— Machine Learning Notes —

Contents

1 Introduction

ML: Tries to automate the process of **inductive inference**.

1. Deduction: Learning from rules
2. Induction: Learning from examples

1.1 Math

1.1.1 Common Norms

• $\|x\|_0$	number of non-zero elements
• $\|x\|_1 = \sum_{i=1}^{d} \|x_i\|$	Sum of absolutes
• $\|x\|_2 = \sqrt{\sum_{i=1}^{d} x_i^2}$	Euclidean distance
• $\|x\|_2^2 = \sum_{i=1}^{d} x_i^2$	Sum of squares
• $\|x\|_\infty = \max_{i=1,\dots,n} \|x_i\|$	Maximum absolute value

1.1.2 determinant, trace, inverse

- $\det(A) = |A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$
- $\operatorname{tr}(A) = \sum_{i=1}^{n} a_{ii}$
- $A^{-1} = \frac{1}{\det(A)} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$

1.1.3 Transpose rules

- $(A^T)^T = A$
- $(A+B)^T = A^T + B^T$
- $(AB)^T = B^T A^T$
- $(A^{-1})^T = (A^T)^{-1}$

1.1.4 Eigenvalues and Eigenvectors

Example: $f(w) = 0.5w^T Mw$

- Hessian: $\nabla^2 f(w) = M = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix}$
- Eigenvalues $1,3$
- Function along eigenvectors like $1x^2$ and $3x^2$

1.1.5 Derivative and Hessian

$$L(w) : \mathbb{R} \to \mathbb{R}^m \Rightarrow \nabla L(w) = \begin{pmatrix} \frac{\partial}{\partial w_1} L(w) \\ \frac{\partial}{\partial w_2} L(w) \\ \vdots \\ \frac{\partial}{\partial w_n} L(w) \end{pmatrix} \Rightarrow \nabla L(w) = \begin{pmatrix} \frac{\partial L_1}{\partial w_1} & \frac{\partial L_2}{\partial w_1} & \cdots & \frac{\partial L_m}{\partial w_1} \\ \frac{\partial L_1}{\partial w_2} & \frac{\partial L_2}{\partial w_2} & \cdots & \frac{\partial L_m}{\partial w_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial L_1}{\partial w_d} & \frac{\partial L_2}{\partial w_d} & \cdots & \frac{\partial L_m}{\partial w_d} \end{pmatrix}$$

$$\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1}(x) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1}(x) & \cdots & \frac{\partial^2 f}{\partial x_d \partial x_d}(x) \end{pmatrix}, \qquad x = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix} \in \mathbb{R}^d$$

2 Supervised learning

- input X, output Y
- $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \dots, x_d^{(i)}) \in \mathbb{R}^d$
- training data: $(x^{(i)}, y^{(i)})_{i=1..n} \subset X \times Y$
- Goal: learn $f : X \to Y$ for model class F on examples

2.1 Least squares regression

$\tilde{X}, \tilde{w}$ are extended with bias:

$$\min_{\tilde{w}} \frac{1}{2} \left\|\tilde{X}\tilde{w} - y\right\|^2 \Rightarrow \min_{w} \frac{1}{2} \|Xw - y\|^2$$

Solve with gradient and set to zero:

$$\begin{aligned} L &= \frac{1}{2} \sum_{i=1}^{n} ((X_i^T w_i) - y_i)^2 \\ &= \frac{1}{2} \left(\sum_{i=1}^{n} (X_i^T w_i)^2 - 2(X_i^T w_i) y_i + y_i^2 \right) \end{aligned}$$

$$\begin{aligned} \nabla L &= \frac{\partial}{\partial w} \left(\frac{1}{2} \left(\sum_{i=1}^{n} (X_i^T w_i)^2 - 2(X_i^T w_i) y_i + y_i^2 \right) \right) \\ &= \frac{1}{2} \left(\sum_{i=1}^{n} 2(X_i^T X_i w_i) - 2(X_i^T) y_i \right) \\ &= \sum_{i=1}^{n} X_i^T X_i w_i - X_i^T y_i \\ &= X^T X w - X^T y \\ &= X^T (Xw - y) \end{aligned}$$

$$\nabla L = X^T(Xw - y) = 0 \Rightarrow (X^T X) w = X^T y \Rightarrow w = (X^T X)^{-1} X^T y$$

