Optimization methods

Grid Search

Complexity: $\mathcal{O}(M^D ND)$, where M is the number of values tried in each dimension. D is the computation of $\tilde{\mathbf{x}}_n^T\boldsymbol{\beta}$.

Gradient Descent

- rule: $\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha\frac{\partial\mathcal{L}(\boldsymbol{\beta}^{(k)})}{\partial\boldsymbol{\beta}}$
- $\mathcal{L}(\boldsymbol{\beta}) = \frac{1}{2N}\sum_{n=1}^N(y_n - \tilde{\mathbf{x}}_n^T\boldsymbol{\beta})^2 = \frac{1}{2N}(\mathbf{y}-\tilde{\mathbf{X}}\boldsymbol{\beta})^T(\mathbf{y}-\tilde{\mathbf{X}}\boldsymbol{\beta})$
- $\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} = -\frac{1}{N}\tilde{\mathbf{X}}^T(\boldsymbol{y}-\tilde{\mathbf{X}}\boldsymbol{\beta})$

Complexity: $\mathcal{O}(IND)$
•Remember to Normalization
•Converge to a local minimu only $\alpha < \alpha_{max}$ where α_{max} is a fixed constant depends on the problem.

Least squares

- Complexity: $\mathcal{O}(ND^2 + D^3)$

Normal equton: $\tilde{\mathbf{X}}^T(\boldsymbol{y}-\tilde{\mathbf{X}}\boldsymbol{\beta}) = \mathbf{0}$
- $\boldsymbol{\beta}^* = (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\mathbf{y}$

•The Gram matrix $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$ is invertible iff $\tilde{\mathbf{X}}$ has full column rank.
•full column rank $\Rightarrow$ null space dimens is 0 $\Rightarrow \tilde{\mathbf{X}}a \neq 0$ unless a=0 $\Rightarrow a^T\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}a > 0\,\forall a\neq 0 \Rightarrow \tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$ is PD $\Rightarrow \tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$ is invertible.
- 1. $D > N$, not full column rank. 2. $\overline{\mathbf{x}}_d$ are collinear$\Rightarrow$ ill conditioned.
- $PD \Rightarrow$ 1. always has positive eigenvalue. 2. always invertible
- Gradient descent is cheaper than LS, but not always.

Ridge Regression

Linear basis function

We can create more complex models while staying in the linear framework by transforming the inputs X of dimensionality D through a function $\boldsymbol{\phi}: D \to M$; $\phi_j : R^D \to R$
- maybe Overfit or Ill-conditioned.

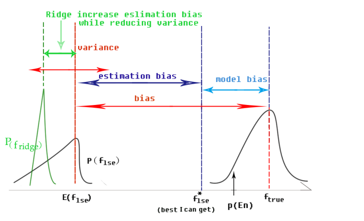
$y_n = \beta_0 + \sum_{i=1}^M \beta_i\phi_i(\mathbf{x_n}) = \tilde{\phi}^{\mathbf{T}}(\mathbf{x}_n^T)\boldsymbol{\beta}$.
- $\boldsymbol{\beta}^* = (\tilde{\boldsymbol{\Phi}}^T\tilde{\boldsymbol{\Phi}})^{-1}\tilde{\boldsymbol{\Phi}}^T\mathbf{y}$ where $\tilde{\boldsymbol{\Phi}}$ is a matrix with N rows and the n-th row is $[1, \phi_1(\mathbf{x}_n), ..., \phi_M(\mathbf{x}_n)]$.
- $\tilde{\boldsymbol{\Phi}}^T\tilde{\boldsymbol{\Phi}}$ should be invertible ($\tilde{\boldsymbol{\Phi}}$ full column-rank).

Ridge regression

- $\mathcal{L}(\boldsymbol{\beta}) = \frac{1}{2N}\sum_{n=1}^N(y_n - \tilde{\mathbf{x}}_n^T\boldsymbol{\beta})^2 + \frac{\lambda}{2N}\sum_{j=1}^M\beta_j^2$
- $\boldsymbol{\beta}_{ridge} = (\tilde{\boldsymbol{\Phi}}^T\tilde{\boldsymbol{\Phi}} + \lambda\boldsymbol{I}_M)^{-1}\tilde{\boldsymbol{\Phi}}^T\boldsymbol{y}$

•The eigenvalues of $(\tilde{\boldsymbol{\Phi}}^T\tilde{\boldsymbol{\Phi}} + \lambda\boldsymbol{I}_M)$ is at least λ – lift eigenvalue
$\tilde{\boldsymbol{\Phi}}^T\tilde{\boldsymbol{\Phi}} = QSQ^T$, $QQ^T = I_M$
$(QSQ^T + \lambda I_M) = (QSQ^T + \lambda QQ^T) = Q(S+\lambda I_M)Q^T$

Bias-Vari Decompo



Bias-variance comes directly out of the test error:
•Expected Test Error: $\overline{teErr}$
$= E[(\text{observation} - \text{prediction})^2]$
$= E_{\mathcal{D}_{tr},\mathcal{D}_{te}}[(y_* - f_{lse})^2]$
$= E_{y_*,\boldsymbol{\beta}_{lse}}[(y_* - f_{lse})^2] =$
$E_{y_*,\boldsymbol{\beta}_{lse}}[(y_* - f_{true} + f_{true} - f_{lse})^2]$
$= \underbrace{E_{y_*}[(y_* - f_{true})^2]}_{\text{var of measurement}}$
$+E_{\boldsymbol{\beta}_{lse}}[(f_{lse} - f_{true})^2]$
$= \sigma^2 + E_{\boldsymbol{\beta}_{lse}}[(f_{lse} - E_{\boldsymbol{\beta}_{lse}}[f_{lse}]$
$-f_{true} + E_{\boldsymbol{\beta}_{lse}}[f_{lse}])^2]$
$= \sigma^2 + \underbrace{E_{\boldsymbol{\beta}_{lse}}[(f_{lse} - E_{\boldsymbol{\beta}_{lse}}[f_{lse}])^2]}_{\text{pred variance}}$
$+ \underbrace{[f_{true} - E_{\boldsymbol{\beta}_{lse}}(f_{lse})]^2}_{\text{pred bias}^2}$
$\bullet reason : E[y_* - f_{true}] = 0$
- For least-squares:

