PCML Cheat Sheet

Contents

1 Math Prerequisites

- Bayes rule

$$p(A,B) = \underbrace{p(A|B)}_{\text{Lik.}}\underbrace{p(B)}_{\text{Prior}} = \underbrace{p(B|A)}_{\text{Post}}\underbrace{p(A)}_{\text{Marg. Lik.}}$$

- Gaussian distribution

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{2\pi|\boldsymbol{\Sigma}|}}\exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))$$

- Production of independent variables:

$$\mathbb{V}(XY) = \mathbb{E}(X^2)\,\mathbb{E}(Y^2) - [\mathbb{E}(X)]^2[\mathbb{E}(Y)]^2$$

- Covariance matrix of a data vector $\mathbf{x}$

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \mathbb{E}(\mathbf{x}))(\mathbf{x}_n - \mathbb{E}(\mathbf{x}))^T$$

1.1 Convexity

- A function $f(x)$ is convex, if for any $x_1, x_2 \in \mathbf{X}$ and for any $0 \leq \lambda \leq 1$, we have :

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$$

- Strictly convex if the inequality is strict.
- Sums of convex functions are also convex.
- A function with the form log-sum-exp is convex.
- The Hessian of a convex function is positive semi-definite and for a strictly-convex function it is positive definite.

$$\mathbf{H}_{i,j} = d^2 f/dx_i dx_j$$

1.2 Linear Algebra

- Column $\mathbf{x} \in^n$, rows $\mathbf{x}^T$, matrix $\mathbf{A} \in^{m\times n}$
- $\mathbf{x}^T\mathbf{x}$ is a scalar, $\mathbf{x}\mathbf{x}^T$ is a matrix
- $\mathbf{A}^{-1}$ exist if $\mathbf{A}$ is full rank
- $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$
- **Condition number** of a function measures how much the output value can change for a small change in the input. A matrix with a high condition number is said to be **ill-conditioned**. If $\mathbf{A}$ is normal ($A^TA = AA^T$) then

$$k(\mathbf{A}) = \left|\frac{\lambda_{max}(\mathbf{A})}{\lambda_{min}(\mathbf{A})}\right|$$

- A positive definite matrix is **symmetric** with all positive eigenvalues
- The real symmetric $N \times N$ matrix $\mathbf{V}$ is said to be **positive semidefinite** if

$$\mathbf{a}^T\mathbf{V}\mathbf{a} \geq 0$$

  for any real $N \times 1$ vector a.
- **positive definite** if $\mathbf{a}^T\mathbf{V}\mathbf{a} > 0$
- Cost of matrix inversion: $O(n^3) \to O(n^{2.372})$
- $det(\mathbf{A})$ using LU decomposition: $O(n^3)$

2 Cost functions

- Cost functions are used to learn parameters that explain the data well.
- It is essential to make sure that a global minimum exist $\to$ lower bounded

Mean square error (MSE):

$$MSE(\boldsymbol{w}) = \sum_{n=1}^{N}(y_n - f(\mathbf{x}_n))^2$$

- MSE is **convex** thus it has only one global minumum value.
- MSE is not good when outliers are present.

Mean Absolute Error (MAE):

$$MAE = \sum_{n=1}^{N}|y_n - f(\mathbf{x}_n)|$$

Huber loss

$$Huber = \begin{cases} \frac{1}{2}z^2 & ,|z| \leq \delta \\ \delta|z| - \frac{1}{2}\delta^2 & ,|z| > \delta \end{cases}$$

- Huber loss is convex, differentiable, and also robust to outliers but hard to set δ.

Tukey's bisquare loss

$$L(z) = \begin{cases} z(\delta^2 - z^2)^2 & ,|z| < \delta \\ 0 & ,|z| \geq \delta \end{cases}$$

Non-convex, non-diff., but robust to outliers.

Hinge loss

$$Hinge = [1 - y_n f(\mathbf{x}_n)]_+ = \max(0, 1 - y_n f(\mathbf{x}_n))$$

Logistic loss

$$Logistic = \log(1 - \exp(y_n f(\mathbf{x}_n)))$$

3 Regression

- **Data** consists of N pairs $(y_n, \mathbf{x}_n)$
  1. y_n the n'th output
  2. $\mathbf{x}_n$ is a vector of D inputs
- **Prediction**: predict the ouput for a new input vector.
- **Interpretation**: understand the effect of inputs on output.
- **Outliers** are data that are far away from most of the other examples.