2.2 Error types

- real value $y^{(i)}$, predicted value $\hat{y}^{(i)}$
- Mean squared error is average error, applied to any regression model:

$$\frac{1}{n}\sum_{i=1}^{n}(y^{(i)}-\hat{y}^{(i)})^2$$

- Mean absolute error is average absolute error:

$$\frac{1}{n}\sum_{i=1}^{n}\left|y^{(i)}-\hat{y}^{(i)}\right|$$

- Root mean squared error:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y^{(i)}-\hat{y}^{(i)})^2}$$

2.3 Precision, Recall, etc.

- Precision: $\frac{TP}{TP+FP}$, how many of predicted positive are actually positive
- Recall (true positive rate, sensitivity): $\frac{TP}{TP+FN}$, how many of actual positive are predicted positive
- TN rate (specificity): $\frac{TN}{TN+FP}$, how many of actual negative are predicted negative
- Accuracy (ACC): $\frac{TP+TN}{TP+TN+FP+FN}$
- Balanced accuracy: $\frac{1}{2}\left(\frac{TP}{TP+FN}+\frac{TN}{TN+FP}\right)$
- F1 score (ignores usually large true negatives): $2\cdot\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$

2.4 Gradient descent

Alternative to least squares regression. Algorithm:

1. Compute gradient $\nabla L(w) = X^T(Xw-y)$
2. Negative gradient shows to steepest descent
3. $w^{(t+1)} = w^{(t)} - \gamma^{(t)}\cdot\nabla L(w^{(t)})$

2.4.1 Derivative examples

- $L(w) = w_1^2 + w_2^2$
  $\Rightarrow \nabla L(w) = \begin{pmatrix} 2w_1 \\ 2w_2 \end{pmatrix}$
- $L(w) = \|w\|_2^2 = w^T w$
  $\Rightarrow \nabla L(w) = 2w$
- $L(w) = w^T A w$
  $\Rightarrow \nabla L(w) = Aw + A^T w$
- $L(w) = \|Xw - y\|^2 = w^T X^T X w - y^T X w - w^T X^T y + y^T y$
  $\Rightarrow \nabla L(w) = 2X^T(Xw - y)$

2.4.2 Convexity

Set C convex if line between any two points of C in C. $\forall x, y \in C$ and $\lambda \in \mathbb{R}$ with $0 \leq \lambda \leq 1$:

$$\lambda x + (1-\lambda) y \in C$$

Function $f : \mathbb{R}^d \to \mathbb{R}$ convex if (f) is a convex set and $\forall x, y \in (f)$, $\lambda \in \mathbb{R}$ with $0 \leq \lambda \leq 1$:

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

Gradient descent returns global optimum for convex functions.

Optimization problem: $\min f(x), x \in X \subseteq \mathbb{R}^d$ has local minimizer $x^* \in X$ if $\exists \varepsilon > 0$ with:

$$\forall y \in X \text{ with } \|x^* - y\| \leq \varepsilon : f(x^*) \leq f(y)$$

Global minimizer if $f(x^*)$ is lowest of all optimizers.

Symmetric matrix A is positive semidefinite ($A \succcurlyeq 0$) if :

$$x^T A x \geq 0, \forall x$$

Positive definite ($A \succ 0$) if $\forall x \neq 0$

Symmetric matrix A is positive semidefinite iff all eigenvalues are ≥ 0 and positive definite iff all > 0.

If function is one-dimensional: Convex if $f''(x) \geq 0$. If multidimensional: Convex if 2nd derivative is psd.