$\overline{teErr} = \sigma^2 + \frac{D}{N}\sigma^2 + E_{\boldsymbol{\beta}_{lse}}[f_{true} - E_{\boldsymbol{\beta}_{lse}}(f_{lse})]^2] \Rightarrow D$ increases, variance of estimator increases; N increases, estimation bias decreases.

	bias	varian
ridge	esti bias+	-
simpler model	+	-
more data	-	

Classification

Logistic Function: $\sigma(t) = \frac{exp(t)}{1+exp(t)}$
Derivative: $\frac{\partial\sigma(t)}{\partial t} = \sigma(t)[1-\sigma(t)]$
Classification with linear regression: Use $y = 0$ as class $\mathcal{C}_1$ and $y = 1$ as class $\mathcal{C}_2$ and then decide a newly estimated y belongs to $\mathcal{C}_1$ if $y < 0.5$.

Logistic Regression

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

$-\mathcal{L}(\boldsymbol{\beta}) = \mathcal{L}_{mle}(\boldsymbol{\beta}) = \sum_{n=1}^N[y_n\tilde{\mathbf{x}}_n^T\boldsymbol{\beta} - log[1+exp(\tilde{\mathbf{x}}_n^T\boldsymbol{\beta})]]$
- $\mathbf{g} = \frac{\partial\mathcal{L}_{mle}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = -\tilde{\mathbf{X}}^T[\sigma(\tilde{\mathbf{X}}\boldsymbol{\beta}) - \mathbf{y}]$

There's no closed form, we can use gradient descent.
- $\mathbf{H}(\boldsymbol{\beta}) := -\frac{\partial\mathbf{g}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}^T} = \frac{\partial^2\mathcal{L}}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T} = \sum_{n=1}^N\frac{\partial\sigma(\tilde{\mathbf{x}}_n^T\boldsymbol{\beta})\tilde{\mathbf{x}}_n}{\partial\boldsymbol{\beta}^T} = \sum_{n=1}^N\tilde{\mathbf{x}}_n\sigma(\tilde{\mathbf{x}}_n^T\boldsymbol{\beta})[1-\sigma(\tilde{\mathbf{x}}_n^T\boldsymbol{\beta})]\tilde{\mathbf{x}}_n^T = \tilde{\mathbf{X}}^T\mathbf{S}\tilde{\mathbf{X}} \Rightarrow$ H is **PSD**. and with $S_{nn} = \sigma(\tilde{\mathbf{x}}_n^T\boldsymbol{\beta})[1-\sigma(\tilde{\mathbf{x}}_n^T\boldsymbol{\beta})]$ and so $\mathbf{S} = \sigma(\tilde{\mathbf{X}}\boldsymbol{\beta})[1-\sigma(\tilde{\mathbf{X}}\boldsymbol{\beta})]^T$.

Newton's method

It uses second order derivatives information and takes steps in the direction that minimizes a quadratic approximation to converge faster.
- $\mathcal{L}(\boldsymbol{\beta}) \cong \mathcal{L}(\boldsymbol{\beta}^{(k)}) + g_k^T(\boldsymbol{\beta}-\boldsymbol{\beta}^{(k)}) + \frac{1}{2}(\boldsymbol{\beta}-\boldsymbol{\beta}^{(k)})^T\mathbf{H}_k(\boldsymbol{\beta}-\boldsymbol{\beta}^{(k)}) \Rightarrow \frac{\partial\mathcal{L}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = 0 + g_k + \mathbf{H}_k(\boldsymbol{\beta}-\boldsymbol{\beta}^{(k)}) = 0$

$\Rightarrow \boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha_k\mathbf{H}_\mathbf{k}^{-1}\frac{\partial\mathcal{L}(\boldsymbol{\beta}^{(k)})}{\partial\boldsymbol{\beta}}$
•Newton's method is equivalent to solving many least-squares problems.
Complexity: $\mathcal{O}(IND^2 + ID^3)$

Penalized Logistic Regr

The cost-function can be unbounded when the data is linearly separable.
$min_\beta - \sum_{n=1}^N logp(y_n|\mathbf{x}_n^T,\boldsymbol{\beta}) + \lambda\sum_{d=1}^D\beta_d^2 \Rightarrow$strict convex($H = \tilde{\mathbf{X}}^\mathbf{T}\mathbf{S}\tilde{\mathbf{X}} + \lambda\mathbf{I}$) is positive-definite

Iter Recur Leas-Squar

express Newton's method with
$\alpha_k = 1$
$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha_k\mathbf{H}_k^{-1}\mathbf{g}_k$
$= \boldsymbol{\beta}^{(k)} - (\tilde{\mathbf{X}}^T\mathbf{S}_k\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T(\boldsymbol{\sigma}_k - \mathbf{y})$
$= (\tilde{\mathbf{X}}^T\mathbf{S}_k\tilde{\mathbf{X}})^{-1}[(\tilde{\mathbf{X}}^T\mathbf{S}_k\tilde{\mathbf{X}})\boldsymbol{\beta}^{(k)} - \tilde{\mathbf{X}}^T(\boldsymbol{\sigma}_k - \mathbf{y})]$
$= (\tilde{\mathbf{X}}^T\mathbf{S}_k\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\mathbf{S}_k[\tilde{\mathbf{X}}\boldsymbol{\beta}^{(k)} + \mathbf{S}_k^{-1}(\mathbf{y} - \boldsymbol{\sigma}_k)]$
$= (\tilde{\mathbf{X}}^T\mathbf{S}_k\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\mathbf{S}_k\mathbf{z}_k$
$\Longrightarrow \boldsymbol{\beta}^{(k+1)} = argmin_{\boldsymbol{\beta}}\sum_{n=1}^N S_{nn}^k(\mathbf{z}_n^k - \tilde{\mathbf{x}}_n^k\boldsymbol{\beta})^2$
i.e.$\mathbf{L}(\boldsymbol{\beta}) = \mathbf{S}_k(\mathbf{z}_k - \tilde{\mathbf{X}}\boldsymbol{\beta})^T(\mathbf{z}_k - \tilde{\mathbf{X}}\boldsymbol{\beta})$