3.1 Linear Regression

- Model that assume linear relationship between inputs and the ouput.

$$y_n \approx f(\mathbf{x}_n) := w_0 + w_1 x_{n1} + ... =_0 + \mathbf{x}_n^T\boldsymbol{w}$$

  with $\boldsymbol{w}$ the parameters of the model.
- Variance grows only linearly with dimensionality

4 Optimization

4.1 Grid search

- Compute the cost over a grid of M points to find the minimum
- Exponential Complexity

– Hard to find a good range of values

4.2 Gradient Descent

– Gradient descent uses only first-order information and takes steps in the direction of the gradient
– Given a cost function $\mathcal{L}(\boldsymbol{w})$ we wish to find $\boldsymbol{w}$ that minimizes the cost:

$$\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})$$

4.3 Batch Gradient Descent

– Take steps in the opposite direction of the gradient

$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \gamma \boldsymbol{\nabla} \mathcal{L}(\boldsymbol{w}^{(t)})$$

with $\gamma > 0$ the learning rate.
– With γ too big, method might diverge. With γ too small, convergence is slow.

4.4 Gradients for MSE

– We define the error vector $\mathbf{e}$:

$$\mathbf{e} := \mathbf{y} - \mathbf{X}\boldsymbol{w}$$

– and MSE as follows:

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{2N}\sum_{n=1}^{N}(y_n - \tilde{\mathbf{x}}_n^T \boldsymbol{w})^2 = \frac{1}{2N}\mathbf{e}^T\mathbf{e}$$

– then the gradient is given by

$$\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{w}) = -\frac{1}{N}\mathbf{X}^T\mathbf{e}$$

– Optimality conditions:
1. *necessary*: gradient equal zero: $\frac{d\mathcal{L}(\boldsymbol{w}^*)}{d\boldsymbol{w}} = 0$
2. *sufficient*: Hessian matrix is positive definite: $\mathbf{H}(\boldsymbol{w}^*) = \frac{d^2\mathcal{L}(\boldsymbol{w}^*)}{d\boldsymbol{w}d\boldsymbol{w}^T}$

– Very sensitive to illconditioning. Therefore, always normalize your feature otherwise step-size selection is difficult since different directions might move at different speed.
– *Complexity*: $O(NDI)$ with I the number of iterations

4.5 Stochastic Gradient Descent

In ML, most cost functions are formulated as a sum over the training examples:

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}_n(\boldsymbol{w})$$

$\Rightarrow$ SGD algo is given by update rule:

$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \gamma \boldsymbol{\nabla}\mathcal{L}_n(\boldsymbol{w}^{(t)})$$

Idea: Cheap but unbiased estimate of grad.

$$\mathbb{E}[\boldsymbol{\nabla}\mathcal{L}_n(\boldsymbol{w})] = \boldsymbol{\nabla}(\boldsymbol{w})$$

4.6 Mini-batch SGD

Update direction ($B \subseteq [N]$):

$$\boldsymbol{g}^{(t)} := \frac{1}{|B|}\sum_{n\in B}\boldsymbol{\nabla}\mathcal{L}_n(\boldsymbol{w}^{(t)})$$

Update rule : $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \gamma \boldsymbol{g}^{(t)}$

4.7 Subgradients (Non-Smooth OPT)

A vector $\boldsymbol{g} \in^D$ s.t.

$$\mathcal{L}(\boldsymbol{u}) \geq \mathcal{L}(\boldsymbol{w}) + \boldsymbol{g}^T(\boldsymbol{u} - \boldsymbol{w}) \quad \forall \boldsymbol{u}$$

is the subgradient to $\mathcal{L}$ at $\boldsymbol{w}$. If $\mathcal{L}$ is differentiable at $\boldsymbol{w}$, we have $\boldsymbol{g} = \boldsymbol{\nabla}\mathcal{L}(\boldsymbol{w})$

5 Least Squares

– In some cases, we can compute the minimum of the cost function analytically.
– use the first optimality conditions:

$$\boldsymbol{\nabla} L(\boldsymbol{w}^*) = 0 \Rightarrow \mathbf{X}^T\mathbf{e} = \mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{w}) = 0$$

– When $\mathbf{X}^T\mathbf{X}$ is invertible, we have the closed-form expression

$$\boldsymbol{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

– thus we can predict values for a new $\mathbf{x}_m$

$$y_m := \mathbf{x_m^T}\boldsymbol{w}^* = \mathbf{x_m^T}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

– The **Gram matrix** $\mathbf{X}^T\mathbf{X}$ is positive definite and is also invertible iff $\mathbf{X}$ has full column rank.
– *Complexity*: $O(ND^2 + D^3) \equiv O(ND^2)$
– $\mathbf{X}$ can be rank deficient when $D > N$ or when the comlumns $\bar{\mathbf{x}}_d$ are nearly collinear. In this case, the matrix is ill-conditioned, leading to numerical issues.