2.4.3 Backtracking line search

Algorithm:

```
1. Input: $x, \Delta x, \alpha \in (0, 0.5), \beta \in (0, 1)$
2. $t = 1$
3. while $f(x + t\,\Delta x) > f(x) + \alpha\, t\, \nabla f(x)^T \Delta x$:
4.     $t = \beta\, t$
```

2.4.4 Solve LSR

1. $L(w) = \frac{1}{2} \|Xw - y\|_2^2$
2. $\nabla L(w) = X^T(Xw - y)$
3. $\nabla L(w) = X^T X$ is symmetric and psd

2.4.5 Subgradient method

If function not differentiable, e.g. $\|w\|_1$

- gradient is subgradient (convex hull of gradients)
- choose constant step length g
- $w^{(t+1)} = w^{(t)} - \gamma^{(t)} \cdot g$ with $\gamma^{(t)} = \frac{1}{\sqrt{t}}$
- find $g \in \mathbb{R}^d$ at $x \in (f)$ with:

$$f(y) \geq f(x) + g^T(y - x), \forall y \in (f)$$

2.5 Polynomial Regression

- $X \in \mathbb{R}, Y \in \mathbb{R}$
- $f(x) = w_d x^d + w_{d-1} x^{d-1} + \ldots + w_1 x^1 + w_0$
- find best $w = (w_d, \ldots, w_0) \in \mathbb{R}^{d+1}$
- loss function is squared loss: $l(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$

With $\hat{y} = f(x^{(i)}) = \sum_{j=0}^{d} w_j (x^{(i)})^j = (\tilde{x}^{(i)})^T w$ rewrite as:

$$
\begin{aligned}
w^* &= \min_w \sum_{i=1}^{n} \frac{1}{2} (y^{(i)} - \hat{y}^{(i)})^2 \\
&= \min_w \sum_{i=1}^{n} \frac{1}{2} (y^{(i)} - (\tilde{x}^{(i)})^T w)^2 \qquad \text{(LSR)}
\end{aligned}
$$

Solve $||Xw - y||^2$ with Basis functions:

$$
X = \begin{pmatrix} f_1(x^{(1)}) & f_2(x^{(1)}) & \ldots & f_m(x^{(1)}) \\ f_1(x^{(2)}) & f_2(x^{(2)}) & \ldots & f_m(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x^{(n)}) & f_2(x^{(n)}) & \ldots & f_m(x^{(n)}) \end{pmatrix} \qquad y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{pmatrix}
$$

2.6 Underfitting / Overfitting

Underfitting: Model too simple, degree low

Overfitting: Model too complex, degree high

Too high model complexity $\rightarrow$ Higher training error

Lower polynomial degree or basis functions $\rightarrow$ Lower model complexity

2.6.1 k-fold Cross Validation

Mitigate Overfitting: Split training data into k (usually 10) and pick one for **validation data**.

Train model on one training block, run on validation data and compute error. Repeat for all blocks and average.

2.6.2 Regularization

Constrain magnitude ($\|w\|_2$, $\|w\|_1$, etc.)

Lagrangian to remove constraint

$$\begin{array}{ll} \min_w & L(w) \\ st & \|w\|_2^2 \leq t \end{array} \quad \rightarrow \quad \min_w \quad L(w) + \frac{\lambda}{2} \|w\|_2^2$$

if $L(w) = \frac{1}{n} \sum_{i=1}^{n} l(y^{(i)}, \hat{y}^{(i)})$:

1. Empirical risk minimization (ERM): $\min_w L(w)$
2. Regularized risk minimization (RRM): $\min_w L(w) + \|w\|$

2.6.3 Bias-Variance Tradeoff

Prediction error is sum of variance and bias

- Variance spreads predictions around true value
- Bias puts predictions away from true value

With complexer model:

1. Test data has min somewhere
2. Bias gets lower
3. Variance gets higher

2.6.4 Regularizers

Ridge Regression: LSR with $\|w\|_2$-regularizer:

$$\min_w \frac{1}{2n} \|Xw - y\|_2^2 + \frac{\lambda}{2} \|w\|_2^2$$

Least absolute shrinkage and selection operator (LASSO): $\|w\|_1$-regularizer:

$$\min_w \frac{1}{2n} \|Xw - y\|_2^2 + \lambda \|w\|_1$$

Solved with subgrad method, performs feature selection.

Elastic Net: Combination of both

$$\min_w \frac{1}{2n} \|Xw - y\|_2^2 + \lambda \left(\alpha \|w\|_1 + \frac{1-\alpha}{2} \|w\|_2^2 \right)$$

Often used for gene expression data.

