IML Summary

Lasse Fierz - lfierz

Version: 4. Februar 2023

Basics

- General p-norm: $||x||_p = (\sum_{i=1}^{n} |x_i|^p)^{1-p}$
- Taylor: $f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \mathbb{O}(x^3)$
- Power series of exp.: $exp(x) := \sum_{k=0}^{\infty} \frac{x^k}{k!}$
- $\sum_{k=0}^{\infty}(xy)^k = \frac{1}{1-xy}$
- Entropy: H(X) $= \mathbb{E}_X\left[-log\mathbb{P}(X=x)\right]$
- KL-Divergence: $D_{KL}(P||Q) = \sum_{x\in\mathbb{X}} P(x)log\left(\frac{P(x)}{Q(x)}\right) \geq 0$
- $1 - z \leq exp(-z)$
- Cauchy-Schwarz: $|\mathbb{E}\left[X,Y\right]|^2 \leq \mathbb{E}(X^2)\mathbb{E}(Y^2)$
- Jensens Inequality: for a convex f(X): $f(\mathbb{E}(X)) \leq \mathbb{E}(f(X))$
- M p.s.d. if $v^T M v \succeq 0$

Probability Theory:

- Gaussian: $\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}exp(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2})$
- $(N)(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}}exp(-\frac{1}{2}(x-\mu)^T\boldsymbol{\Sigma}^{-1}(x-\mu))$
- $X \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma}), Y = A + BX \Rightarrow$ $Y \sim \mathcal{N}(A + B\boldsymbol{\mu}, B\boldsymbol{\Sigma^{-1}}B^T)$
- Binomial Distr.: $f(k,j;p) = \mathbb{P}(X=x) = \binom{n}{k}p^k(1-p)^{n-k}$
- $\mathbb{V}(X) = \mathbb{E}\left[(X - \mathbb{E}(X))^2\right] = \mathbb{E}(X^2) - \left[\mathbb{E}(X)\right]^2$
- $\mathbb{V}\left[X+Y\right] = \mathbb{V}\left[X\right] + \mathbb{V}\left[Y\right] + 2Cov(X,Y)$
- $Cov(X,Y) = \mathbb{E}\left[(X-\mathbb{E}(X))(Y-\mathbb{E}(Y))\right]$
- $Cov(aX,bY) = abCov(X,Y)$

Calculus

- $\int uv'dx = uv - \int u'vdx$ • $\frac{\partial}{\partial x}\frac{g}{h} = \frac{g'h}{h^2} - \frac{gh'}{h^2}$
- $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{b^T A x}) = A^T b$ • $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{b^T x}) = \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x^T b}) = b$
- $\frac{\partial}{\partial \boldsymbol{X}}(\boldsymbol{c^T X^T b}) = \boldsymbol{bc}^T$ • $\frac{\partial}{\partial \boldsymbol{X}}(\boldsymbol{c^T X b}) = \boldsymbol{cb^T}$
- $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x^T A x}) = (\boldsymbol{A^T} + \boldsymbol{A})x \overset{\text{A sym.}}{=} 2\boldsymbol{Ax}$
- $\frac{\partial}{\partial \boldsymbol{X}}Tr(\boldsymbol{X^T A}) = A$ • Tr.trick: $\boldsymbol{x^T A x} \overset{\text{inner prod.}}{=} TR(\boldsymbol{x^T A x}) \overset{\text{cyclic perm.}}{=} Tr(\boldsymbol{x x^T A}) = Tr(\boldsymbol{A x x^T})$
- $|X^{-1}| = |X|^{-1}$ • $\frac{\partial}{\partial \boldsymbol{X}}log|x| = x^{-T}$ • $\frac{\partial}{\partial x}|x| = \frac{x}{|x|}$
- $\frac{\partial}{\partial \boldsymbol{x}}||x||_2 = \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x^T x}) = 2x$
- $\frac{\partial}{\partial \boldsymbol{x}}||\boldsymbol{x} - \boldsymbol{b}||_2 = \frac{\boldsymbol{x}-\boldsymbol{b}}{||\boldsymbol{x}-\boldsymbol{b}||_2}$
- $\frac{\partial}{\partial \boldsymbol{x}}||x||_1 = sgn(x)$
- $\sigma(x) = \frac{1}{1+exp(-x)} \Rightarrow$
- $\nabla\sigma(x) = \sigma(x)(1-\sigma(x)) = \sigma(x)\sigma(-x)$
- $tanhx = \frac{2sinhx}{2coshx} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- $\nabla tanhx = 1 - tanh^2 x$
- $sin(a \pm b) = sin(a)cos(b) \pm cos(a)sin(b)$
- $cos(a \pm b) = cos(a)cos(b) \mp sin(a)sin(b)$

(Linear) Regression

General Regression: find $\hat{y} = f(x) \leftrightarrow \underset{\hat{y}(x)}{min}||y - \hat{y}(x)||_2^2$.