Generalized linear model

•partition func, natural param, log-partition func
$p(y|\boldsymbol{\eta}) := \frac{h(y)}{Z(\boldsymbol{\eta})}exp[\boldsymbol{\eta}^T\boldsymbol{\Phi}(y) - A(\boldsymbol{\eta})]$
•Logistic Regression:
$p(y_n|\eta_n) := \frac{exp(y_n\eta_n)}{1+exp(\eta_n)} = exp[y_n\eta_n - log(1+exp(\eta_n))]$
The GLM consists of three elements:

- A probability distribution from the exponential family.
- A linear predictor $\hat{y} = \mathbf{X}\boldsymbol{\beta}$.
- A link function g: $E(y) = \mu = g^{-1}(\eta)$.

•Bernoulli distribution:
$p(y|\mu) := \mu^y(1-\mu)^{(1-y)} = exp[y\,log\frac{\mu}{1-\mu} + log(1-\mu)]$
$\frac{\partial A(\eta)}{\partial\eta} = E[\Phi(y)]$, $\frac{\partial^2A(\eta)}{\partial\eta^2} = Var(\Phi(y)) \Rightarrow$A is convex
•Gaussian distribution:
$\boldsymbol{\Phi}(y) = [y, y^2]^T$, $\boldsymbol{\eta}^T = [\frac{\mu}{\sigma^2}, \frac{-1}{2\sigma^2}]$, $h(y) = (2\pi)^{-1/2}$, $Z(\boldsymbol{\eta}) = (-2\eta_2)^{1/2}$, $A(\boldsymbol{\eta}) = \frac{-\eta_1^2}{4\eta_2}$
•Multinomial distribution:
•Maximum likelihood estimate:
$min\mathcal{L}(\boldsymbol{\beta}) = -\sum_{n=1}^N log\,p(y_n|\tilde{\mathbf{x}}_n^T\boldsymbol{\beta})$
1. $\mathcal{L}_n(\boldsymbol{\beta}) = -logp(y_n|\tilde{\mathbf{x}}_n^T\boldsymbol{\beta}) = -[\eta_n\Phi(y_n) - A(\eta_n)] + cnst$
2. $\frac{\partial\mathcal{L}_n}{\partial\boldsymbol{\beta}} = \underline{\frac{\partial\eta_n}{\partial\boldsymbol{\beta}}\frac{\partial\mathcal{L}_n}{\partial\eta_n}} = \tilde{\mathbf{x}}_n[g^{-1}(\eta_n) - \Phi(y_n)]$
3.vector: $\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} = \mathbf{X}^T[g^{-1}(\boldsymbol{\eta}) - \Phi(\mathbf{y})]$
4. $\frac{\partial^2\mathcal{L}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T} = \mathbf{X^TSX}$, with $S_{nn} = \frac{\partial^2A(\eta_n)}{\partial\eta_n^2}$ always PSD

Kernel Ridge Reg

- $(\mathbf{P_{NxM}Q_{MxN}} + \mathbf{I}_N)^{-1}\mathbf{P} = \mathbf{P}(\mathbf{QP} + \mathbf{I_M})^{-1}$
- $\boldsymbol{\beta}^* = (\mathbf{X^TX} + \lambda\mathbf{I}_D)^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{X}^T(\mathbf{XX}^T + \lambda\mathbf{I}_N)^{-1}\mathbf{y} = \mathbf{X}^T\boldsymbol{\alpha}^*$

complexity: $O(D^2N + D^3)$
$\Rightarrow O(N^2D + N^3)$
•**Representer Theorem**: For any $\boldsymbol{\beta}^*$ minimizing $\min_\beta\sum_{n=1}^N\mathcal{L}(y_n, \mathbf{x}_n^T\boldsymbol{\beta}) + \sum_{d=1}^D\lambda\beta_d^2$ there exists an $\boldsymbol{\alpha}^*$ such that $\boldsymbol{\beta}^* = \mathbf{X}^T\boldsymbol{\alpha}^*$.
•Instead of working in the **column space** of our data, we can work in the **row space**:
$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta} = \mathbf{XX^T}\boldsymbol{\alpha} = \mathbf{K}\boldsymbol{\alpha}$
•♠ $\boldsymbol{\beta}^* = \mathbf{X}^T\boldsymbol{\alpha}^*$ implies for ridge reg that $\mathbf{x}_*^T\boldsymbol{\beta}^* = \sum_{n=1}^N\alpha_n(\mathbf{x}_*^T\mathbf{x}_n)$
- $\boldsymbol{\beta}^* = \sum_{n=1}^N\alpha_n^*\mathbf{x}_n \Rightarrow \boldsymbol{\beta}^*$ lies in row space of $\mathbf{X}$
- $\widehat{\mathbf{y}} = \sum_{d=1}^D\beta_d^*\overline{\mathbf{x}}_d \Rightarrow \widehat{\mathbf{y}}$ lies in column space of $\mathbf{X}$
- $\boldsymbol{\beta}^* = arg\,min_{\boldsymbol{\beta}}\frac{1}{2}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) + \frac{\lambda}{2}\boldsymbol{\beta}^T\boldsymbol{\beta}$
- $\boldsymbol{\alpha}^* = arg\,max_{\boldsymbol{\alpha}} - \frac{1}{2}\boldsymbol{\alpha}^T(\mathbf{XX}^T + \lambda\mathbf{I}_N)\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{y}$
- Properties of a Kernel:

1. $\mathbf{K}$ should be symmetric: $\mathbf{K}^T = \mathbf{K}$
Note: $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$
2. $\mathbf{K}$ should be PSD: $\forall$ nonzero vector $\mathbf{a}$, $\mathbf{a}^T\mathbf{K}\mathbf{a} \geq 0$.
- $t^t\mathbf{K}t = \sum_i\sum_j K_{ij}t_it_j$

$(\mathbf{K})_{i,j} = \kappa(\mathbf{x_i}, \mathbf{x_j}) = \vec{\phi}(\mathbf{x_i})^T\vec{\phi}(\mathbf{x_j})$.
- The ϕ **(basis function)** are not that important in the end, because we only use the Kernel as is.

$\kappa(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i}^T\mathbf{x_j}$
$\kappa(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i}^T\mathbf{x_j} + c)^d$
$\kappa(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\frac{\lvert\lvert\mathbf{x_i}-\mathbf{x_j}\rvert\rvert^2}{2\sigma^2}\right)$

SVM

•Cost func(hinge loss): $g(\boldsymbol{\beta}) = \sum_{n=1}^N C[1 - y_n\tilde{\phi}_n^T\boldsymbol{\beta}]_+ + \frac{1}{2}\sum_{j=1}^M\beta_j^2$ with $[t]_+ = \max(0, t)$, $C = \frac{1}{\lambda}$
•Convex, not Differentiable.
- $\mathcal{G}(\boldsymbol{\beta}, \boldsymbol{\alpha}) = \sum_{n=1}^N\alpha_n(1 - y_n\tilde{\phi}_n^T\boldsymbol{\beta}) + \frac{1}{2}\sum_{j=1}^M\beta_j^2$
- $\mathcal{G}(\boldsymbol{\beta}, \boldsymbol{\alpha})$ convex in $\boldsymbol{\beta}$, concave in $\boldsymbol{\alpha}$

$min_{\boldsymbol{\beta}}g(\boldsymbol{\beta}) = min_{\boldsymbol{\beta}}max_{\boldsymbol{\alpha}}\mathcal{G}(\boldsymbol{\beta}, \boldsymbol{\alpha}) = max_{\boldsymbol{\alpha}}min_{\boldsymbol{\beta}}\mathcal{G}(\boldsymbol{\beta}, \boldsymbol{\alpha}) = max_\alpha g^*(\boldsymbol{\alpha})$
- $\max_{\alpha\in[0;C]^N}\boldsymbol{\alpha}^T\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Y\Phi\Phi^TY}\boldsymbol{\alpha}$, $\boldsymbol{\alpha}^T\mathbf{y} = 0, \mathbf{Y} := diag(\mathbf{y})$, $\boldsymbol{\Phi}$ has no first 1 column.
- adv: Differentiable, kernelized,

K-means Clustering

- $min_{\mathbf{r},\boldsymbol{\mu}}\mathcal{L}(\mathbf{r}, \boldsymbol{\mu}) = \sum_{k=1}^K\sum_{n=1}^N r_{nk}\lvert\lvert\mathbf{x}_n - \boldsymbol{\mu}_k\rvert\rvert_2^2$
- CONVEX with $\boldsymbol{\mu}$, no notion convex with $\mathbf{r}$ and $\boldsymbol{\mu}$

1. For all n, compute $\mathbf{r}_n$ given $\boldsymbol{\mu}$
$r_{nk} = 1$ if k$=$
$arg\,min_{j=1,2,...,K}\lvert\lvert\mathbf{x}_n - \boldsymbol{\mu}_j\rvert\rvert_2^2$
2. For all k, compute $\boldsymbol{\mu}_k$ given $\mathbf{r}$
$\mathcal{L}(\boldsymbol{\mu}_k) = \sum_{n=1}^N r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k)^T(\mathbf{x}_n - \boldsymbol{\mu}_k) + cnst \Rightarrow$
$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\mu}_k} = -2\sum_{n=1}^N r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k) = 0$
$\Rightarrow \boldsymbol{\mu}_k = \frac{\sum_{n=1}^N r_{nk}\mathbf{x}_n}{\sum_{n=1}^N r_{nk}}$
•If use L_1 distance, called K-median
•Complexity: $\mathcal{O}(NKDI)$
•Probabilistic model:
$-log\,p(\mathbf{r}, \mu) = \frac{1}{2}\mathcal{L}(\mathbf{r}, \mu) + cnst$
$\Rightarrow p(\mathbf{r}, \boldsymbol{\mu}) \propto exp^{-\frac{1}{2}\sum_k\sum_n r_{nk}\lvert\lvert\mathbf{x}_n - \boldsymbol{\mu}_k\rvert\rvert^2}$
$\prod_n\prod_k[exp^{-\frac{1}{2}\lvert\lvert\mathbf{x}_n - \boldsymbol{\mu}_k\rvert\rvert^2}]^{r_{nk}} \Rightarrow$
$p(\mathbf{r}, \boldsymbol{\mu}) = \prod_n\prod_k[\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \mathcal{I})]^{r_{nk}}$

Gaussian Mixtu Mode

GMM is a latent variable model, $\mathbf{r}_n$ is latent variables
- $p(\mathbf{X}, \mathbf{r}|\mu, \boldsymbol{\Sigma}, \pi) = \prod_{n=1}^N[p(\mathbf{x}_n|\mathbf{r}_n, \mu, \boldsymbol{\Sigma})p(r_n|\pi)]$