Robust Regression with $\|w\|_1$-regularizer:

$$\min_w \frac{1}{n} \|Xw - y\|_1$$

Solved with subgrad method. Often used with Huber Loss for faster, simpler optimization.

2.7 Feature Scaling

- Features should be $[0,1]$ or $[-1,1]$
- Regularizer not invariant to scaling
- also on test data!

Normalize data: Center and scale each feature of data matrix $X_{i,j} = (x_j^{(i)})$

$$X_{:,j}^{\text{centered}} = X_{:,j} - \bar{x}_j = X_{:,j} - \frac{1}{n} \sum_{i=1}^{n} x_j^{(i)}$$

$$X_{:,j}^{\text{scaled}} = \frac{X_{:,j}^{\text{centered}}}{\left\| X_{:,j}^{\text{centered}} \right\|_2}$$

2.7.1 MLE and MAP

Example: For Coin-throw with $p(\text{head}) = \theta$: 3 heads, 7 tails. What is most likely θ?

$$p(y^{(1)}, y^{(2)}, \ldots, y^{(n)} \mid \theta) = \prod_i p(y^{(i)} \mid \theta) = \theta^3 (1-\theta)^7$$

Maximum Likelihood Estimator (MLE): Find θ for max probability:

$$\max_\theta \theta^3 (1-\theta)^7$$

Maximum A Posteriori (MAP): Find θ for max probability with prior:

$$\max_\theta \theta^3 (1-\theta)^7 \cdot p(\theta \mid \text{observation})$$

with $p(\theta \mid \text{observation}) = \frac{p(\text{observation}|\theta) \cdot p(\theta)}{p(\text{observation})}$

Here, we maximize the product of the likelihood times the prior:

$$\arg\max_w p(w \mid X, y) = \arg\max_w p(y \mid X, w) \cdot p(w)$$

Which is the same as minimising the loss and the regularizer:

$$\arg\min_w \sum_{i=1}^{n} (y^{(i)} - (x^{(i)})^T w)^2 + \lambda \, \|w\|_2^2$$

prior (variance) is regularizer.

Empirical risk min.	Maximum likelihood
Minimize	Maximize
Sum	Product
Risk / Loss function	Noise Distribution
l_1-loss	Gaussian Distribution
l_2-loss	Laplacian Distribution

Logarithmic equivalence

2.8 Binary Classification

- $X = \mathbb{R}^d$
- $Y = \{-1, 1\}$
- training data $(x^{(i)}, y^{(i)})_{i=1..n}$
- learn $f : X \rightarrow Y$
- linear function (hyperplane in multidimensional space) $f(x) = x^T w + b = 0$ returning $(f(x))$

Loss functions:

1. Logistic function
    - penalizes points on correct side
    - close to decision boundary
    - asymptotical linear growth
    - MLE applied to logistic function $\log(1 + \exp(-t))$ is logistic regression
    - Logistic regression: $f(x) = \log(1 + \exp(-y \cdot (x^T w + b)))$
2. Hinge loss: $f(x) = \max(0, 1 - t)$
3. Squared hinge loss: $f(x) = \max(0, 1 - t)^2$

2.8.1 Support Vector Machine (SVM)

Pick hyperplane with largest margin between classes.

1. Hard margin SVM: No misclassified data points

$$\begin{array}{ll} \min_{w\in\mathbb{R}^d, b\in\mathbb{R}} & \frac{1}{2}\|w\|_2^2 \\ st & y^{(i)}\cdot(x^{(i)T}w+b)\geq 1 \end{array}$$

2. Soft margin SVM: Allow errors

$$\begin{array}{lll} \min_{w\in\mathbb{R}^d, b\in\mathbb{R}, \xi\in\mathbb{R}^n} & \frac{1}{2}\|w\|_2^2+\frac{C}{n}\sum_{i=1}^{n}\xi_i & \\ st & y^{(i)}\cdot((x^{(i)T})w+b)\geq 1-\xi_i & ,\xi_i\geq 0 \end{array}$$

   This would be with e.g. the hinge loss (minimize regularizer + empirical risk $\rightarrow$ RRM):

$$\min_{w\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2}\|w\|_2^2+\frac{C}{n}\sum_{i=1}^{n}\max(0,1-y^{(i)}\cdot(x^{(i)T}w+b))$$

Duality: If Primal problem convex and some conditions satisfied, optimal solution is equal to dual problem's solution (Strong duality).