Linear Regression: Weights are applied linearly:
$f(x) = \omega x$ or nonlinear **base fct**: $f(x) = \omega\phi(x)$
Multidim.: $L = \underset{\omega}{min}||\boldsymbol{Y} - \boldsymbol{X\omega}||^2$,

$Y \in \mathbb{R}^n, x \in \mathbb{R}^{nxd}, \omega \in \mathbb{R}^d$

Closed Solution

If X^TX is invertible (X^TX has full rank $\Leftrightarrow rank(X) = min(d,n)$) $\Rightarrow$ closed solution: $\omega = \boldsymbol{(X^TX)^{-1}X^TY}$
∇L is $\mathbb{O}(nd)$, closed solution is $\mathbb{O}(nd^2)$.
Can't apply closed solution for linearly dependent features.
Note: the closed solution can also be seen as finding the geom. proj. of y onto the hyperplane span(X).
$(\boldsymbol{y} - \boldsymbol{X\hat{\omega}})^T X\omega = 0$

Optimization

If not solvable in closed form or expensive to invert $X^TX \rightarrow$ Gradient Descent:
$\omega_{t+1} \leftarrow \omega_t - \eta\nabla L(\boldsymbol{\omega_t})$, η is the learning rate.
Convergence guaranteed for $\eta \geq \frac{2}{\lambda_{max}}$, where λ_{max} is the max EV of X^TX.
X^TX diagonal $\Rightarrow$ contour lines (L const) are ellipses

Nonlinear Regression

Use fixed nonlinear feature maps of the inputs $\phi(x)$ but still tune $\omega \leftrightarrow \underset{\omega}{min}||y - \phi(x)\omega||^2$, with $\phi(x) \in \mathbb{R}^{nxp}$

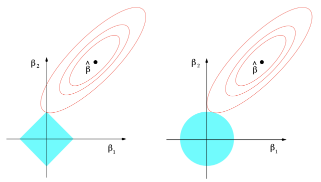
Note: When working with NNs both the weights and the non-linear functions are chosen.

For closed solution same applies rank$\phi(x) \overset{!}{=} min(n,p)$

Regularization

Among all unbiased solutions $(X^TX)^{-1}X^TY$ is the solution that has the smallest variance $\Rightarrow$ minimizes gen. Error
However the variance can get big $\Rightarrow$ small $L_{train}(\omega)$ but large $L_{gen}(\omega)$ due to overfitting. Noise increases weights and regularization counters that effect. $\Rightarrow$ Regularization:

One can set the ω of higher order features manually to zero ($\leftrightarrow$ choose a less complex model) or
Ridge Regression
$\underset{\omega}{min}||Y - X\omega||^2 + \lambda||\omega||^2$
Always allows for closed solution and lets LS converge faster through better conditioned problem (EVs of Hessian X^TX change)
Equivalent to performing Bayesianism approach with $p(\omega) = \mathcal{N}(\omega|0,\boldsymbol{\Lambda}^{-1})$ or linearly $p(\omega) = \mathcal{N}(\omega|0,1)$
Weights are decreased in general but not necessarily to exactly 0.
Lasso Regression
Not a convex loss $\Rightarrow$ no closed form solution
$\underset{\omega}{min}||Y - X\omega||^2 + \lambda|\omega|$
Equivalent to performing Bayesianism approach with Laplacian prior: $p(\omega_i) = \frac{\lambda}{4\sigma^2}exp(-|\omega_i|\frac{\lambda}{w\sigma^2})$
The weights of higher complexity features go to absolute zero $\Rightarrow$ sparse weight vector result



Left: Lasso, Right: Ridge
In general with increasing λ the bias increases. λ_{opt} can be found using CV.

Gradient Descent and Convexity

Gradient Descent

$\omega_{t+1} \leftarrow \omega_t - \eta L(\omega_t)$
Converges to a stationary point. $\nabla L(\omega) = 0 \Rightarrow$ GD stuck.
Complex fcts: $\nabla L(\omega)$ from lin. approx. and use small η
Large EVs for data depending heavily on one attribute and vice versa. Well conditioned if λ_{max} and λ_{min} are in similar range.
GD is sometimes slower and less accurate but there is more control and less comp. complexity
Gradient Methods: Momentum usage, Adaptive Methods, 2nd order methods
Stochastic GD: Use subsample from data for update step. Helps against saddle point conversion.