6 Maximum Likelihood

– Let define our mistakes $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$.

$$\rightarrow y_n = \mathbf{x}_n^T\boldsymbol{w} + \epsilon_n$$

– Another way of expressing this:

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \mathbf{w}) &= \prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \boldsymbol{w}) \\ &= \prod_{n=1}^{N}\mathcal{N}(y_n|\mathbf{x}_n^T\boldsymbol{w}, \sigma^2) \end{aligned}$$

which defines the likelihood of observing $\mathbf{y}$ given $\mathbf{X}$ and $\boldsymbol{w}$
– Define cost with log-likelihood

$$\begin{aligned} \mathcal{L}_{lik}(\boldsymbol{w}) &= \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{w}) \\ &= -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^T\boldsymbol{w})^2 + cnst \end{aligned}$$

– Maximum likelihood estimator (MLE) gives another way to design cost functions

$$\operatorname*{argmin}_{\boldsymbol{w}} \mathcal{L}_{MSE}(\boldsymbol{w}) = \operatorname*{argmax}_{\boldsymbol{w}} \mathcal{L}_{lik}(\boldsymbol{w})$$

– MLE can also be interpreted as finding the model under which the observed data is most likely to have been generated from.
– With Laplace distribution

$$p(y_n|\mathbf{x}_n, \boldsymbol{w}) = \frac{1}{2b}e^{-\frac{1}{b}|y_n - \mathbf{x}_n^T\boldsymbol{w}|}$$

$$\sum_n \log p(y_n|\mathbf{x}_n, \boldsymbol{w}) = \sum_n |y_n - \mathbf{x}_n^T \boldsymbol{w}| + cnst$$

7 Ridge Regression

– Linear models usually underfit. One way is to use nonlinear basis functions instead.

$$y_n = w_0 + \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}_n) = \tilde{\boldsymbol{\phi}}(\mathbf{x}_n)^T \boldsymbol{w}$$

– This model is linear in $\boldsymbol{w}$ but nonlinear in $\mathbf{x}$. Note that the dimensionality is now M, not D.
– Polynomial basis

$$\boldsymbol{\phi}(x_n) = [1, x_n, x_n^2, ..., x_n^M]$$

– The least square solution becomes

$$\boldsymbol{w}_{lse}^* = (\tilde{\boldsymbol{\Phi}}^T \tilde{\boldsymbol{\Phi}})^{-1} \tilde{\boldsymbol{\Phi}}^T \mathbf{y}$$

– Complex models overfit easily. Thus we can choose simpler models by adding a **regularization term** which penalizes complex models

$$\min_{\boldsymbol{w}} \left(\mathcal{L}(\boldsymbol{w}) + \frac{\lambda}{2N} \sum_{j=1}^{M} w_j^2 \right)$$

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} \left(\frac{1}{2} (\mathbf{y} - \mathbf{X}\boldsymbol{w})^T (\mathbf{y} - \mathbf{X}\boldsymbol{w}) + \frac{\lambda}{2} \boldsymbol{w}^T \boldsymbol{w} \right)$$

– Note that w_0 is not penalized.
– By differentiating and setting to zero we get

$$\boldsymbol{w}_{ridge} = (\tilde{\boldsymbol{\Phi}}^T \tilde{\boldsymbol{\Phi}} + \boldsymbol{\Lambda})^{-1} \tilde{\boldsymbol{\Phi}}^T \mathbf{y}$$

$$\boldsymbol{\Lambda} = \begin{bmatrix} 0 & \underline{0} \\ \underline{0} & \lambda I_m \end{bmatrix}$$

– Ridge regression improves the condition number of the Gram matrix since the eigenvalues of $(\tilde{\boldsymbol{\Phi}}^T \tilde{\boldsymbol{\Phi}} + \lambda I_m)$ are at least λ
– **Maximum-a-posteriori (MAP) estimator**:
  – Maximizes the product of the likelihood *and* the **prior**.

$$\boldsymbol{w}_{MAP} = \operatorname*{argmax}_{\boldsymbol{w}} \left(p(\mathbf{y}|\mathbf{X}, \boldsymbol{\Lambda}) p(\boldsymbol{w}|\boldsymbol{\Sigma}) \right)$$

  – Assume $w_0 = 0$

$$\boldsymbol{w}_{ridge} = \operatorname*{argmax}_{\boldsymbol{w}} \left(\log \left[\prod_{n=1}^{N} \mathcal{N}(y_n|\mathbf{x}_n^T \boldsymbol{w}, \boldsymbol{\Lambda}) \times \mathcal{N}(\boldsymbol{w}|0, \mathbf{I}) \right] \right)$$

– **Lasso regularizer** forces some w_i to be strictly 0 and therefore forces sparsity in the model.

$$\min_{\boldsymbol{w}} \frac{1}{2N} \sum_{n=1}^{N} (y_n - \tilde{\boldsymbol{\phi}}(\mathbf{x}_n)^T \boldsymbol{w})^2,$$

$$\text{such that } \sum_{i=1}^{M} |w_i| \leq \tau$$

8 Cross-Validation

– We should choose λ to minimize the mistakes that will be made in the future.
– We split the data into train and validation sets and we pretend that the validation set is the future data. We fit our model on the training set and compute a prediction-error on the validation set. This gives us an *estimate* of the *generalization error*.
– **K-fold cross validation** randomly partition the data into K groups. We train on $K-1$ groups and test on the remaining group. We repeat this until we have tested on all K sets. We then average the results.
– Cross-validation returns an unbiased estimate of the generalization error and its variance.