$= \prod_{n=1}^N[\prod_{k=1}^K[\mathcal{N}(\mathbf{x}_n|\mu_k, \boldsymbol{\Sigma}_k)]^{r_{nk}}\prod_{k=1}^K[\pi_k]^{r_{nk}}]$
- $p(\mathbf{x}_n|\theta) = \sum_{k=1}^K p(\mathbf{x}_n, \mathbf{r}_n = k|\theta) = \sum_{k=1}^K p(\mathbf{x}_n|\mathbf{r}_n = k, \theta)p(\mathbf{r}_n = k|\theta) = \sum_{k=1}^K\pi_kp_k(\mathbf{x}_n|\mathbf{r}_n = k, \theta) = \sum_{k=1}^K\pi_k\mathcal{N}(\mathbf{x}_n|\underbrace{\mu_k, \boldsymbol{\Sigma}_k}_{\theta})$
- $max_\theta\mathcal{L}(\theta) = max_\theta logp(\mathbf{X}|\theta) = \sum_{n=1}^N logp(\mathbf{x}_n|\theta) = \sum_{n=1}^N log\sum_{k=1}^K\pi_k\mathcal{N}(\mathbf{x}_n|\mu_k, \boldsymbol{\Sigma}_k)$

the negative of it is **not Convex** wrt θ, **not Identifiable**
- To use this for clustering, we first fit this mixture and then compute the posterior $p(z_i = k|\mathbf{x}_i, \theta)$. This yields *soft* cluster assignments.

EM

$log\sum_{k=1}^K\pi_k\mathcal{N}(\mathbf{x}_n|\mu_k, \boldsymbol{\Sigma}_k) \geq \sum_{k=1}^K p_{kn}log\frac{\pi_k\mathcal{N}(\mathbf{x}_n|\mu_k, \boldsymbol{\Sigma}_k)}{p_{kn}}$
with equality when,
$p_{kn} = \frac{\pi_k\mathcal{N}(\mathbf{x}_n|\mu_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K\pi_j\mathcal{N}(\mathbf{x}_n|\mu_j, \boldsymbol{\Sigma}_j)}$
- 1. E-step: Compute responsibilities $p_{kn}^{(i)}$

$p_{kn}^{(i)} = \frac{\pi_k^{(i)}\mathcal{N}(\mathbf{x}_n|\mu_k^{(i)}, \boldsymbol{\Sigma}_k^i)}{\sum_{k=1}^K\pi_k\mathcal{N}(\mathbf{x}_n|\mu_k^{(i)}, \boldsymbol{\Sigma}_k^{(i)})}$
- 2. Compute the marginal likelihood (Cost)

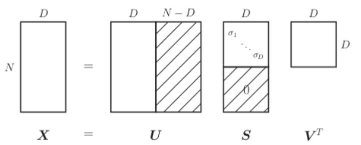
$\mathcal{L}(\boldsymbol{\theta}^{(i)}) = \sum_{n=1}^N log\sum_{k=1}^K\pi_k^{(i)}\mathcal{N}(\mathbf{x}_n|\mu_k^{(i)}, \boldsymbol{\Sigma}_k^{(i)})$
- 3. M-step: Update $\mu_k^{(i+1)}$, $\boldsymbol{\Sigma}_k^{(i+1)}$, $\pi_k^{(i+1)}$

$\mu_k^{(i+1)} = \frac{\sum_{n=1}^N p_{kn}^{(i)}\mathbf{x}_n}{\sum_{n=1}^N p_{kn}^{(i)}}$
$\boldsymbol{\Sigma}_k^{(i+1)} = \frac{\sum_{n=1}^N p_{kn}^{(i)}(\mathbf{x}_n - \mu_k^{(i+1)})(\mathbf{x}_n - \mu_k^{(i+1)})^T}{\sum_{n=1}^N p_{kn}^{(i)}}$
$\pi_k^{(i+1)} = \frac{1}{N}\sum_{n=1}^N p_{kn}^{(i)}$
EM in general: $logp(\mathbf{x}_n|\boldsymbol{\theta}) = log\sum_r[\frac{p(\mathbf{x}_n, \mathbf{r}|\boldsymbol{\theta})}{p(\mathbf{r}|\boldsymbol{\theta}^{(i)}, \mathbf{x}_n)} * p(\mathbf{r}|\boldsymbol{\theta}^{(i)}, \mathbf{x}_n)]$
$\geq \sum_r[logp(\mathbf{x}_n, \mathbf{r}|\boldsymbol{\theta})]p(\mathbf{r}|\boldsymbol{\theta}^{(i)}, \mathbf{x}_n) + H[p(\mathbf{r}|\boldsymbol{\theta}^{(i)}, \mathbf{x}_n)]$
$\Rightarrow \boldsymbol{\theta}^{(i+1)} = argmax_\theta\sum_{n=1}^N E_{p(\mathbf{r}_n|\mathbf{x}_n, \boldsymbol{\theta}^{(i)})}[logp(\mathbf{x}_n, \mathbf{r}_n|\boldsymbol{\theta})]$

SVD

- $\mathbf{X_{DxN}} = \mathbf{USV}^T = \sum_{j=1}^D s_j\mathbf{u}_j\mathbf{v}_j^T$. All rank 1.
- $\mathbf{v}_j^T\mathbf{v}_j = 1$, $\mathbf{v}_j^T\mathbf{v}_i = 0$, $\mathbf{U}^T\mathbf{U} = I$

