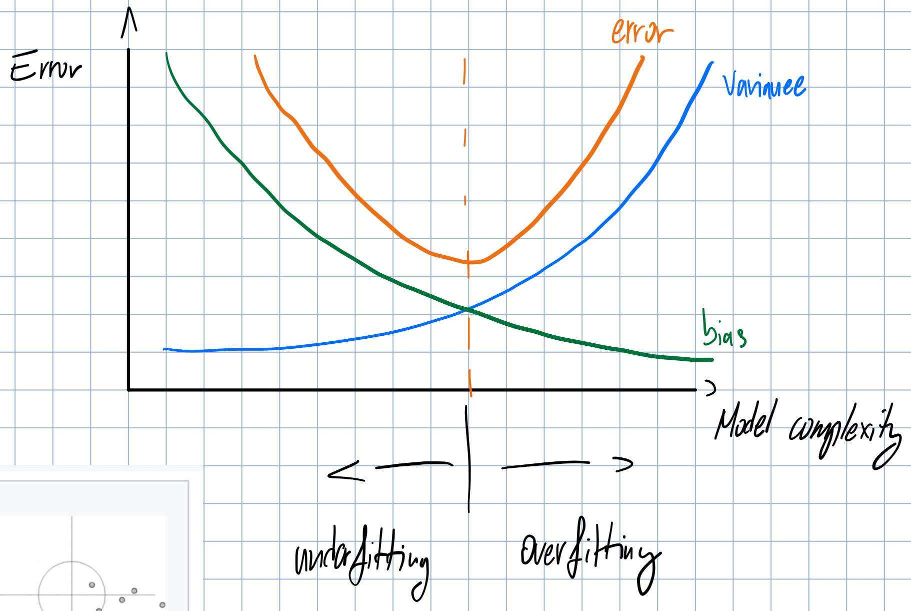
Bias - Variance tradeoff

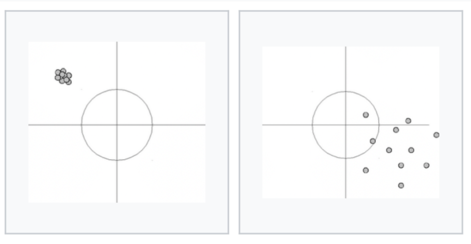
$$MSE = \sum_i (\hat{y}(x_i) - y_i)^2 = BIAS[\hat{f}(x)] + VARIANCE[\hat{f}(x)] + \sigma^2$$

Bias $\rightarrow$ Its the different between the prediction and real target of the model.

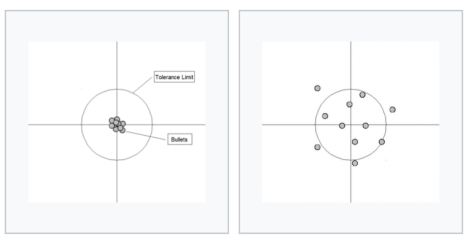
Variance $\rightarrow$ Represent the amount of change of the model output when we change the training set. The higher variance, the more we learn spec. features of the training dataset.

σ^2 $\rightarrow$ it is the existing real noise of the data we cant get rid of.




High bias, low variance

High bias, high variance



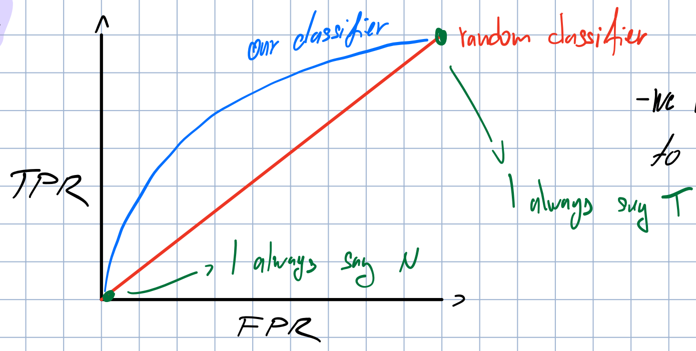
Low bias, low variance

Low bias, high variance

For k-NN, low k ~ low bias, high variance
high k ~ high bias, low variance

The higher k, the more we average over the sample and the lower we get the variance. However, the higher gets the bias because it is harder to catch local features.