Convexity

Always:

- global min/max $\Rightarrow$ local min/max
- local min/max $\Rightarrow$ stationary point
- $L(\omega) < L(v)\forall v \neq \omega \Leftrightarrow \omega$ is a global min

Convexity:

- 0-order condition: $L(sw + (1-s)v) \leq sL(w) + (1-s)L(v)$ aka function is lower or equal to linear connection of two points.
- 1st-order: $L(v) \geq L(\omega) + \nabla L(\omega)^T(v - \omega)$ aka any point v on function is higher than point on linear approximation drawn at position ω
- 2nd-order: $\nabla^2 L(\omega)$ is p.s.d. aka non-neg. curvature throughout function.
- ω stationary $\Rightarrow \omega$ is local minimum
- ω is local minimum $\Rightarrow \omega$ is global minimum

Strong Convexity:

- 0-order: $L(sw + (1-s)v) + \epsilon \leq sL(w) + (1-s)L(v)$ so fct always a bit below linear connection of points
- 1st-order: same as convex
- 2nd-order: strictly positive curvature always
- ω is global minimum $\Rightarrow L(\omega) < L(v)\forall v \neq \omega$
- Only one global minimum

Convexity Operations:

- Linear Comb. of convex functions are convex
- $f(g(x))$ is convex if f convex and g affine or f non-decreasing and g convex.
- Adding a convex and a strictly convex fct. yields a strictly convex function

Model Selection

In general $y = f(x) + \epsilon$, where ϵ is random noise
We can never know $f(x)$ as we can only observe y. So we can't determine the estimation error $(f(x) - \hat{f}(x))^2$
We use the gen. error $(y - \hat{f}(x))^2 =$

$$\underbrace{(f(x) - \hat{f}(x))^2}_{\text{estimation error}} + \underbrace{\epsilon^2}_{\text{irreducible noise}} - \underbrace{2\epsilon(\hat{f}(x - f(x)))}_{\text{0 on average}}$$

Often interested in $\mathbb{E}\left[(y - \hat{f}(x))^2\right] \approx \underbrace{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2}_{\text{empirical error}}$

Bias and Variance

- **Bias** $= \mathbb{E}\left[(f(x) - \hat{f}(x))^2\right]$ Badness of model
  High for simple models and complex Ground Truths
- **Variance** $= \mathbb{E}\left[(\hat{f}(x) - \mathbb{E}\hat{f}(x))^2\right]$ fluctuation of $\hat{f}$
  High for a too complex model and too little data (overfitting)

For the noiseless case $y = f(x)$ a complex model can still overfit if the sample data is not representative of all data.
Generalization Error = bias^2 + **variance**, idea of regularization: increase bias a bit to strongly decrease variance

Cross Validation

To estimate gen. error $\Rightarrow$ train and test data. Usual splits are 50/50 and 80/20 (more often 80/20 because data is scarce)
To choose hyperparameters (e.g. regularization param λ or what choice of nonlinear features $\phi(x)$) we perform k-fold cross validation: Split training data into k batches

1. For each option of hyperparameter:

2. for each batch:
    - Train model on the whole training data except for the batch
    - Calculate validation error on remaining batch
3. Average validation error over all batches
4. Choose hyperparameter with lowest avg. val. error
5. Train model with that hyperparameter on the whole training set
6. Determine test error

Leave one out CV (LOOCV):

- Split training data into sets of one ⇒ validation batch is of size 1
- Results in best model approximation
- Validation error is pretty bad (only one sample) but avg. ok
- Computationally expensive

Dataset Size

In general more data is always better. A limited dataset might not be representative of the underlying distribution.
Usually y is noisy: $y = f(x) + \epsilon$ in that case a small number of samples and a complex model will overfit the sample noise.
In the noiseless case $n \to \infty \Rightarrow L_{train}(f(x)) \to 0$
For $n < d$ GD finds the solution that minimizes $||\omega||_2$

Classification

- Probabilistic generative: p(x,y)
  allows for sample generation and outlier detection
- Prob. discriminative: p(y—x)
  classification with certainty
- Purely discr. c: $X \to y$
  just classification, easiest

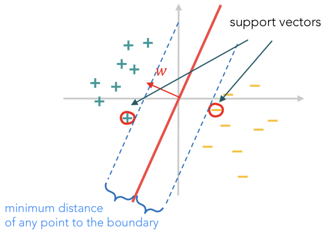
Lin. seperable data ⇒ infinitely many solutions ⇒ SVM