9 Bias-Variance decomposition

– The expected test error can be expressed as the sum of two terms
  – **Squared bias**: The average *shift* of the predictions
  – **Variance**: measure how data points vary around their average.

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

– Both model bias and estimation bias are important
– Ridge regression increases estimation bias while reducing variance
– Increasing model complexity increases test error
– Small $\lambda \to$ low bias but large variance
– Large $\lambda \to$ large bias but low variance

$$err = \sigma^2 + E[f_{lse} - E[f_{lse}]]^2 + [f_{true} - E[f_{lse}]]^2$$

10 Logistic Regression

– **Classification** relates input variables $\mathbf{x}$ to discrete output variable y
– **Binary classifier**: we use $y = 0$ for $\mathbf{C}_1$ and $y = 1$ for $\mathbf{C}_2$.
– Can use least-squares to predict $\hat{y}_*$

$$\hat{y} = \begin{cases} \mathbf{C}_1 & \hat{y}_* < 0.5 \\ \mathbf{C}_2 & \hat{y}_* \geq 0.5 \end{cases}$$

– **Logistic function**

$$\sigma(x) = \frac{\exp(x)}{1 + \exp(x)}$$

$$p(y_n = \mathbf{C}_1|\mathbf{x}_n) = \sigma(\boldsymbol{x}^T \boldsymbol{w})$$
$$p(y_n = \mathbf{C}_2|\mathbf{x}_n) = 1 - \sigma(\boldsymbol{x}^T \boldsymbol{w})$$

– The probabilistic model:

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{w}) = \prod_{n=1}^{N} \sigma(\boldsymbol{x}_n^T \boldsymbol{w})^{y_n} (1 - \sigma(\boldsymbol{x}_n^T \boldsymbol{w}))^{1-y_n}$$

- The log-likelihood:

$$\mathcal{L}_{MLE}(\boldsymbol{w}) = \sum_{n=1}^{N} \left(y_n \boldsymbol{x}_n^T \boldsymbol{w} - \log(1 + \exp(\boldsymbol{x}_n^T \boldsymbol{w})) \right)$$

- We can use the fact that

$$\frac{d}{dx} \log(1 + \exp(x)) = \sigma(x)$$

- Gradient of the log-likelihood

$$\begin{aligned} \mathbf{g} = \frac{d\mathcal{L}}{d\boldsymbol{w}} &= \sum_{n=1}^{N} \left(\boldsymbol{x}_n y_n - \boldsymbol{x}_n \sigma(\boldsymbol{x}_n^T \boldsymbol{w}) \right) \\ &= -\mathbf{X}^T [\sigma(\mathbf{X}\boldsymbol{w}) - \mathbf{y}] \end{aligned}$$

- The negative of the log-likelihood $-\mathcal{L}_{mle}(\boldsymbol{w})$ is convex
- **Hessian** of the log-likelihood
  - We know that

$$\frac{d\sigma(t)}{dt} = \sigma(t)(1 - \sigma(t))$$

  - Hessian is the derivative of the gradient

$$\begin{aligned} \mathbf{H}(\boldsymbol{w}) = -\frac{d\mathbf{g}(\boldsymbol{w})}{d\boldsymbol{w}^T} &= \sum_{n=1}^{N} \frac{d}{d\boldsymbol{w}^T} \sigma(\boldsymbol{x}_n^T \boldsymbol{w}) \boldsymbol{x}_n \\ &= \sum_{n=1}^{N} \boldsymbol{x}_n \sigma(\boldsymbol{x}_n^T \boldsymbol{w})(1 - \sigma(\boldsymbol{x}_n^T \boldsymbol{w})) \boldsymbol{x}_n^T \\ &= \tilde{\mathbf{X}}^T \mathbf{S} \tilde{\mathbf{X}} \end{aligned}$$

  where $\mathbf{S}$ is a $N \times N$ diagonal matrix with diagonals

$$S_{nn} = \sigma(\boldsymbol{x}_n^T \boldsymbol{w})(1 - \sigma(\boldsymbol{x}_n^T \boldsymbol{w}))$$

  - The negative of the log-likelihood is not strictly convex.
- **Newton's Method**
  - Uses second-order information and takes steps in the direction that minimizes a quadratic approximation

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}) = \mathcal{L}(\boldsymbol{w}^{(k)}) &+ \mathbf{g}_k^T (\boldsymbol{w} - \boldsymbol{w}^{(k)}) \\ &+ (\boldsymbol{w} - \boldsymbol{w}^{(k)})^T \mathbf{H}_k (\boldsymbol{w} - \boldsymbol{w}^{(k)}) \end{aligned}$$

  and it's minimum is at

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^{(k)} - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}_k$$

  where $\mathbf{g}_k$ is the gradient and α_k the learning rate.
  - Complexity: $O((ND^2 + D^3)I)$
- **Penalized Logistic Regression**

$$\min_{\boldsymbol{w}} \left(-\sum_{n=1}^{N} \log p(y_n | \mathbf{x}_n^T \boldsymbol{w}) + \lambda \sum_{d=1}^{D} w_d^2 \right)$$

