Intro to ML Summary 25.07.2019

1. Regression

Residual: quantifies *goodness of fit*

$$\hat{R}(w) = \sum_i r_i^2 = \sum_i (y_i - w^T x_i)^2$$

Least-squares: solve direct or iteratively

$$w^* = \arg\min \sum (y_i - w^T x_i)^2$$

Closed form: $w^* = (X^T X)^{-1} X^T y$

Convex: local minima is global minima

$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x')$

Gradient decent: start at any $w_0 \in \mathbb{R}^d$

$$w_{t+1} = w_t - \eta_t \nabla \hat{R}(w_t)$$

"Empirical risk": $\nabla \hat{R}(w) = -2 \sum_i r_i x_i^T$

Line search: $\eta_t \in argmin\, \hat{R}\big(w_t - \eta \nabla \hat{R}\big)$

Bold driver: increase step size if less risk

2. Model selection & validation

Expected error / "true risk"

$$R(w) = \int P(x,y)\,(y - w^T x)^2\, dxdy = E_{x,y}[(y - w^T x)^2]$$

$$\hat{R}_D(w) = \frac{1}{|D|} \sum (y_i - w^T x_i)^2$$

Ridge regression: regularize weights

$$\min_w \frac{1}{n} \sum (y_i - w^T x_i)^2 + \lambda \|w\|^2$$

$$\hat{w} = (X^T X + \lambda I)^{-1} X^T y$$

Standardization: $\mu = 0, \sigma^2 = 1$

$$\tilde{x}_{i,j} = (x_i - \mu_j)/\sigma_j$$

3. Classification

0/1 loss: $l_{0/1}(x_i) = [y_i \neq sign(w^T x_i)]$

Perceptron: $l_p(x_i) = \max(0, -y_i w^T x_i)$

Hinge loss: $l_h(w) = \max(0, 1 - y\, w^T x)$

Stochastic Gradient Decent: few points

$$w_{t+1} = w_t - \eta_t \nabla\, l(w_t; x', y')$$

Mini-batch: avg over batch of points (reduces variance, allows parallelization)

Perceptron: SGD with l_p and $\eta = 1$

Support Vector Machine: max. margin

Use l_h to enforce margin + regularize

$$\arg\min \frac{1}{n} \sum \max(0, 1 - y\, w^T x) + \lambda \|w\|_2^2$$

4. Feature selection

Greedy forward select: choose best feature and add 1 if it decreases loss

Greedy backward: remove if advantage

Lasso: use l_1 for sparsity regularization, force weights to 0 to simplify model

$$\min \frac{1}{n} \sum (y_i - w^T x)^2 + \lambda \|w\|_1$$

5. Kernels

Ansatz: $\hat{w} = \sum \alpha_i\, y_i\, x_i$

Kernels calc inner products efficiently:

$$\arg\min \frac{1}{n} \sum_i \max\left(0, -\sum_j \alpha y_i y_j x_i^T x_j\right)$$

$k(x, x') = \phi(x)^T \phi(x')$ for $x \rightarrow \phi(x)$

Kernel trick: inner product → kernel

Kernelized Perceptron: for any (x_i, y_i)

$$\hat{y} = sign\left(\sum_j \alpha_j y_j k(x_j, x_i)\right)$$

$\hat{y} \neq y_i$: $\alpha_i = \alpha_i + \eta_t$, $else\ \alpha_{t+1} = \alpha_t$

Kernel properties:

i) must be *symmetric*: $k(x,y) = k(y,x)$

ii) kernel matrix K must be *positive semi-definite:* $x^T K\, x \geq 0 \leftrightarrow \lambda_i \geq 0\ \forall\ EVs$

Can always construct a feature map:

$K = U\, D\, U^T = \phi^T \phi$, $\phi^T = U\, D^{1/2}$

$k(i,j) = K_{i,j} = \phi_i^T\, \phi_j$, $\phi: i \rightarrow \phi_i$

Linear: $k(x, x') = x^T x'$

Polynomial: $k(x, x') = (x^T x' + 1)^d$

Gaussian: $k = \exp(-\|x - x'\|_2^2 / 2h^2)$

Laplacian: $k = \exp(-\|x - x'\|_1 \ /\ h\ )$

Combine: $k_1 + k_2,\ k_1 * k_2,\ c * k_1\ (c > 0)$

use prev. knowledge, e.g. linear + Gaussian

K-Nearest neighbour

$$y = sign\left(\sum y_i [x_i\ among\ k\ NN\ of\ x]\right)$$

Kernelized Perceptron + SVM

$$\arg\min \frac{1}{n} \sum \max(0, \quad - y_i \alpha^T k_i)$$

$$\arg\min \frac{1}{n} \sum \max(0, 1 - y_i \alpha^T k_i) + \lambda \alpha^T D_y K D_y \alpha$$

where $k_i = [y_1 k(x_i, x_1), \ldots, y_n k(x_i, x_n)]$

Kernelized Linear Regression (KLR)

$$\hat{\alpha} = \arg\min \frac{1}{n} \|\alpha^T K - y\|_2^2 + \lambda \alpha^T K \alpha$$

Closed-form: $\hat{\alpha} = (K + n\lambda I)^{-1} y$

6. Class imbalance

Replace cost: $l_{CS}(w, x, y) = c_y\, l(w, x, y)$

Accuracy: $\frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{n}$

Precision: $\frac{TP}{TP + FP}$, Recall: $\frac{TP}{TP + FN}$

F1 score: $\frac{2\,TP}{2\,TP + FP + FN} = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$

True Positive: $\frac{TP}{TP + FN}$, False Positive: $\frac{FP}{TN + FP}$

Multi-class Hinge Loss: train many $w's$

$$l_{MC} = \max\left(0, 1 + max\, w^{(j)T} x - w^{(y)T} x\right)$$

7. Neural nets

Sigmoid: $\frac{1}{1+\exp(-z)}$, Tanh: $\frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$

ReLu: $\max(z, 0)$, Leaky ReLu: $\alpha\, z, z < 0$

Forward-propagation

$$z^{(l)} = W^{(l)} v^{(l-1)}, \qquad v^{(l)} = \varphi\big(z^{(l)}\big)$$

Back propagation

Error: $\delta^{(l)} = \varphi'\big(z^{(l)}\big) * \big(W^{(l+1)T} \delta^{(l+1)}\big)$

Gradient: $\nabla_{W^{(l)}}\, l(W, x, y) = \delta^{(l)} v^{(l-1)T}$

Momentum

$$a_{t+1} = m * a_t + \eta_t \nabla_w\, l(W, y, x)$$

$$W_{t+1} = W_t - a_{t+1}$$

8. Clustering

k-means: linear decision boundaries

$$\hat{R}(\mu) = \sum_i \min_j \left\|x_i - \mu_j\right\|_2^2$$

Lloyd's heuristic:

$$z_i^{(t)} = \arg\min_j \left\|x_i - \mu_j^{(t-1)}\right\|^2$$

$$\mu_j^{(t)} = \frac{1}{n_j} \sum\nolimits_{z_i^{(t)} = j} x_i$$

9. Dimension reduction

Linear dimension reduction: $\|w\|_2 = 1$

$$(w^*, z^*) = \arg\min \sum \|z_i w - x_i\|_2^2$$

Principal Component Analysis: k-dim.

$$z_i = W^T x_i\,, \qquad W = (v_1|\dots|v_k)$$

$$\Sigma = \sum \lambda_i\, v_i\, v_i^T = \frac{1}{n}\sum x_i\, x_i^T\,, \quad \lambda_i \geq 0$$

Can also be obtained using SVD:

$$X = U\,S\,V^T \rightarrow W = (v_1|\dots|v_k)$$

Kernel PCA

$$\max_{\alpha}\ \alpha^T K^T K\alpha \quad s.t. \quad \alpha^T K\alpha = 1$$

$$k(x_i, x_j) = x_j^T\, x_i\,, \qquad K = (k_1|\dots|k_n)$$

$$\alpha^{(i)} = \arg\max \alpha^T K^T K\alpha \;=\; v_i/\sqrt{\lambda_i}$$

$$z_i = \sum \alpha_j^{(i)}\, k(x, x_j) \in \mathbb{R}^k$$

Autoencoder

$$f(x;\theta) = f_2(f_1(x;\theta_1);\theta_2)$$

10. Probabilistic modelling

Prediction risk: should be minimized

$$R(h) = \int P(x,y)\, l(y,h) = E_{x,y}[l(y,h)]$$

Maximum Likelihood Estimation (MLE)