Loss Functions

- Cross Entropy:
  $\mathcal{L}^{CE} = -\left[y' log \hat{f}(x)' + (1-y')log(1-\hat{f}(x)')\right]$
  Where $y' = \frac{1+y}{2}$ and $\hat{f}(x)' = \frac{1+\hat{f}(x)}{2}$
- Zero one loss: $\mathbb{L}^{0/1} = \mathbb{I}\{sign(\hat{f}(x) \neq y)\}$
  Not convex nor continuous ⇒ surrogate logistic loss
- $\mathbb{L}^{\text{Hinge}} = \text{max}(0, 1 - y\hat{f}(x))$
- $\mathbb{L}^{\text{percep}} = \text{max}(0, -y\hat{f}(x))$
- $\mathbb{L}^{\text{logistic}} = log(1 + exp(-y\hat{f}(x)))$
- multidim. logistic loss: softmax:
  $\mathbb{L}_i^{\text{softmax}} = \frac{e^{-af_i}}{\sum_{j=1}^{K} e^{-af_j}}$
- $\mathbb{L}^{\text{exp}}(x)_i = exp(-y\hat{f}(x))$

GD on logistic loss:
$\omega_{t+1} = \omega_t - \eta\frac{1}{n}\sum_{i=1}^{n}\nabla_\omega g(y\langle\omega_t, x\rangle) =$
$\omega_t + \eta_t\frac{1}{n}\sum_{i=1}^{n}\frac{y_i x_i}{1+e^{y_i\omega_t x_i}}$ Converges to the ω that minimizes the l2-distance to the decision boundary (SVM sol.)
If classification error is not equally high for different classes ⇒ error metrics (see additionals)
Worst group error(related to group fairness): Highest error among all clusters of a class (e.g. if one blob is 100% false)
Robust generalization w.r.t. perturbations
Data augmentation, models that allow for invariance (e.g. CNNs)
Distribution shifts aka test data is different to training data: try to have the lowest possible error on the test samples that are similar to the training data.

SVM



Find ω that maximizes the min distance of the closest points (support vectors) to the decision boundary. (There are at least 3 SVs)
margin $= \min_i y_i\langle\omega, x_i\rangle$, distance to SV $= \frac{y_i\langle\omega,x_i\rangle}{||\omega||}$
Objective: maximize max margin direction:
$\underset{\omega}{\text{argmax}}$ margin(ω) so that
Either $||\omega|| = 1$ or $||\omega|| = \frac{1}{||margin||}$
Latter case: can look for ω in the smaller subspace of ω which yield a margin of 1
Objective: $\mathcal{L}$(soft margin) $=$

$$\underset{\omega,\xi}{min}\frac{1}{2}||\omega||^2 + C\sum_i \xi_i$$
$$\text{s.t. } y_i\omega^T x_i \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \; \forall i = 1,..,n$$

Solve using lagrangian:

$$\mathcal{L} = \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i(1 - \xi_i - y_i\omega^T x_i)$$

Kernels

If we choose at least one nonlinear $\phi(x)$ then $\hat{f}(x)$ can be non-linear
Note the comp. complexity of constructing $\phi(x)$ (degree m polynomial of features X $\in \mathbb{R}^{nxd}$)is $\mathcal{O}(nd^m)$ ⇒ huge for high dim. data

Kernel Trick

Feature maps only enter $\hat{f}(x)$ by their inner product.
Can write one of the possible global minimizers $\hat{\omega} = \phi^T a$, $a \in \mathbb{R}^n$ ⇒Can write objective as:
$L(\omega) = \frac{1}{n}\sum_{i=+}^{n} l$

Other Nonlinear Models

K Nearest Neighbours

- For each datapoint determine the k nearest neighbours and assign a class based on the majority of the there present datapoints.
- Heavily dependent on k ⇒ Cross Validation
- Error prone in high dim. because of large distances
- Needs a lot of data but $\mathcal{O}(nd)$ can be reduced to $\wr(n^p), p < 1$ if we allow for some error probability

Decision Trees

- At each node split data w.r.t. to one feature and a threshold (boundary at $x_i > t_i$)
- Each node returns class of the subset by majority
- Greedy Method: best step for current situation as opposed to generally best step ⇒ errors propagate.
- Very prone to overfitting as partitions can get very detailed
- ⇒ random forest (averaged result over trees with random splits.)