11 Generalized Linear Model

- **Exponential family distribution**

$$p(\mathbf{y}|\boldsymbol{\eta}) = \frac{h(y)}{Z} \exp(\boldsymbol{\eta}^T \boldsymbol{\phi}(\mathbf{y}) - A(\boldsymbol{\eta}))$$

  - Bernoulli distribution

$$\begin{aligned} p(y|\mu) &= \mu^y (1 - \mu)^{1-y} \\ &= \exp(y \log(\frac{\mu}{1 - \mu} + \log(1 - \mu))) \end{aligned}$$

  - there is a relationship between η and μ throught the **link function**

$$\eta = \log(\frac{\mu}{1 - \mu}) \leftrightarrow \mu = \frac{e^\eta}{1 + e^\eta}$$

  - Note that μ is the mean parameter of y
- Relationship between the mean $\boldsymbol{\mu}$ and $\boldsymbol{\eta}$ is defined using a link function g

$$\boldsymbol{\eta} = \mathbf{g}(\boldsymbol{\mu}) \Leftrightarrow \boldsymbol{\mu} = \mathbf{g}^{-1}(\boldsymbol{\eta})$$

- First and second derivatives of $A(\eta)$ are related to the mean and the variance

$$\frac{dA(\eta)}{d\eta} = \mathbb{E}[\boldsymbol{\phi}(\eta)], \quad \frac{d^2 A(\eta)}{d\eta^2} = \mathbb{V}[\boldsymbol{\phi}(\eta)]$$

- $A(\eta)$ is convex
- The generalized maximum likelihood cost to minimize is

$$\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) = -\sum_{n=1}^{N} \log(p(y_n | \boldsymbol{x}_n^T \boldsymbol{w}))$$

  where $p(y_n | \boldsymbol{x}_n^T \boldsymbol{w})$ is an exponential family distribution
- We obtain the solution

$$\frac{d\mathcal{L}}{d\boldsymbol{w}} = \mathbf{X}^T [\mathbf{g}^{-1}(\boldsymbol{\eta}) - \boldsymbol{\phi}(\mathbf{y})]$$

12 k-Nearest Neighbor (k-NN)

- The k-NN prediction for $\mathbf{x}$ is

$$f(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}_n \in nbh_k(\mathbf{x})} y_n$$

  where $nbh_k(\mathbf{x})$ is the neightborhood of $\mathbf{x}$ defined by the k closest points $\mathbf{x}_n$ in the training data
- **Curse of dimensionality**: Generalizing correctly becomes exponentially harder as the dimensionality grows.
- Gathering more inputs variables may be a bad thing

13 Support Vector Machine

- Combination of the kernel trick plus a modified loss function (Hinge loss)
- Solution to the dual problem is sparse and non-zero entries will be our **support vectors**.
- **Kernelised feature vector** where $\boldsymbol{\mu}_k$ are centroids

$$\boldsymbol{\phi}(\mathbf{x}) = [k(\mathbf{x}, \boldsymbol{\mu}_1), ..., k(\mathbf{x}, \boldsymbol{\mu}_K)]$$

- In practice we'll take a subset of data points to be prototype $\rightarrow$ **sparse vector machine**.
- Assume $y_n \in \{-1, 1\}$
- SVM optimizes the following cost

$$\mathcal{L}(\boldsymbol{w}) = \min_{\boldsymbol{w}} \sum_{n=1}^{N} [1 - y_n \tilde{\boldsymbol{\phi}}_n^T \boldsymbol{w}]_+ + \frac{\lambda}{2} \sum_{j=1}^{M} w_j^2$$

- Minimum doesn't change with a rescaling of $\boldsymbol{w}$
- choose the hyperplane so that the distance from it to the nearest data point on each side is maximized
- **Duality**:
  - Hard to minimize $g(\boldsymbol{w})$ so we define

$$\mathcal{L}(\boldsymbol{w}) = \max_{\boldsymbol{\alpha}} G(\boldsymbol{w}, \boldsymbol{\alpha})$$

  - we use the property that

$$C[v_n]_+ = \max(0, Cv_n) = \max_{\alpha_n \in [0,C]} \alpha_n v_n$$

  - We can rewrite the problem as

$$\min_{\boldsymbol{w}} \max_{\boldsymbol{\alpha} \in [0,C]^N} \sum_{n=1}^{N} \alpha_n (1 - y_n \boldsymbol{\phi}_n^T \boldsymbol{w}) + \frac{1}{2} \sum_{j=1}^{M} w_j^2$$