ROC -AUC



- We want to get the curve as close to the left-top corner as possible.

$$\text{Recall} = \frac{TP}{TP+FN} \qquad F_\beta\text{-score} := \frac{(1+\beta^2)}{(\beta^2 \cdot \text{recall}^{-1}) + \text{precision}^{-1}} = \frac{(1+\beta^2)\cdot TP}{(1+\beta^2)\cdot TP + \beta^2 \cdot FN + FP}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

Gradient Boosted Trees

- we train trees sequentially, every next tree tries to fix values of the previous
- we could use second-order derivatives (we don't need to set learning rates anymore)
  - but there are too many and its expensive

Trees vs. MLP

- it makes sense to use Tree where each feature has its clear impact on output
- —||— MLP for data where features are unstructured.

PCA

- dim reduction, feature extract.

Maximum Variance Form

- the higher the variance in a dim, the more there is information encoded
  - -> I want to remove those dimensions with low variance
- they correspond to the $M < N$ highest eigenvectors of the covariance matrix S.

Minimum Error Form.

- we have basis vectors, projection of data x into the ortogonal basis and we choose such vectors from basis that we minimize the projecting loss.
- we end up with $M < N$ eigenvectors representing the subspace.

K-Means clustering

- first initialize means of clusters

1) find closest cluster for each input
2) move cluster center towards mean of the cluster

EM / Gaussian mixture

- we suppose data are normally distributed
- we try to map k distributions for k clusters.
- we use EM alg. for evaluating the distributions and maximizing the prob. for correct sample classification

Hypothesis testing

p-value: prob. of obtaining test statistics at least as extreme as the one observed, assuming null-hypothesis.

1) Formulate H_0, H_1
2) Choose test statistics
3) Compute test statistics
4) Calculate p-value
5) Reject H_0 if p-value below α

→ it only shows when the H_0 really doesn't hold!

My solution:

Why MLP performed poorly:

- I merged salaries with one-hot, therefore gradient was struggling to flow through the 0-1 features compared to 20 000 salary.
- I wrote „sharp boundaries", which is misleading and was meant to support the decision trees, which are scale free.

- I should have used MinMaxScaler or StandardScaler

Missing text processing:

- Skipped because I wanted to first come up with any baseline.
- It carries the most important information
- Instead, we can incorporate TF-IDF as the basic measure.
  - take all words from title, remove stop words, remove extreme counts
    - generate bag of words
    - I could also introduce lemmatization/stemming to decrease bag size
  - there is tdidf vektorizer in sklearn
- We could use Word2Vec or another tokenizer (even pretrained)
- We could finetune small language model (BERT) to make latents about the title.

Missing Salary

- again for Trees it doesn't matter, for MLP it does matter!
- instead I should have modelled it with another variable.

Models used:

Random Forests

- Builds a set of full decision trees with bagging (subsets with replacement)
- When doing splitting, the tree is looking only at certain (random) num of features
- Was using only small number of features for split (1), therefore it couldn't perform well.

AdaBoost

- Building classifier from many weak learners (trees of depth 1) sequentially
- Each stump tries to classify based on a feature,
  is evaluated and gets $\alpha = \frac{1}{2} \ln\left(\frac{1-\text{error}}{\text{error}}\right)$ error ~ sum of weights of misclassified.
- We then decrease weights for correctly classified, increase weight for misclassified.
  - the model is then forced to look for hard cases in next round.
- Finally I make decision as a weighted vote of all weak learners with weight α_t.

Why it worked so well?

- Technically it works as a feature selector (each stump)
- Each stump is scale-invariant
- AdaBoost will also converge to try solving the „hard cases" and
  therefore might have better final score.

What might be its weakness?

- It is sensitive to noise and outliers, as it more and more tries
  to solve primarily the difficult items.

Gaussian Process Classifier

- inputs close in feature space should have same labels
- it uses kernel functions (like RBF) to define
  the feature space and the distance between them
- it is $O(N^3)$ in time, therefore it would be demanding to run
  on big dataset.
- the kernel function works as a covariance metric
- we can split the domain-space into various subregions.