$\mathbf{V}^T\mathbf{V} = I$
★ $\mathbf{X}\mathbf{v}_j = s_j\mathbf{u}_j$
•The singular vals of a $D\times N$ matrix $\mathbf{X}$ are the square roots of the eigenvalues of the $D\times D$ matrix $\mathbf{XX^T}$ $(D < N)$
•The singular vals of a $N\times D$ matrix $\mathbf{X}$ are the square roots of the eigenvalues of the $D\times D$ matrix $\mathbf{X^TX}$ $(D < N)$
The same as with PCA, we can do with SVD:



PCA

• $S := \frac{1}{N}\sum_n (x_n - \overline{x})(x_n - \overline{x})^T$
•Find the eigenvectors of the covariance matrix $\mathbf{X_{DxN}X^T}$ of the data. These form an orthonormal basis $\{\mathbf{w}_1, ..., \mathbf{w}_D\}$ for the data in the directions that have highest variance. One can then use the first $M < D$ vectors to rebuild the data: $\hat{\mathbf{x}}_n = \mathbf{W}\mathbf{z}_n = \mathbf{W}\mathbf{W}^T\mathbf{x}_n$, with $\mathbf{W} = \left[\mathbf{w}_1; ...; \mathbf{w}_M\right]$. This minimizes MSE $\frac{1}{N}\sum_{n=1}^{N}||\mathbf{x}_n - \hat{\mathbf{x}}_n||^2$.
• $\mathbf{x}_{n,rot} = \mathbf{W}^T\mathbf{x}_n$,
$\mathbf{W}^T\mathbf{W} = I$, $\mathbf{W}\mathbf{W}^T = I$
$\Rightarrow \mathbf{x}_n = \mathbf{W}\mathbf{x}_{n,rot}$
•When use, we use $\tilde{\mathbf{x}} = \mathbf{W}^T\mathbf{x}_i$, which is the lower-dimension approximation to speed up.
•Computation Cost:$O(D^3 + ND^2 + DN^2)$??

PCA, SVD relation

• $\mathbf{X}$ is centered(zero mean).
• $\mathbf{X}\mathbf{X}^T = \mathbf{W}\mathbf{L}\mathbf{W}^{-1} = \mathbf{U}\mathbf{S^2}\mathbf{U}^T$
• $\hat{\mathbf{X}} = U(:, 1:M) * S(1:M, 1:M) * V(:, 1:M)^T$

Bayesian methods

•The **prior** $p(\mathbf{f}|\mathbf{X})$ encodes our prior belief about the "true" model $\mathbf{f}$. The **likelihood** $p(\mathbf{y}|\mathbf{f})$ measures the probability of our (possibly noisy) observations given the prior.
•Least-squares tries to find model parameters $\boldsymbol{\beta}$ which maximize the likelihood. Ridge regression maximizes the **posterior** $p(\boldsymbol{\beta}|\mathbf{y})$

Bayesian networks

This example can be factorized as follows: $p(y_1, y_2, z_1, z_2, z_3) = p(y_1|z_1, z_2)p(y_2|z_2, z_3)p(z_1)p(z_2)p(z_3)$

$$\begin{tikzcd}
z_1 \arrow[d] & z_2 \arrow[dl] \arrow[dr] & z_3 \arrow[d] \\
y_1 & & y_2
\end{tikzcd}$$

We can then obtain the distribution over latent factors (z_i) by marginalizing over the unknown variables:
$p(z_1, z_2, z_3|y_1, y_2) = \frac{\text{joint}}{p(y_1, y_2)}$
$\implies p(z_1|y_1, y_2) = \sum_{z_2, z_3}\frac{\text{joint}}{p(y_1, y_2)}$
•Complexity:$\mathcal{O}(2^M E)$ with M is the recursive variables, E is the edges.
•Sum-Product Algo: $p(z_1|y_a, y_b) \propto \sum_{z_2}\sum_{z_3} p(y_a|z_1, z_2)p(y_b|z_2, z_3) p(z_1)p(z_2)p(z_3) = p(z_1)\{\sum_{z_2} p(y_a|z_1, z_2)p(z_2) [\sum_{z_3} p(y_b|z_2, z_3)p(z_3)]\}$

Belief propagation

•z: variables; y: observations
•From variables to observations:
$m_{i\to a}(z_i) = p(z_i)\prod_{b\in\mathcal{N}(i)\backslash a} m_{b\to i}(z_i)$
•From observations to variables:
$m_{a\to i}(z_i) = \sum_{j\neq i}\{p(y_a|Paret_a)\prod_{j\in\mathcal{N}(a)\backslash i} m_{j\to a}(z_j)\}$
•1. Initialization: $m_{i\to a} = p(z_i)\,\forall a$
2. Compute marginals by multiplying all the message received at node i.
$p(z_i|\mathbf{y}) = p(z_j)\prod_{j\in\mathcal{N}(a)} m_{j\to a}(z_j)$
The Right One:
$p(z_i|\mathbf{y}) = p(z_i)\prod_{a\in\mathcal{N}(i)} m_{a\to i}(z_i)$
Belief propagation is a message-passing based algorithm used to compute desired marginals (e.g. $p(z_1|y_1, y_2)$) efficiently. It leverages the factorized expression of the joint.

Neural Networks

A feed forward Neural Network is organized in K layers, each layer with $M^{(k)}$ hidden units $z_i^{(k)}$.
Activations $a_i^{(k)}$ are computed as the linear combination of the previous layer's terms, with weights $\boldsymbol{\beta}^{(k)}$ (one $M^{(k-1)} \times 1$ vector of weights for each of the $M^{(k)}$ activations).
Activations are then passed through a (possibly nonlinear) function h to compute the hidden unit $z_{in}^{(k)}$.