  - This is differentiable, convex in $\boldsymbol{w}$ and concave in $\boldsymbol{\alpha}$
  - **Minimax theorem**:

$$\min_{\boldsymbol{w}} \max_{\boldsymbol{\alpha}} G(\boldsymbol{w}, \boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}} \min_{\boldsymbol{w}} G(\boldsymbol{w}, \boldsymbol{\alpha})$$

    because G is convex in $\boldsymbol{w}$ and concave in $\boldsymbol{\alpha}$.
  - Derivative w.r.t. $\boldsymbol{w}$:

$$\boldsymbol{\nabla}_{\boldsymbol{w}} G(\boldsymbol{w}, \boldsymbol{\alpha}) = -\sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n + \lambda \boldsymbol{w}$$

  - Equating this to 0, we get:

$$\boldsymbol{w}(\boldsymbol{\alpha}) = \frac{1}{\lambda} \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n = \frac{1}{\lambda} \mathbf{X} \mathbf{Y} \boldsymbol{\alpha}$$
$$\mathbf{Y} := \text{diag}(\boldsymbol{y})$$

  - Plugging $\boldsymbol{w}^*$ back in the dual problem

$$\max_{\boldsymbol{\alpha} \in [0,1]^N} \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2\lambda} \boldsymbol{\alpha}^T \mathbf{Y} \boldsymbol{X}^T \boldsymbol{X} \mathbf{Y} \boldsymbol{\alpha}$$

  - This is a differentiable least-squares problem. Optimization is easy using Sequential Minimal Optimization. It is also naturally kernelized with $\mathbf{K} = \boldsymbol{X}^T \boldsymbol{X}$
  - The solution $\boldsymbol{\alpha}$ is sparse and is non-zero only for the training examples that are instrumental in determining the decision boundary.

14 Kernel Ridge Regression

- The following is true for ridge regression

$$\begin{aligned} \boldsymbol{w}^* &= (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_D)^{-1} \mathbf{X}^T \mathbf{y} \\ &= \mathbf{X}^T (\mathbf{X} \mathbf{X}^T + \lambda \mathbf{I}_N)^{-1} \mathbf{y} = \mathbf{X}^T \boldsymbol{\alpha}^* \end{aligned} \tag{1}$$

- Complexity of computing $\boldsymbol{w}$: (1) $O(D^2 N + D^3)$, (2) $O(DN^2 + N^3)$
- Thus we have

$$\boldsymbol{w}^* = \mathbf{X} \boldsymbol{\alpha}^*, \quad \text{with } \boldsymbol{w}^* \in^D \text{ and } \boldsymbol{\alpha}^* \in^N$$

- The representer theorem allows us to write an equivalent optimization problem in terms of $\boldsymbol{\alpha}$.

$$\boldsymbol{\alpha} = \underset{\boldsymbol{\alpha}}{\text{argmax}} \left(-\frac{1}{2} \boldsymbol{\alpha}^T (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_N) \boldsymbol{\alpha} + \boldsymbol{\alpha}^T \mathbf{y} \right)$$

- $\mathbf{K} = \mathbf{X} \mathbf{X}^T$ is called the **kernel matrix** or **Gram matrix**.
- If $\mathbf{K}$ is positive definite, then it's called a **Mercer Kernel**.
- $\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$
- If the kernel is Mercer, then there exists a function $\boldsymbol{\phi}(\mathbf{x})$ s.t.

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{x}')$$

- **Kernel trick**:
  - We can work directly with $\mathbf{K}$ and never have to worry about $\mathbf{X}$
  - Replace $\langle \mathbf{x}, \mathbf{x}' \rangle$ with $k(\mathbf{x}, \mathbf{x}')$.
  - Kernel function can be interpreted as a measure of similarity
  - The evaluation of a kernel is usually faster with k than with $\boldsymbol{\phi}$
- Kernelized rigde regression might be computationally more efficient in some cases.
- **Radial Basis function kernel (RBF)**

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\frac{1}{2} (\mathbf{x} - \mathbf{x}')^T (\mathbf{x} - \mathbf{x}'))$$

- Properties of a kernel to ensure the existance of a corresponding $\boldsymbol{\phi}$:
  - $\mathbf{K}$ should be symmetric: $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$
  - $\mathbf{K}$ should be positive semidefinite.
- Thus we get

$$\mathbf{y} = \boldsymbol{w}^T \mathbf{x} = \sum_{i=1}^{K} \alpha_i \mathbf{x}_i^T \mathbf{x} = \sum_{i=1}^{K} \alpha_i k(\mathbf{x}, \mathbf{x}_i)$$

15 K-means

- **Unsupervised learning**: Represent particular input patterns in a way that reflects the statistical structure of the overall collections of input partterns.
- **Cluster** are groups of points whose inter-point distances are small compared to the distances outside the cluster.