$$\theta^* = \arg\max_{\theta} \hat{P}(y_1,\dots,y_n|x_1,\dots,x_n,\theta)$$

Lin. Gaussian: $\arg\min \sum\ (y_i - w^T x_i)^2$

Maximum A Posteriori (MAP)

$$\begin{aligned} \mathrm{w}_{\mathrm{MAP}} &= \arg\max_{w} P(w|D)\,, \quad P(w)\ known \\ &= \arg\max_{w} \prod P(y_i|x_i,w)P(w) \\ &= \arg\min_{w} \sum l\,(w,D) - \log P(w) \end{aligned}$$

11. Logistic regression

MLE for logistic regression (classification)

$$l(w) = \log(1 + exp(-yw^T x))$$

where assume Bernoulli noise: $P(y|x,w)$

$$= Ber\big(y, \sigma(w^T x)\big) = \frac{1}{1+\exp(-yw^T x)}$$

$$\hat{R} = \sum_i \log(1 + exp(-y_i\, w^T x_i)) + reg$$

SGD for L2-reg. logistic regression

Update: $P(Y \neq y|w,x) = \frac{1}{1+\exp(yw^T x)}$

Step: $w \leftarrow w(1 - 2\lambda\eta_t) + \eta_t yx P(Y \neq y)$

Multi-class logistic regression: e.g. NN

$$P(Y = i \mid x, w_1, \dots, w_c) = \frac{\exp(w_i^T x)}{\sum \exp\big(w_j^T x\big)}$$

12. Bayesian decision theory

Min. cost: $a^* = \arg\min E[C(y,a)|x]$

Logistic regression: most likely action

$$a^* = \arg\max \hat{P}(y\,|x) = sign(w^T x)$$

Doubtful logistic regression: "better ask"

$$C\ \begin{cases}[y \neq a], a \in \{\pm 1\} \\ c\,, \quad a = D\end{cases},\quad a^*\ \begin{cases} y\,, P(y|x) \geq 1 - c \\ D\,, \quad otherwise\end{cases}$$

Least-squares regression

$$C(y,a) = (y-a)^2\,, \qquad a^* = w^T x$$

13. Generative modelling

Discriminative models: learn $P(y|x)$

Generative models: learn $P(y,x)$

1. Estimate prior on label: $P(y)$
2. Conditional dist. per class: $P(x \mid y)$
3. Predictive dist: $P(y\,|x) \approx P(y)P(x|y)$
   each feature is indep. of other, given y

$$P(y|x) = \frac{1}{Z}\, P(y)\, P(x|y) = \frac{P(x,y)}{P(x)}$$

$$Z = \sum_y P(y)\, P(x|y) = P(x)$$

To maximize: $\hat{y} = \arg\max P(y' \mid x)$

Gaussian Naïve Bayes Classifier

$$P(Y = y) = p_y = Count(Y = y)/n$$

$$P(x|y) = \mathcal{N}\big(x_i; \mu_{y,i}, \sigma_{y,i}^2\big)$$

$$\mu_{y,i} = \frac{\sum_{y_j = y} x_{j,i}}{Count(y)}, \quad \sigma_{y,i}^2 = \frac{\sum_{y_j=y}\big(x_{j,i} - \mu_{j,i}\big)^2}{Count(Y=y)}$$

$$y = \arg\max_{y'} P(y') \prod\nolimits_i P(x_i|y')$$

Shared variance: linear class boundary

$$y = sign\big(f(x)\big), \qquad f(x) = w^T x + \omega_0$$

Discriminant function

$$f(x) = \log\frac{P(Y=1\mid x)}{P(Y=-1\mid x)}$$

Gaussian Bayes Classifier

As features no more independent, we require the *empirical covariance matrix*

$$\Sigma_y = \frac{1}{Count(y)}\sum\big(x_i - \mu_y\big)\big(x_i - \mu_y\big)^T$$