$\mathbf{x}_n \xrightarrow{\boldsymbol{\beta}_i^{(1)}} a_{in}^{(1)} \xrightarrow{h} z_{in}^{(1)} \xrightarrow{\boldsymbol{\beta}^{(2)}}$
$\ldots \mathbf{z}_n^{(K-1)} \xrightarrow{link\,func} \mathbf{y}_n$
• $a_{in}^{(k)} = (\boldsymbol{\beta}_i^{(k)})^T\mathbf{z}_n^{(k-1)}$,
$z_{in}^{(k)} = h(a_{in}^{(k)})$
• first layer: $\mathbf{z}_n^{(0)} = \mathbf{x}_n$
• $\mathbf{a}_n^{(k)} = \mathbf{B}^{(k)}\mathbf{z}_n^{(k-1)}, \mathbf{z}_n^{(k)} = h(\mathbf{a}_n^{(k)})$

Backpropagation

It's an algorithm which computes the gradient of the cost $\mathcal{L}$ w.r.t. the parameters $\boldsymbol{\beta}^{(k)}$.
Forward pass: compute a_i, z_i and $\mathbf{y}_n$ from $\mathbf{x}_n$.
Backward pass: work out derivatives from outputs to the target $\boldsymbol{\beta}_i^{(k)}$.
Using the chain rule:
• $\boldsymbol{\delta}_n^{(k-1)} = \frac{\partial\mathcal{L}}{\partial\mathbf{a}_n^{(k-1)}} = diag[\mathbf{h}'(\mathbf{a}_n^{(k-1)})](\mathbf{B^{(k)}})^T\boldsymbol{\delta}_n^{(k)}$
• $\frac{\partial\mathcal{L}}{\partial\mathbf{B}^{(1)}} = \boldsymbol{\delta}_n^{(1)}\mathbf{x}_n^T$
• $\frac{\partial\mathcal{L}}{\partial\mathbf{B}^{(k)}} = \boldsymbol{\delta}_n^{(k)}(\mathbf{z}_n^{(k)})^T$

Regularization

NN are not *identifiable* (existence of many local optima), therefore the maximum likelihood estimator is not *consistent*.
NN are universal density estimators, and thus prone to severe overfitting. Techniques used to reduce overfitting include early stopping (stop optimizing when test error starts increasing) and "weight decay" (i.e. L_2 regularization).

General

Cost functions

•MSE(and matrix formulation):
$\frac{1}{2N}\sum_{n=1}^{N}[y_n - f(\mathbf{x}_i)]^2$
$= \frac{1}{2N}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$
•Mean absolute error (MAE):
$\frac{1}{2N}\sum_{n=1}^{N}|y_n - f(\mathbf{x}_i)|$
•Huber loss(convex, differentiable, robust to outliers):

$$\mathcal{L}_\delta(a) = \begin{cases}\frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{otherwise.}\end{cases}$$

•Turkey's loss(non-convex, non-differentiable, robust to outliers)
•RMSE: $\sqrt{2 * \text{MSE}}$
•Epsilon insensitive ("hinge loss"):
$\mathcal{L}_\epsilon(y, \hat{y}) =$

$$\begin{cases}0 & \text{if } |y - \hat{y}| \le \epsilon, \\ |y - \hat{y}| - \epsilon, & \text{otherwise.}\end{cases}$$

Classif Cost functions

RMSE: $\sqrt{\frac{1}{N}\sum_{n=1}^{N}[y_n - \hat{p_n}]^2}$
0-1 Loss: $\frac{1}{N}\sum_{n=1}^{N}\delta(y_n, \hat{y_n})$
Log-Loss: $-\frac{1}{N}\sum_{n=1}^{N} y_n\log(\hat{p_n}) + (1 - y_n)\log(1 - \hat{p_n})$

Distributions

•Multivariate gaussian distribution:
$\mathcal{N}(\mathbf{X}|\mu, \boldsymbol{\Sigma}) = p(\mathbf{X} = \mathbf{x}) = (2\pi)^{-d/2}|\boldsymbol{\Sigma}|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x} - \mu)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \mu)\right]$
•Gaussian distribution:
$\mathcal{N}(X|\mu, \sigma^2) = p(X = x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2\right)$
•Poisson distribution: $\mathcal{P}(X|\lambda)$
$\implies p(X = k) = \frac{\lambda^k}{k!}\exp(-\lambda)$
•Laplace distribution:
$p(y_n|\tilde{\mathbf{x}}_n, \boldsymbol{\beta}) = \frac{1}{2b}e^{\frac{1}{b}|y_n - \tilde{\mathbf{x}}_n^T\boldsymbol{\beta}|}$

Properties

• If $\mathbf{V}$ symmetric positive definite, then for all $\mathbf{P} \neq 0$, $\mathbf{P^T V P}$ is positive semi-definite (and even positive definite if $\mathbf{P}$ is not singular).
•**Jensen's inequality** applied to log: $\log(\mathbb{E}[X]) \ge \mathbb{E}[\log(X)]$
$\implies \log(\sum_x x\cdot p(x)) \ge \sum_x p(x)\log(x)$
•Useful derivative:
$\frac{\partial\mathbf{x^T B x}}{\partial\mathbf{x}} = (\mathbf{B} + \mathbf{B^T})\mathbf{x}$
• **Marginal and Conditional Gaussians:**
$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^-1)$
$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^-1)$
$\Downarrow$
$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^-1 + \mathbf{A}\boldsymbol{\Lambda}^-1\mathbf{A}^T)$
$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A^T L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\mu\}, \boldsymbol{\Sigma})$
where $\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A^T L A})^{-1}$
• **derivative for expo like PDF**:
$\frac{\partial f(x)}{\partial x} = \frac{\partial e^{ln f(x)}}{\partial x} = f(x)\frac{\partial ln f(x)}{\partial x}$

Concepts

Convexity

Sum of two convex functions is convex. Composition of a convex function with a convex, nondecreasing function is convex. Linear, exponential and $\log(\sum\exp)$ functions are convex.