Neural Networks

Backpropagation: Running time grows linearly with num of params in feed forward momentum and that stuff
Vanishing, exploding gradient. vanishing problem not there for every input
Weight decay reduces complexity
what functions can be approximated at what point. A NN with one hidden layer and a nonlinear act. function can approximate every continuous function
CNNs: need also nonlinear act fcts to approximate nonlin fcts

Activation Functions

A neural network with one hidden layer and nonlinear activation functions can approximate every continuous function.

- Sigmoid
- Relu
- Relu Variants

Backpropagation

Weight updates make use of the chain rule of the NN structure.

Vanishing / Exploding Gradient

Vanishing Gradient

In Backprop. the weight updates depend on the gradients of preceeding weights. If the gradients are small (e.g. saturation in sigmoid) then the small gradients further shrink the update and for large networks the gradient of certain layers vanish.

Exploding Gradient

Similarly as for vanishing gradient but with big gradients. Mostly due to bad weight initialization.
Counteractions Good weight initialization is half the work. Secondly for the vanishing gradient problem a relu function can be useful as for that we don't have the saturation problem.

Convolutional Neural Networks

Unfeasible to fully connect vectorized images due to number of parameters.

- Invariant regarding shifts, scale and rotation
- Updates still through backpropagation

Dimension of image after CNN layer image: n x n, m kernels of size k*k, stride s, padding p:

$$l = \frac{n+2p-k}{s} + 1$$

(Max)Pooling

Strongly reduces number of parameters

Regularization in NNs

- Early stopping (before convergence to lowest training error)
- Dropout: deactivate about 50% of the nodes during training
- Data augmentation
- Batch normalization

Batch normalization

Normalize unit activations for a layer.
BN(v, γ, β)

- $\mu_s = \frac{1}{|S|}\sum_{i\in S} v_i$
- $\sigma_S^2 = \frac{1}{|S|}\sum i \in S(v_i - \mu_S)^2$
- $\hat{v}_i = \frac{v_i - \mu_S}{\sqrt{\sigma_S^2 + \epsilon}}$
- Scale and shift: $\bar{v}_i = \gamma\hat{v}_i + \beta$

Residual NNs

- Add possibility to skip layers e.g. feed input to intermediate layers
- Helps avoid vanishing gradients
- Allows for very deep NNs (1000+ layers)
- Can skip more than one layer (Dense Nets)

Clustering

Dimensionality Reduction

Oftentimes we have high dimensional data which leads to high computational costs.
One countermeasure for this is dimensionality reduction. $f : \mathbb{R}^D \to \mathbb{R}^d$, where $D > d$
Standardization PCA parallel to other method

Autoencoders

Statistical Perspective

Generative Modelling

Gaussian Mixture Model

Additionals

Standardization

Standardizing features $x_{new} = \frac{x-\mu}{\sigma}$ yields values between 0 and 1. Necessary if one feature is comprised of comparatively larger values than others and has thus a bigger influence on the weights. Especially important for euclidian distance based methods like **knn,SVM,PCA,NNs,GD**

- KNN and SVM are methods based on the euclidian distance between the points
- NNs converge faster with standardized data. Also helps with vanishing gradients.
- PCA requires standardization because it considers the variance of the featues in order to find the principle components.

Stdz always **after** train-test split.
Stdz not necessary for distance independent methods:

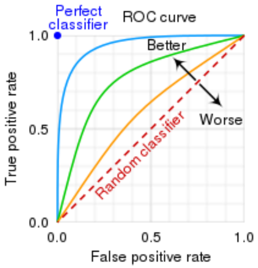
- Naive Bayes
- LDA
- Tree based methods (boosting, Random forests) etc.

Classification Metrics

Define as positive the outcome which is crucial to get right.
Hypothesis test: Set hypothesis, reject it if $\hat{p}(x) > \tau$ and accept it if $\hat{p}(x) < \tau$
Reject hypothesis ⇒ positive — higher τ ⇒ more negatives

- acc.= $\frac{TP+TN}{n}$ • prec.= $\frac{TP}{TP+FP}$
- FPR= $\frac{FP}{FP+TN}$ • Recall / TPR= $\frac{TP}{TP+FN}$
- balanced acc.= $\frac{1}{n}\sum_i TPR_i$ FDR= $\frac{FP}{TP+FP}$
- F1-score= $\frac{2TP}{\underbrace{2TP+FP+FN}_{\frac{2}{\frac{1}{Recall}+\frac{1}{prec.}}}}$ ROC= $\frac{\text{FPR}}{\text{TPR}}$

F1-score: only high if both Recall and Precision are high
Useful if only interested in positive class
ROC curve:



ROC curve is always increasing. Not necessarily convex curve.
The higher up the better
AUROC = area under ROC

Individual Additions