In dual problems we only need scalar products of data points.

3. Dual SVM:

$$\begin{array}{ll} \max_{\alpha\in\mathbb{R}^n} & \sum_{i=1}^{n}\alpha_i-\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y^{(i)}y^{(j)}(x^{(i)})^T x^{(j)} \\ st & \sum_{i=1}^{n}\alpha_i y^{(i)}=0 \\ & 0\leq\alpha_i\leq\frac{C}{n} \end{array}$$

   Used when many dimensions and hyperplane separator not linear and kernels are used.

2.8.2 Kernels

- $x \in \mathbb{R}^d$
- map to R^m using non-linear feature map ϕ
- Use linear SVM in R^m
- never compute $\phi(x)$, only scalar products
- use kernel function $k(x^{(i)}, x^{(j)})$

$k(a,b)$ is a kernel function...

- If ϕ exists for $k(a,b) = \phi(a)\phi(b)$ for $a, b \in \mathbb{R}$
- If ϕ exists for $K = \phi(X)\phi(X)^T$ with data matrix X
- Iff K is positive semidefinite ($K \succcurlyeq 0$) for any n input points
  - Sometimes ϕ maps to $\mathbb{R}^\infty$ (Reproducing Kernel Hilbert Space)

Examples

- Linear kernel for $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$
$$k(x^{(i)}, x^{(j)}) = (x^{(i)})^T x^{(j)}$$
- Cosine kernel: Assume data have all norm 1: $\left\| x^{(i)} \right\|_2 = 1$, angle measures similarity
$$\cos(x^{(i)}, x^{(j)}) = \frac{(x^{(i)})^T x^{(j)}}{\left\| x^{(i)} \right\| \left\| x^{(j)} \right\|} = (x^{(i)})^T x^{(j)}$$
- Gaussian kernel (Radial basis function)
$$k(x^{(i)}, x^{(j)}) = \exp\left(- \frac{\left\| x^{(i)} - x^{(j)} \right\|_2^2}{2\sigma^2} \right)$$
maps to infinite dimensions
- Polynomial Kernel for $c > 0$ and $s \in \mathbb{N}$:
$$k(x^{(i)}, x^{(j)}) = (x^{(i)})^T x^{(j)} + c)^s$$
- Sum and positive scalar product of kernels are kernels

Kernel for Regularized Risk Minimization if norm from 2:

- $\min_w L(w,X,y) + \lambda R(w)$
- iff $R = h(\|w\|_2)$ for non-decreasing h, then $w^* = \sum_i \beta_i \phi(x^{(i)})$
- LSR: $\min_w \|Xw - y\|_2^2 \Rightarrow \min_\beta \|K\beta - y\|_2^2$
- Ridge Regression: $\min_w \|Xw - y\|_2^2 + \lambda \|w\|_2^2 \Rightarrow \min_\beta \|K\beta - y\|_2^2 + \lambda \beta^T K \beta$
- Regularized Logistic Regression, etc.

2.9 Parametric vs Nonparametric Models

- Parametric if model depends on fixed number of parameters
- Nonparametric e.g. SVM with Gaussian kernel

2.10 Multiclass Classification

- One-vs-Rest: Train binary classifier for each class-pair, predict highest score class
- One-vs-One: Train binary classifier for each class-pair, predict class with most votes
- Direct methods: e.g. K-nearest neighbor

2.10.1 K-nearest neighbor

- Find k nearest neighbors in training data
- Predict class with most votes
- Choose k with cross-validation
- Small k $\rightarrow$ overfitting, large k $\rightarrow$ underfitting
- Advantage: Easy, simple, no training phase, non-parametric
- Disadvantage: Computationally expensive, sensitive to Noise