Categorical Naïve Bayes Classifier

$$\theta_{c|y} = P(X_i = c|y) = \frac{C(X_i = c, Y = y)}{Count(Y=y)}$$

14. Missing data

Gaussian Mixture Models (GMM)

try to guess label by modelling data

$$P(x \mid \theta) = \sum_i w_i\, \mathcal{N}(x; \mu_i, \Sigma_i)$$

$$w_i \geq 0\,, \qquad \sum\nolimits_i w_i = 1$$

Represent data as clusters of Gaussians:

$$L = -\sum\nolimits_i \log \sum\nolimits_j w_j\, \mathcal{N}\big(\, x_i\,|\mu_j, \Sigma_j\big)$$

Estimating $P(x)$ permits *outlier detection*

Gaussian Mixture Classifier (GMC)

$$P(y|x) = \frac{P(y)}{P(x)}\sum\nolimits_j w_j^{(y)} \mathcal{N}\left(x; \mu_j^{(y)}, \Sigma_j^{(y)}\right)$$

Hard-EM: force data point choose class

E-step (expectation): predict most likely

$$z_i^{(t)} = \arg\max_z P\big(z\big|\theta^{(t-1)}\big)P\big(x_i\big|z,\theta^{(t-1)}\big)$$

$$D^{(t)} = \left\{\left(x_1, z_1^{(t)}\right), \dots, \left(x_n, z_n^{(t)}\right)\right\}$$

M-step (maximization): compute MLE

$$\theta^{(t)} = \arg\max_{\theta} P\big(D^{(t)}\big|\theta\big)$$

Soft-EM: assign "responsibilities"

$$\gamma_j(x) = P(Z = j|x,\theta) = \frac{w_j\, P\big(x\big|\mu_j, \Sigma_j\big)}{\sum_l w_l P(x|\mu_l, \Sigma_l)}$$

$$w_j^{(t)} = \frac{1}{n}\sum \gamma_j^{(t)}(x_i), \quad \mu_j^{(t)} = \frac{\sum \gamma_j^{(t)}(x_i) x_i}{\sum \gamma_j^{(t)}(x_i)}$$

$$\Sigma_j^{(t)} = \frac{\sum \gamma_j^{(t)}(x_i)\left(x_i - \mu_j^{(t)}\right)\left(x_i - \mu_j^{(t)}\right)^T}{\sum \gamma_j^{(t)}(x_i)}$$

Choose k using cross-validation, avoid $\sigma \approx 0$

Semi-supervised learning

For labelled data: $\gamma_j(x_i) = [j = y_j]$

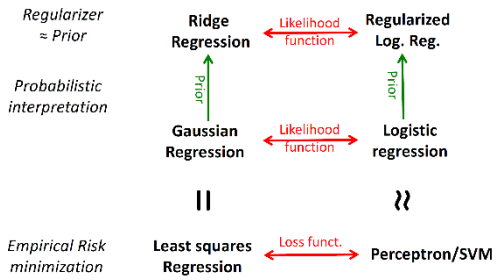
M-step ≅ GBC with weighted data

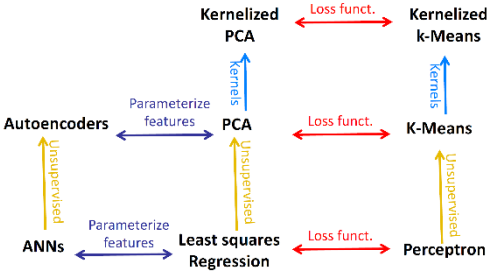
Generative Adversarial Network

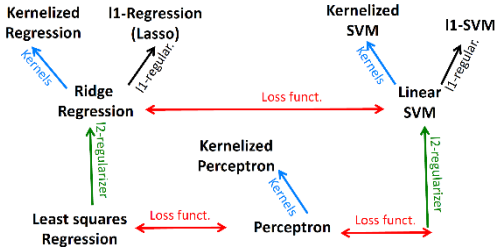
Use discriminative learning to train generative model (hard to distinguish)