Things I didn't know:

- given LogisticRegression and MLP classifier, what are the losses?
- can you describe Adam and RMSProp
- gradient split for multihead solution
- given L1 and L2, what can be considered a feature selection? —> L1

Another things:

- Bias-Variance tradeoff
- Overfitting / underfitting
- Some general questions between classification and regression
- When having both regression and classification head from same backbone, how do you propagate gradient?

- explain mini-batch, batch and stochastic SGD

I would be so happy to get any offer from expedia.
Additionally, landing the Machine Learning Scientist II job
would be fantastic as it would quickly move me to super
position. On the other hand, the Machine Learning Scientist Graduate
will allow me to explore what ML Scientist work looks like
at slower pace with more time around.
Both possibilities are perfect!
Please please I would be so happy if this goes well.

Losses

Gradient

Logistic regression

$NLL \sim \frac{1}{N} \cdot \sum_i - \log(p(C_{t_i} | x_i, w))$ $\qquad (y(x) - t) \cdot x$

$\bar{y}(x_i; W) = \log\left(\frac{p(C_1|x_i)}{p(C_0|x_i)}\right)$

Multiclass logistic regr

$NLL \sim \frac{1}{N} \cdot \sum_i - \log(p(C_{t_i} | x_i, w))$ $\qquad (y(x) - 1_t) x$

– considering that $\bar{y}(x_i; W) = \log\left(\frac{p(C_{t_i}|x_i)}{p(C_0|x_i)}\right)$

$\hookrightarrow \frac{p(C_{t_i}|x_i)}{p(C_0|x_i)} = e^{z_i}$

Linear regression

$MSE \sim \frac{1}{2} \mathbb{E}(y(x) - t)^2$ $\qquad (y(x) - t) \cdot x$

– in fact also NLL when considering we predict normally dist. data

Additional losses

– Huber loss

– MSE when small error, MAE when large error

$\longrightarrow$ it does not explode with large outliers

– Hinge loss:

$L = \max(0, 1 - y \cdot \hat{y})$

– when correct classification beyond safety boundary (margin > 1), gradient is always zero.

– used in SVM

RMSProp

$\sim\to$ mean of the second order

$$V_t = \beta \cdot V_{t-1} + (1-\beta) \cdot g_t^2$$

$\rightsquigarrow$ β usually $\sim 0{,}9$

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{g_t}{\sqrt{V_t + \epsilon}}$$

$\to$ values updated by normalized running avg. of the gradient

Adam

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) \cdot g_t \qquad \hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$V_t = \beta_2 \cdot V_{t-1} + (1-\beta_2) \cdot g_t^2 \qquad \hat{V}_t = \frac{V_t}{1-\beta_2^t}$$

$\to$ fast convergence

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{V}_t + \epsilon}}$$

I keep updating gradient momentum and only update gradient by the updated momentum, considering previous steps.

Hypothesis testing:

We will measure $P(\text{data} \mid H_0)$, H_0 = „no diff. between model A and B"

My projects

- market value price change indicator : LSTM + Regressor (large training dataset, pytorch)
- train driver navigation (large geospatial data, python)
- Market Value Server (biggest software written)

Dissertation

- Iterative space exploration with LLM-aided decision making
  - „how to make LLMs more accurate in picking from list of options?"
    - $\to$ right balance between context size and information distr.
- LLM uncertainty / reasoning
  - „how to measure the certainty of LLM output?", „how to more effectively think?"

Please I hope Anne Morman will be fine tomorrow, we will have a nice chat and she will like me. I would be so happy getting my offer, as it would open my doors to the Machine Learning Science world, which I would be very happy part of. Please!!

Final interview

- chat only about behavioural questions, partially focuse on my previous experience.
- we had great chat, only I emphasized I want to move from SE towards ML, but they ideally want to have Fullstack ML Engineer / Scientist.
- Also she was quiet happy to start at the end of the year, so I can imagine start on October... —> I would have some free time.