Regression: Compute average of k nearest neighbors

$$y = \frac{1}{k} \sum_{i \in S} y^{(i)}$$

or weighted

$$y = \frac{1}{k} \sum_{i \in S} \frac{1}{\|x - x^{(i)}\|} \cdot y^{(i)}$$

2.10.2 Naive Bayes

- Probabilistic ML algorithm
- Assume features are conditionally independent
- Compute $p(y \mid x) = \frac{p(x|y) \cdot p(y)}{p(x)}$
- $p(y \mid x)$ is event y in class x
- Bayesian theorem: $p(y \mid x) = \frac{p(x|y) \cdot p(y)}{p(x)}$
- Laplacian Smoothing for zero values: e.g. with words w_i of class x

$$p(w_i \mid x) = \frac{\text{frequency of } w_i + 1}{\text{words in } x + \text{total unique words}}$$

2.10.3 Decision Trees

- $X = \mathbb{R}^d$
- $Y = 1, 2, ..., m$
- Recursive partitioning of data
- Split data into subsets
- Choose feature and threshold to split (entropy, gini index)
- Repeat until stopping criterion
- Predict class with majority vote
- Advantage:
- Disadvantage: Overfitting, sensitive to noise

2.10.4 Ensemble Learning

- Combine multiple models to improve performance
- Bagging: Train multiple models on random subsets of data, smaller variance
- Bootstrapping: Randomly sample data with replacement
- Boosting: Train multiple models sequentially, each model corrects previous model

Random Forest

- Bagging with decision trees
- Each decision tree trained on random bootstrap subset of size m of data of size n
- Construct decision tree T_b:
  - For each node that contains more than n_{min} data points:
  - Randomly select p of d features
  - Split node with best feature and threshold
  - Repeat until all nodes smaller
- Output: Random decision trees $T_1, T_2, \dots, T_B$

3 Generative Models

- Discrimative: Learn $p(y \mid x)$
- Generative: Learn $p(y \mid x) \cdot p(x) = p(x, y)$ joint probability

3.1 Gaussian Discriminant Analysis

- generate new data, also for classification
- $\mu = \frac{1}{n} \sum_i x^{(i)}$
- $\Sigma = \frac{1}{n} \sum_i (x^{(i)} - \mu)(x^{(i)} - \mu)^T$
- for each class, get μ_c and Σ_c
- estimate $p(x, y) = p(x \mid y) \cdot p(y)$ with $p(y = c) = \frac{n_c}{n}$ where n_c is number of data points in class c and N is total amount of data points
- this becomes $x^T \Sigma x + v^T x + t = 0$
- Boundary is quadratic
- number of parameters to be estimated: $2\frac{d(d+1)}{2} + 2d$

3.2 Linear Discriminant Analysis

- LSR with $y \in \{-1, +1\}\}$: $\min_w \|Xw - y\|_2^2$
- numbers to be estimated: $d + 1$
- For generation:
  - Assume $\Sigma = \Sigma_{-1} = \Sigma_{+1}$ for all classes
  - number of parameters to be estimated: $\frac{d(d+1)}{2} + 2d$ (much higher than for discriminative models)

4 Supervised learning

- predict label for $(x^{(i)})_i$
- find structure, possibly assign labels
- can generate new data

4.1 k-means Objective

- for $x^{(i)} \in \mathbb{R}^d$ assume given data points $(x^{(i)})_{i=1}^n$
- group into clusters $C = \{C_1, C_2, \ldots, C_k\}$
- construct class k class centers μ_i representing groups
- Find centers such that sum of squared distances of data points to closest center is minimized:

$$\min_{\mu_1,\ldots,\mu_k \in \mathbb{R}^d, C} \sum_{j=1}^{k} \sum_{x^{(i)} \in C_j} \left\| x^{(i)} - \mu_j \right\|^2$$

- non-convex min. problem

4.1.1 LLoyds algorithm

1. Input: data points $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d$, number k of clusters
2. randomly init centers $\mu_1^{(0)}, \ldots, \mu_k^{(0)}$
3. while not converged:
   (a) assign each data point to closest center $C_1^{(i+1)}, \ldots, C_k^{(i+1)}$ with

   $$x^{(s)} \in C_j^{(i+1)} \Leftrightarrow \left\| x^{(s)} - \mu_j^{(i)} \right\|^2 \leq \left\| x^{(s)-\mu_l^i} \right\|^2, l = 1, \ldots, k$$

   ($x^{(s)}$ is assigned to cluster $C_j^{(i+1)}$ if it is closest to μ_j)

   (b) compute new cluster centers

   $$\mu_j^{(i+1)} = \frac{1}{|C_j^{(i+1)}|} \sum_{x^{(s)} \in C_j^{(i+1)}} x^{(s)}$$

4. Output: $C_1, \ldots, C_k$

This algorithm can be stuck in local optimum. Finding global optimum is NP-hard.