$$\min_{\mathbf{z}, \boldsymbol{\mu}} \mathcal{L}(\mathbf{z}, \boldsymbol{\mu}) = \sum_{k=1}^{K} \sum_{n=1}^{N} z_{nk} ||\mathbf{x}_n - \boldsymbol{\mu}_k||_2^2$$

  such that $z_{nk} \in \{0, 1\}$ and $\sum_{k=1}^{K} z_{nk} = 1$
- K-means algorithm (Coordinate Descent):
  Initialize $\boldsymbol{\mu}_k$, then iterate

1. For all n, compute $\mathbf{z}_n$ given $\boldsymbol{\mu}$

$$z_{nk} = \begin{cases} 1 & \text{if } k = \text{argmin}_j \, ||\mathbf{x}_n - \boldsymbol{\mu}||_2^2 \\ 0 & \text{otherwise} \end{cases}$$

2. For all k, compute μ_k given $\mathbf{z}$

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N z_{nk}\mathbf{x}_n}{\sum_{n=1}^N z_{nk}}$$

- A good initialization procedure is to choose the prototypes to be equal to a random subset of K data points.
- Probabilistic model

$$p(\mathbf{z}, \boldsymbol{\mu}) = \prod_{n=1}^N \prod_{k=1}^K [\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \mathbf{I})]^{z_{nk}}$$

- K-means as a Matrix Factorization

$$\min_{\mathbf{z},\boldsymbol{\mu}} \mathcal{L}(\mathbf{z}, \boldsymbol{\mu}) = ||\mathbf{X} - \mathbf{M}\mathbf{Z}^T||_{\text{Frob}}^2$$

- Computation can be heavy, each example can belong to only on cluster and clusters have to be spherical.

16 Gaussian Mixture Models

- Clusters can be elliptical using a full covariance matrix instead of isotropic covariance.

$$p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{z}) = \prod_{n=1}^N \prod_{k=1}^K [\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)]^{z_{nk}}$$

- **Soft-clustering**: Points can belong to several cluster by defining z_n to be a random variable.

$$p(z_n = k) = \pi_k \text{ where } \pi_k > 0, \forall k, \sum_{k=1}^K \pi_k = 1$$

- Joint distribution of Gaussian mixture model

$$\begin{aligned} p(\mathbf{X}, \mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) &= \prod_{n=1}^N p(\mathbf{x}_n|\mathbf{r}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma})p(\mathbf{z}_n|\boldsymbol{\pi}) \\ &= \prod_{n=1}^N \prod_{k=1}^K [(\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k))^{z_{nk}}] \prod_{k=1}^K [\pi]^{z_{nk}} \end{aligned}$$

- z_n are called *latent* unobserved variables
- Unknown parameters are given by $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}\}$
- We get the **marginal likelihood** by marginalizing z_n out from the likelihood

$$\begin{aligned} p(\mathbf{x}_n|\boldsymbol{\theta}) &= \sum_{k=1}^K p(\mathbf{x}_n, z_n = k|\boldsymbol{\theta}) \\ &= \sum_{k=1}^K p(z_n = k|\boldsymbol{\theta})p(\mathbf{x}_n|z_n = k, \boldsymbol{\theta}) \\ &= \sum_{k=1}^K \pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \end{aligned}$$

- Without a latent variable model, number of parameters grow at rate $O(N)$
- After marginalization, the growth is reduced to $O(D^2K)$
- To get maximum likelihood estimate of $\boldsymbol{\theta}$, we maximize

$$\max_{\boldsymbol{\theta}} \sum_{n=1}^N \log \sum_{k=1}^K \pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

17 Expectation Maximization Algorithm

- *[ALGORITHM]* Start with $\boldsymbol{\theta}^{(1)}$ and iterate

1. *Expectation step*: Compute a lower bound to the cost such that it is tight at the previous $\boldsymbol{\theta}^{(t)}$ with equality when,

$$q_{kn} = \frac{\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^K \pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

2. *Maximization step*: Update $\boldsymbol{\theta}$

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{\text{argmax}}\, \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$$

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{n=1}^N \gamma^{(i)}(r_{nk})\mathbf{x}_n}{\sum_{n=1}^N q_{kn}^{(t)}}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{\sum_{n=1}^N q_{kn}^{(t)}(\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})^T}{\sum_{n=1}^N q_{kn}^{(t)})}$$

$$\pi_k^{(t+1)} = \frac{1}{N}\sum_{n=1}^N q_{kn}^{(t)}$$

- If covariance is diagonal $\rightarrow$ K-means.