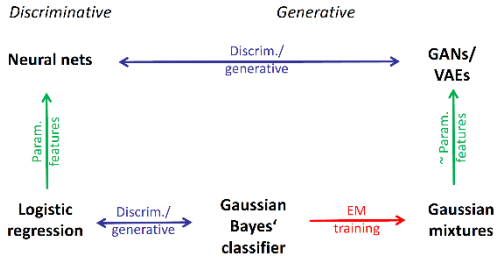
- *Generator* G vs *Discriminator* D

Summary slides









General terminology

Regression: predict real-valued labels

Representation: should be
- independent of length of document
- include ordering (e.g. pairs of words)
- aggregate similar words ("embedding")

Overfitting: learn training noise as well
- inefficient, no generalizable solution

Unsupervised learning: no given output
- *clustering, dim. reduction, generative*
- common representation of data sets
- identification of latent variables
- exploratory data analysis, anomalies
- feature learning / embedding

Transfer learning: learn on one domain, test / apply knowledge on another one

Reinforcement learning: interact with an unknown environment to learn

Homogeneous representation

$$\tilde{w} = [\, w_1 \dots w_d \; w_0 \,], \tilde{x} = [x_1 \dots x_d \; 1]^T$$

Always test on separate testing set!
Else underestimate the prediction error (estimated error ≤ expected error):

$$E\left[\hat{R}_D(w_D)\right] \leq E[R(w_D)]$$

K-fold cross-validation: reduce variance
1. partition data into k folds
2. Test on (k-1) folds, evaluate on last

Surrogate loss: replace original function, as better computationally behaving (e.g. differentiable at all places); validate with the loss you actually care about!

Stochastic gradient decent: evaluate gradient only at few points for efficiency

Feature selection: reduce features for
- interpretability ("understand" class)
- generalization (simpler, no overfitting)
- computation (less storage)

Nonlinear classification boundaries: use non-linear transformation of feature vectors and linear classification

Kernels: allow to efficiently compute on high-dimensional feature vector without explicitly calculating the transformation; as don't explore high-dimensional space (only based on data), no overfitting

Kernelized perceptron: similar to k-NN, but with optimized weights α_i; can capture global trends (sparse $\alpha_i's$) and only depend on "wrongly classified" data

Non-parametric learning: number of parameters = number of data points

Class-imbalance due to small sample set
- *subsampling:* remove majority samples
- *upsampling:* repeat minority samples
- *cost-sensitive classification:* $l_{CS} = c_y \, l$

Receiver Operator Curve (ROC): True Positive Rate (y) / False Positive Rate (x)
- *Area under Curve (AUC)* of ROC

One-vs-all (OvA): train one classifier per class, one with highest confidence wins (requires unit length on weights to remain comparable, else cannot)
- inherently imbalanced data

One-vs-one (OvO): train one classifier for each pair of classes, class with highest number of positive prediction wins

Error correcting output codes: try to estimate "label" of class as bit string

Artificial Neural Network (ANN): nonlinear functions which are nested compositions of (variable) linear functions composed with (fixed) non-linearities; use recursively inside

$$f(x, w, \theta) = \sum_j \omega_j \, \varphi\left(\theta_j^T \, x\right)$$

Vanishing Gradient Problem: activation functions lack gradient away from origin

Forward propagation:
for *evaluation* from input to output
Backward propagation:
for *optimization* from output to input

Universal Approximation Theorem: "can approximate any piece-wise linear fct with sufficient neurons" using sigmoids

Softmax: used for multi-class classification at output of NN

$$\sigma_i = \frac{e^{\gamma_i}}{\sum_j e^{\gamma_j}}$$

Weight initialization: keep variance of weights approximately constant using Glorot / He initialization

Learning rate: guarantee convergence by reducing step size after a while, e.g.

$$\eta_t = \min(0.1, 100/t)$$

Overfitting for NNs can be prevented
- *early stopping:* don't converge fully, but stop after validation error increases
- *regularization:* add penalty for weights
- *dropout:* don't train all weights always
- *batch-normalization:* ensure that inputs of hidden layers are normalized

Augmentation: create artificial samples by using invariant transformation

Convolutional Neural Network: specialized applications where each hidden unit depends only on "close-by" input, weights identical across one layer

Pooling layer: aggregate inputs by e.g. doing average or max pooling over patch

Convex optimization: For kernels, can never get stuck in local minima as convex problem; ANN & k-means non-convex!