4.1.2 Alternatives

- k-median. $\|\cdot\|_2^2$ instead of $\|\cdot\|_2$
  - more robust to outliers
- weighted k-means: weights for individual data-points
- soft k-means: no hard assignments, but probabilities

4.2 Principal Component Analysis (PCA)

- Dimensionality reduction technique
- identify patterns and relationsships in high-D data
- many approaches, e.g.:
  - Maximize variance of projected data $\rightarrow$ covariance matrix
  - Minimize squared error $\rightarrow$ SVD

4.2.1 Maximize Variance

1. Input: Data points $x^{(1)},\dots,x^{(n)} \in \mathbb{R}^d$, parameter $l < d$ for reduced dimension
2. Output: Projection π_S on affine subspace S so variance of projected points is maximized:
$$\max_S \mathrm{Var}_l(\pi_S(X))$$
3. Assume data is centered: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x^{(i)} = 0$ or center by $\tilde{x}^{(i)} = x^{(i)} - \bar{x}$
4. Compute Covariance Matrix:
   - For one-dimensional case $l = 1$:
$$\begin{array}{rcl} & \max_{a\in\mathbb{R}^d} & \mathrm{Var}(\pi_a(X)) \\ & \text{st} & \|a\| = 1 \\ \Rightarrow & \max_{a\in\mathbb{R}^d} & \sum_{i=1}^{n}(\pi_a(x^{(i)}))^2 \\ & \text{st} & \|a\| = 1 \\ \Rightarrow & \max_{a\in\mathbb{R}^d} & \sum_{i=1}^{n}(a^T x^{(i)})^2 \\ & \text{st} & \|a\| = 1 \\ \Rightarrow & \max_{a\in\mathbb{R}^d} & \|Xa\|^2 \\ & \text{st} & \|a\| = 1 \\ \Rightarrow & \max_{a\in\mathbb{R}^d} & a^T(X^TX)a \\ & \text{st} & \|a\|^2 = 1 \end{array}$$
   - Solution is eigenvector v to largest eigenvalue λ_{max} of covariance matrix X^TX
   - for bigger projected dimensions $l > 1$ project data on space spanned by eigenvectors of the l largest eigenvalues of $C = X^TX$
5. Compute Eigendecomposition $C = V\Lambda V^T$
6. Define V_l as the matrix containing the l largest eigenvectors, i.e. first l columns of V if eigenvalzes in Λ are decreasing order
7. Compute new data points
   - $y^{(i)} = V_l^T \tilde{x}^{(i)} \in \mathbb{R}^l$ is the projection of $x^{(i)}$ on l-dimensional subspace
     - distance between point and projection is reconstruction error
     - Reconstruction error bound by $\sum_{k=l+1}^{d} \lambda_k$
   - $z^{(i)} = P\tilde{x}^{(i)} + \bar{x} \in \mathbb{R}^d$ with $P = V_l V_l^T$ to map points back to original d-dimensional space (they only span l-dimensional affine space now)

4.2.2 Kernel PCA

- With $x^{(i)}, \dots, x^{(n)}$ stacked in data matrix X as rows
- Covariance matrix is $C = X^T X \in \mathbb{R}^{d \times d}$ with d eigenvalues
- Kernel matrix is $K = XX^T \in \mathbb{R}^{n \times n}$ with n eigenvalues
- Express eigenvalues/eigenvectors of C by those of K and vice versa

4.3 Low-dimensional Embedding

- Given distance matrix $D \in \mathbb{R}^{n \times n}$ containing $d_{ij} = \left\| x_i - x_j \right\|$ between data points
- Recover $(x^{(i)})_{i=1,\dots,n} \in \mathbb{R}^d$
- find $\phi : X \to \mathbb{R}^d$ for $\left\| \phi(x^{(i)}) - \phi(x^{(j)}) \right\| = d_{ij}$
- if D from points in $\mathbb{R}^d$, D is Euclidean distance matrix (perfect ϕ computable)