18 Matrix factorization

- We have D movies and N users
- $\mathbf{X}$ is a matrix $D \times N$ with x_{dn} the rating of n'th user for d'th movie.
- We project data vectors $\mathbf{x}_n$ to a smaller dimension $\mathbf{z}_n \in \mathbb{R}^M$
- We have now 2 latent variables:
  - $\mathbf{Z}$ a $N \times K$ matrix that gives features for the users
  - $\mathbf{W}$ a $D \times K$ matrix that gives features for the movies

$$x_{dn} \approx \mathbf{w}_d^T \mathbf{z}_n$$

- We can add a regularizer and minimize the following cost:

$$\mathcal{L}(\mathbf{W}, \mathbf{Z}) = \frac{1}{2} \sum_{(d,n)\in\Omega} [x_{dn} - (\mathbf{W}\mathbf{Z}^T)_{dn}]^2$$
$$+ \frac{\lambda_w}{2} \sum_{d=1}^{D} \mathbf{w}_d^T \mathbf{w}_d + \frac{\lambda_z}{2} \sum_{n=1}^{N} \mathbf{z}_n^T \mathbf{z}_n$$

- We can use coordinate descent algorithm, by first minimizing w.r.t. $\mathbf{Z}$ given $\mathbf{W}$ and then minimizing $\mathbf{W}$ given $\mathbf{Z}$. This is called **Alternating least-squares (ALS)**:

$$\mathbf{Z}^T \leftarrow (\mathbf{W}^T\mathbf{W} + \lambda_z \mathbf{I}_K)^{-1}\mathbf{W}^T\mathbf{X}$$
$$\mathbf{W}^T \leftarrow (\mathbf{Z}^T\mathbf{Z} + \lambda_w \mathbf{I}_K)^{-1}\mathbf{Z}^T\mathbf{X}^T$$

- *Complexity*: $O(DNK^2 + NK^3) \rightarrow O(DNK^2)$
- Probabilistic model

$$\prod_{(d,n)\in\Omega} \mathcal{N}(x_{dn}|\mathbf{w}_d^T\mathbf{z}_n, I) \times \prod_{n=1}^{N} \mathcal{N}(\mathbf{z}_n|0, \frac{1}{\lambda_z} I)$$
$$\times \prod_{d=1}^{D} \mathcal{N}(\mathbf{w}_d|0, \frac{1}{\lambda_w} I)$$

- Since many ratings are missing we cannot normalize the data. A solution is to add offset terms:

$$\frac{1}{2} \sum_{(d,n)\in\Omega} (x_{dn} - \mathbf{w}_d^T\mathbf{z}_n - w_{0d} - z_{0n} - \mu)^2$$

19 Singular Value Decomposition

- Matrix factorization method

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

  - $\mathbf{U}$ is a unitary $D \times D$ matrix
  - $\mathbf{V}$ is a unitary $N \times N$ matrix
  - $\mathbf{S}$ is a non-negative diagonal matrix of size $D \times N$ which are called **singular values** appearing in a descending order.
  - Columns of $\mathbf{U}$ and $\mathbf{V}$ are the left and right **singular vectors** respectively.
- Assuming $D < N$ we have

$$\mathbf{X} = \sum_{d=1}^{D} s_d \mathbf{u}_d \mathbf{v}_d^T$$

  This tells you about the spectrum of $\mathbf{X}$ where higher singular vectors contain the *low-frequency information* and lower singular values contain the *high-frequency information.*
- Dimensionality Reduction
  Take the matrix $\mathbf{S}^{(K)}$ with the K first diagonal elements non zero. Then, rank-K approx:

$$\mathbf{X} \approx \mathbf{X}_K = \mathbf{U}\mathbf{S}^{(K)}\mathbf{V}^T$$

19.1 Principal Componement Analysis

- PCA is a dimensionality reduction method and a method to decorrelate the data $\mathbf{X} \approx \tilde{\mathbf{X}} = \mathbf{W}\mathbf{Z}^T$ such that columns of $\mathbf{W}$ are orthogonal.
- If the data is zero mean

$$\mathbf{\Sigma} = \frac{1}{N}\mathbf{X}\mathbf{X}^T \Rightarrow \mathbf{X}\mathbf{X}^T = \mathbf{U}\mathbf{S}^2\mathbf{U}^T$$
$$\Rightarrow \mathbf{U}^T\mathbf{X}\mathbf{X}^T\mathbf{U} = \mathbf{U}^T\mathbf{U}\mathbf{S}^2\mathbf{U}^T\mathbf{U} = \mathbf{S}^2$$

- Thus the columns of matrix $\mathbf{U}$ are called the **principal components** and they decorrelate the covariance matrix.
- Using SVD, we can compute the matrices in the following way

$$\mathbf{W} = \mathbf{U}\mathbf{S}_D^{1/2}, \mathbf{Z}^T = \mathbf{S}^{1/2}\mathbf{V}^T$$

20 Neural Net

- Coucou

21 Bayes Net

- Graph example: $p(x, y, z) = p(y|x)p(z|x)p(x) : (y \leftarrow x \rightarrow z)$
- **D-Separation** X and Y are D-separated by Z if every path from $x \in X$ to $y \in Y$ is blocked by Z.
- **Blocked Path** if the path contains a variable that
  - is in Z and is **head-to-tail** or **tail-to-tail**
  - the node is **head-to-head** and neither the node nor the descendant are in Z.
- **Markov Blanket** (which blocks node A from the rest of the net) contains:
  - parents of A
  - children of A
  - parents of children of A