Unsupervised learning: "learning without labels", learn functional relationship, exploratory data analysis
- *clustering:* unsupervised classification
- *dimension reduction:* unsup. Regression

Clustering
- *hierarchical:* build a tree of clusters
- *partitional:* based on partition cost
- *model-based:* maintain cluster model

K-means: vulnerable to over- or underrepresent clusters by random
- K-Means++: pick likelihood increases with distance to existing centers

$$p = \frac{1}{z} \min \|x_i - \mu_e\|^2$$

Elbow method: diminishing return with increased complexity, hence introduce cost for additional complexity to prevent ever-decreasing loss functions

Embedding: low-dimensional representation of complex feature vector

Principal Component Analysis: project feature vector onto k largest eigenvectors to compress with least loss

Spectral clustering: Kernel k-means
- apply k-means in the feature space induced by the kernel k

Non-linear feature maps
- can discover non-linear feature maps in closed form using kernel PCA
- Kernel PCA requires data to create PCA

Autoencoder: try to losslessly compress ("encode") and decompress ("decode")
- number of output = number of inputs, but hidden layers are usually smaller
- "Supervised" by using original data
- equivalent to PCA for $\varphi(z) = z$
- can be used to denoise (capacity too small to capture noise)

Bias Variance trade-off: more complex functions have less bias but high variance

$Prediction\ error = Bias^2 + Variance + Noise$

Bias: excess risk knowing P(X,Y), inf. data
Variance: risk due to using limited data
Noise: risk incurred by optimal model

MAP: explicitly express assumptions on your parameter distribution with a *prior*
- can be seen as regularization on MLE

Prior: can regard prior as a regularization
- *L2:* Gaussian prior
- *L1:* Laplacian prior

Asymmetric loss: can punish false positives more than false negatives

Active learning: request labels for the data you are most uncertain about
- called "uncertainty sampling"
- violates i.i.d. assumption

Discriminant function: for GNB, same predictions as logistic regression

$$y = sign\left(log \frac{P(Y=1|x)}{P(Y=-1|x)}\right)$$

GNB vs GBC
- GNB can suffer from overconfidence if features are not independent
- GBC requires quadratic complexity in *d*

Linear Discriminant analysis
projection to 1-dim subspace that maximizes ratio of between-class / within-class variance
- LDA: little within, max between
- PCA: maximize all variance

Outlier detection
Generative models allow outlier detection by defining $P(x) \leq \tau$; however, are less robust if model assumptions are not valid

Conjugate distribution: posterior distribution remains same family as prior

GMM: try to guess latent variable (clustering), as don't have the label
- else could directly do MLE / GBC

Formulae

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}, \qquad p \in (1, \infty)$$

$$\langle x, y \rangle \leq \|x\|_p \, \|y\|_q, \qquad \frac{1}{p} + \frac{1}{q} = 1$$

Probability

$$P(A_i|B) = \frac{P(B|A_i)\,P(A_i)}{\sum P(B|A_j)\,P(A_j)}$$

$$\begin{aligned} Var(X) &= E\left[(X - E[X])^2\right] \\ &= E\left[X^2\right] - E[X]^2 \end{aligned}$$

Gaussian

$$N(\mu, \sigma^2) \sim \frac{1}{\sqrt{2\pi\sigma^2}}\, exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{i,j}, \quad \sigma_j^2 = \frac{1}{n}\sum_{i=1}^{n}(x_{i,j} - \mu_j)^2$$

Empirical covariance matrix

$$\Sigma = \frac{1}{n} \sum_{i=1}^{n} x_i\, x_i^T$$

Bayes rule

$$P(w|x, y) = \frac{P(w|x)\,P(y|\,x, w)}{P(y|x)}$$

Least square loss

$$l(y, h) = \left(Y - h(x)\right)^2$$

Poisson distribution

$$P(X = x) = \frac{\lambda^x\, e^{-\lambda}}{x!}$$

KL divergence

$$KL(p \,||\, q) = \sum -p(x) log \frac{p(x)}{q(x)}$$