Identifiability

We say that a statistical model $\mathcal{P} = \{P_\theta : \theta \in \Theta\}$ is identifiable if the mapping $\theta \mapsto P_\theta$ is one-to-one: $P_{\theta_1} = P_{\theta_2} \quad\Rightarrow\quad \theta_1 = \theta_2 \quad$ for all $\theta_1, \theta_2 \in \Theta$.
A non-identifiable model will typically have many local optima yielding the same cost, e.g. $\mathcal{L}(W, Z) = \mathcal{L}(aW, \frac{1}{a}Z)$
• check permutation of labels.

Curse of dimensionality

With the dimensionality increase, every data point becomes arbitrarily far from every other data point and therefore the choice of nearest neighbor becomes random. In high dimension, data only covers a tiny fraction of the input space, making generalization extremely difficult.
• expected edge length $=$ $e_D(r) = r^{1/D}$; $e_{10}(0.1)$=0.80, to capture 10% of data, must cover 80% of input.
•The median distance from the origin to the closest data point is: $(1 - \frac{1}{2}^{1/N})^{1/D}$. choice of KNN becomes random.

Maximum Likelihood

• $\boldsymbol{\beta}_{lse} = arg\,min_\beta\frac{1}{2}(\mathbf{y} - \tilde{\boldsymbol{\Phi}}\boldsymbol{\beta})^T(\mathbf{y} - \tilde{\boldsymbol{\Phi}}\boldsymbol{\beta})$
$= arg\,max_\beta[log\mathcal{N}(\mathbf{y}|\tilde{\boldsymbol{\Phi}}\boldsymbol{\beta}, \mathbf{I})]$
• $\boldsymbol{\beta}_{ridge} = arg\,min_\beta\frac{1}{2}(\mathbf{y} - \tilde{\boldsymbol{\Phi}}\boldsymbol{\beta})^T(\mathbf{y} - \tilde{\boldsymbol{\Phi}}\boldsymbol{\beta}) + \frac{1}{2}\lambda\boldsymbol{\beta}^T\boldsymbol{\beta} = arg\,max_\beta[log\mathcal{N}(\mathbf{y}|\tilde{\boldsymbol{\Phi}}\boldsymbol{\beta}, \mathbf{I})\mathcal{N}(\boldsymbol{\beta}|\mathbf{0}, \frac{1}{\lambda}\mathbf{I})]$
•multinomial: $p(\mathbf{x}_n|\boldsymbol{\mu}) = \prod_{k=1}^{K}\mu_k^{x_k} = exp[\sum_{k=1}^{K} x_k ln\,\mu_k] = exp[\sum_{k=1}^{K-1} x_k ln(\frac{\mu_k}{1-\sum_{j=1}^{K-1}}) + ln(1 - \sum_{k=1}^{K-1}\mu_k)]$, with $\sum_{k=1}^{K}\mu_k = 1$, $E[\mathbf{x}] = \boldsymbol{\mu}$, $\sum_{k=1}^{K} x_k = 1, x_k = 1$.

Matrix Factorization

• $\mathbf{X}_{D*N} = \mathbf{W}_{D*M}\mathbf{Z}_{N*M}^T$, with M is the num of factors.
• $\mathcal{L}(\mathbf{W}, \mathbf{Z}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{d=1}^{D}(x_{dn} - \mathbf{w}_d^T\mathbf{z}_n)^2$
$+\frac{\lambda_w}{2}\sum_{d=1}^{D}\mathbf{w}_d^T\mathbf{w}_d$
$+\frac{\lambda_z}{2}\sum_{n=1}^{N}\mathbf{z}_n^T\mathbf{z}_n$
• $\mathcal{L}(\mathbf{Z}|\mathbf{W}) = \frac{1}{2}\sum_{n=1}^{N}[(\mathbf{x}_n - \mathbf{W}\mathbf{z}_n)^T(\mathbf{x}_n - \mathbf{W}\mathbf{z}_n)] + \lambda_z\mathbf{z}_n^T\mathbf{z}_n$
• $\mathbf{z}_n^* = (\mathbf{W}^T\mathbf{W} + \lambda_z\mathbf{I})^{-1}\mathbf{W}^T\mathbf{x}_n$
$\mathbf{w}_d^* = (\mathbf{X}^T\mathbf{Z} + \lambda_w\mathbf{I}_M)^{-1}\mathbf{Z}^T\mathbf{x}_d$
$\mathbf{Z}^T = (\mathbf{W}^T\mathbf{W} + \lambda_z\mathbf{I}_M)^{-1}\mathbf{W}^T\mathbf{X}$
$\Rightarrow \mathcal{O}((M^2D + M^3)N)$
$\mathbf{W}^T = (\mathbf{Z}^T\mathbf{Z} + \lambda_w\mathbf{I}_M)^{-1}\mathbf{X}^T\mathbf{X}^T$
$\Rightarrow \mathcal{O}((M^2N + M^3)D)$
•
$\prod_{n=1}^{N}\prod_{d\in O_n} N(x_{dn}|\mathbf{w}_d^T\mathbf{z}_n, 1)\prod_n N(\mathbf{z}_n|\mathbf{0}, \frac{1}{\lambda_z}\mathbf{I})\prod_d N(\mathbf{w}_d|\mathbf{0}, \frac{1}{\lambda_w}\mathbf{I})$

Decision Tree

• $L = (x_i, y_i)$, $s.t.\, x_{ik} \succ \tau$, $R...$
• $p_L = \frac{\sharp y_i=1}{N_L}$, $p_R...$
• $I_{split} = N_L I(p_L) + N_R I(p_R)$
• Cross-entropy = I(p)
• gini impurity = 2p(1-p)
• regr impuri=$\frac{1}{N_{L,R}}\sum_n(y_n - \overline{y})^2$