4.3.1 Multidimensional Scaling (MDS)

- Given distance matrix D, find $x^{(i)}$ that best-match D
- Classic MDS: Minimize error wrt pairwise scalar product
- Metric MDS: Minimize error wrt distance matrix D
- Non-metric MDS: D is not a distance matrix but has ordering $\rightarrow$ preserve ordering
- non-convex, NP-hard

Classic MDS

1. The problem is:
$$\min_{x^{(1)},\dots,x^{(n)}} \sum_{i,j} (d_{i,j} - (x^{(i)})^T x^{(j)})^2$$
$$\Rightarrow \min_{X \in \mathbb{R}^{n \times d}} \left\| S - XX^T \right\|_{Fro}^2$$
2. Compute matrix $S = \frac{1}{2}(d_{1,i}^2 + d_{1,j}^2 + d_{i,j}^2)$
3. Compute eigenvalue decomposition $S = V\Lambda V^T$
4. V_d is matrix containing d largest eigenvectors of S
5. Λ_d is is $d \times d$ diagonal matrix with first d eigenvalues on diagonal
6. $X = V_d \sqrt{\Lambda_d}$
7. row i of X is coordinates of embedded point $x^{(i)}$

Note: If data points are in $\mathbb{R}^d$, S has rank d, and d eigenvalues > 0 and $n-d$ eigenvalues $= 0$. Spectrum of S provides information on d.

(Non-)Metric MDS

- Metric MDS:
$$\min_{x^{(1)},\dots,x^{(n)}} \sum_{i,j} (\left\| x^{(i)} - x^{(j)} \right\| - d_{i,j})^2$$
- Non-Metric MDS: Distance matrix consists of $d_{i,j} > d_{i,k}$:
$$\min_{x^{(1)},\dots,x^{(n)}} \sum_{i,j} (f(\left\| x^{(i)} - x^{(j)} \right\|) - d_{i,j})^2$$

4.4 Isomap

- Detect lower-dimensional structures in data
- Keep local distances of points
- Compute geodesic distances for large distances with k-nearest neighbor graph, shortest path distances approximate geodesic distances
- Store distances in distance matrix D
- Metric MDS on D (preserves geodesic distances)

4.5 Recommender Systems

- Fill out missing values in data matrix
- For U, V, we can compute whole rating matrix $M = U \cdot V^T$
- Sparse matrix R that matches predictions
- Goal: Find U, V that predicition matches input ratings

$$\begin{aligned}
&\min_{U\in\mathbb{R}^{m\times k}, V\in\mathbb{R}^{n\times k}} \sum_{(i,j)\in\Omega} (r_{i,j} - (u^{(i)})^T v^{(j)})^2 \\
=&\min_{U\in\mathbb{R}^{m\times k}, V\in\mathbb{R}^{n\times k}} \left\|R - UV^T\right\|_\Omega^2 \qquad \text{where} \qquad \|A\|_\Omega^2 = \sum_{(i,j)\in\Omega} a_{i,j}^2 \\
=&\min_{U\in\mathbb{R}^{m\times k}, V\in\mathbb{R}^{n\times k}} \left\|R - UV^T\right\|_{Fro}^2
\end{aligned}$$

- Non-convex, NP-hard
- Gradient descent or alternating least squares (fix U or V and optimize the other, then switch)

4.6 Topic Modelling

- Word frequency transformation
- rare and frequent words discarded
- $w_{i,j}$ means how often word j appears in document i
- Term Frequency TF $= \frac{w_{i,j}}{\text{length of document i}}$
- Inverse Document Frequency IDF $= \log \frac{N}{\text{number of documents containing word j}}$
- smoothed IDF: IDF $= \log \frac{N+1}{\text{number of documents containing word j+1}} + 1$
- TF-IDF: TF-IDF = TF $\cdot$ IDF
- in Matrix W store TF-IDF values
- Kullback-Leibler divergence computation on W to find topics
- Gets feature vectors for words, can assign topics to words and documents

4.7 Matrix Factorization

- Useful for images
- Uncovers latent factors (topics)
- Imputes data