

Introduction to Machine Learning SS20	0/1 loss	Kernelized linear regression (KLR)	Backpropagation
Fundamental Assumption	$l_{0/1}(w; y_i, x_i) = 1$ if $y_i \neq \text{sign}(w^T x_i)$ else 0	Ansatz: $w^* = \sum_{i=1}^n \alpha_i x$	Output layer:
Data is iid for unknown P : $(x_i, y_i) \sim P(X, Y)$	Perceptron algorithm	$\alpha^* = \underset{\alpha}{\text{argmin}} \alpha \ \alpha^T K - y \ _2^2 + \lambda \alpha^T K \alpha$	Error: $\delta^{(L)} = \mathbf{l}'(\mathbf{f}) = [l'(f_1), \dots, l'(f_p)]$
True risk and estimated error	Use $l_P(w; y_i, x_i) = \max(0, -y_i w^T x_i)$ and SGD	$= (K + \lambda I)^{-1} y$, Prediction: $\hat{y} = \sum_{i=1}^n \alpha_i k(x_i, \hat{x})$	Gradient: $\nabla_{\mathbf{W}^{(L)}} \ell(\mathbf{W}; \mathbf{y}, \mathbf{x}) = \delta^{(L)} \mathbf{v}^{(L-1)T}$
True risk: $R(w) = \int P(x, y) (y - w^T x)^2 \partial x \partial y = \mathbb{E}_{x,y} [(y - w^T x)^2]$	$\nabla_w l_P(w; y_i, x_i) = \begin{cases} 0 & \text{if } y_i w^T x_i \geq 0 \\ -y_i x_i & \text{otherwise} \end{cases}$	k-NN	Hidden layers:
Est. error: $\hat{R}_D(w) = \frac{1}{ D } \sum_{(x,y) \in D} (y - w^T x)^2$	Data lin. separable \Leftrightarrow obtains a lin. separator (not necessarily optimal)	$y = \text{sign} \left(\sum_{i=1}^n y_i [x_i \text{ among } k \text{ nearest neighbours of } x] \right)$ – No weights \Rightarrow no training! But depends on all data.	Error: $\delta^{(\ell)} = \phi'(\mathbf{z}^{(\ell)}) \odot \mathbf{W}^{(\ell+1)T} \delta^{(\ell+1)}$
Standardization	Support Vector Machine (SVM)	Imbalance	Gradient: $\nabla_{\mathbf{W}^{(\ell)}} \ell(\mathbf{W}; \mathbf{y}, \mathbf{x}) = \delta^{(\ell)} \mathbf{v}^{(\ell-1)T}$
Centered data with unit variance: $\tilde{x}_i = \frac{x_i - \hat{\mu}}{\hat{\sigma}}$	Hinge loss: $l_H(w; x_i, y_i) = \max(0, 1 - y_i w^T x_i)$	up-/downsampling	Learning with momentum
$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n x_i$, $\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{\mu})^2$	$\nabla_w l_H(w; y, x) = \begin{cases} 0 & \text{if } y_i w^T x_i \geq 1 \\ -y_i x_i & \text{otherwise} \end{cases}$	Cost-Sensitive Classification	$a \leftarrow m \cdot a + \eta_t \nabla_{\mathbf{W}} l(\mathbf{W}; y, x); W_{t+1} \leftarrow W_t - a$
Cross-Validation	$w^* = \underset{w}{\text{argmin}}_w l_H(w; x_i, y_i) + \lambda \ w\ _2^2$	Scale loss by cost: $l_{CS}(w; x, y) = c_{\pm} l(w; x, y)$	Clustering
For all models m , for all $i \in \{1, \dots, k\}$ do:	Kernels	Metrics	k-mean
1. Split data: $D = D_{\text{train}}^{(i)} \uplus D_{\text{test}}^{(i)}$ (Monte-Carlo or k-Fold) 2. Train model: $\hat{w}_{i,m} = \underset{w}{\text{argmin}}_w \hat{R}_{\text{train}}^{(i)}(w)$	efficient, implicit inner products	$n = n_+ + n_-$, $n_+ = TP + FN$, $n_- = TN + FP$	$\hat{R}(\mu) = \sum_{i=1}^n \min_{j \in \{1, \dots, k\}} \ x_i - \mu_j\ _2^2$
3. Estimate error: $\hat{R}_m^{(i)} = \hat{R}_{\text{test}}^{(i)}(\hat{w}_{i,m})$	Properties of kernel	Accuracy: $\frac{TP+TN}{TP+FP}$, Precision: $\frac{TP}{TP+FP}$	$\hat{\mu} = \underset{\mu}{\text{argmin}}_{\mu} \hat{R}(\mu)$...non-convex, NP-hard
Select best model: $\hat{m} = \underset{m}{\text{argmin}}_m \frac{1}{k} \sum_{i=1}^k \hat{R}_m^{(i)}$	$k : X \times X \rightarrow \mathbb{R}$, k must be some inner product (symmetric, positive-definite, linear) for some space	Recall/TPR: $\frac{TP}{n_+}$, FPR: $\frac{FP}{n_-}$	Lloyd's Heuristic: 1. Initialize cluster centers $\mu^{(0)} = [\mu_1^{(0)}, \dots, \mu_k^{(0)}]$ While not converged:
Gradient Descent	V . i.e. $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_V \stackrel{\text{Eucl.}}{=} \phi(\mathbf{x})^T \phi(\mathbf{x}')$ and $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$	F1 score: $\frac{2TP}{2TP+FP+FN} = \frac{2}{\frac{1}{\text{prec}} + \frac{1}{\text{rec}}}$	$\mu^{(0)} = [\mu_1^{(0)}, \dots, \mu_k^{(0)}]$ While not converged:
1. Pick arbitrary $w_0 \in \mathbb{R}^d$	Kernel matrix	ROC Curve: $y = \text{TPR}$, $x = \text{FPR}$	1. Assign points $z_i^{(t)} \leftarrow \underset{j}{\text{argmin}}_j \ \mathbf{x}_i - \mu_j^{(t-1)}\ _2^2$
2. $w_{t+1} = w_t - \eta_t \nabla \hat{R}(w_t)$	$K = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \dots & k(x_n, x_n) \end{bmatrix}$	Multi-class	2. Update centers $\mu_j^{(t)} \leftarrow \frac{1}{n_j} \sum_{i: z_i^{(t)} = j} \mathbf{x}_i$
Stochastic Gradient Descent (SGD)	Positive semi-definite matrices \Leftrightarrow kernels k	one-vs-all (c), one-vs-one ($\frac{c(c-1)}{2}$), encoding	k-Means++: Start with random data point as center and add centers randomly, proportionally to the squared distance to closest center.
1. Pick arbitrary $w_0 \in \mathbb{R}^d$	Important kernels	Multi-class Hinge loss	Dimension reduction
2. $w_{t+1} = w_t - \eta_t \nabla_w l(w; x', y')$, with u.a.r. data point $(x', y') \in D$	Linear: $k(x, y) = x^T y$	$l_{MC-H}(w^{(1)}, \dots, w^{(c)}; x, y) = \max(0, 1 + \max_{j \in \{1, \dots, y-1, y+1, \dots, c\}} w^{(j)T} x - w^{(y)T} x)$	PCA
Regression	Polynomial: $k(x, y) = (x^T y + 1)^d$	Neural networks	$D = x_1, \dots, x_n \subset \mathbb{R}^d$, $\Sigma = \frac{1}{n} \sum_{i=1}^n x_i x_i^T$, $\mu = 0$
Solve $w^* = \underset{w}{\text{argmin}}_w \hat{R}(w) + \lambda C(w)$	Gaussian: $k(x, y) = \exp(-\ x - y\ _2^2 / (2h^2))$	Parameterize feature map with θ : $\phi(x, \theta) = \varphi(\theta^T x) = \varphi(z)$ (activation function φ)	$(W, z_1, \dots, z_n) = \underset{W}{\text{argmin}} \sum_{i=1}^n \ W z_i - x_i\ _2^2$,
Linear Regression	Laplacian: $k(x, y) = \exp(-\ x - y\ _1 / h)$	$\Rightarrow w^* = \underset{w, \theta}{\text{argmin}}_{w, \theta} \sum_{i=1}^n l(y_i; \sum_{j=1}^m w_j \phi(x_i, \theta_j))$	$W = (v_1 \dots v_k) \in \mathbb{R}^{d \times k}$, orthogonal; $z_i = W^T x_i$
$\hat{R}(w) = \sum_{i=1}^n (y_i - w^T x_i)^2 = \ Xw - y\ _2^2$	Composition rules	$f(x; w, \theta_{1:d}) = \sum_{j=1}^m w_j \varphi(\theta_j^T x) = w^T \varphi(\Theta x)$	v_i are the eigenvectors of Σ
$\nabla_w \hat{R}(w) = -2 \sum_{i=1}^n (y_i - w^T x_i) \cdot x_i$	Valid kernels k_1, k_2 , also valid kernels: $k_1 + k_2$; $k_1 \cdot k_2$;	Activation functions	Kernel PCA
$w^* = (X^T X)^{-1} X^T y$	$c \cdot k_1$, $c > 0$; $f(k_1)$ if f polynomial with pos. coeffs. or exponential	Sigmoid: $\frac{1}{1 + \exp(-z)}$, $\phi'(z) = (1 - \phi(z)) \cdot \phi(z)$	Kernel PC: $\alpha^{(1)}, \dots, \alpha^{(k)} \in \mathbb{R}^n$, $\alpha^{(i)} = \frac{1}{\sqrt{\lambda_i}} v_i$,
$\mathbf{E}[w^*] = w$, $\mathbf{V}[w^*] = (X^T X)^{-1} \sigma^2$	Reformulating the perceptron	$\tanh: \varphi(z) = \tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$	$K = \sum_{i=1}^n \lambda_i v_i v_i^T$, $\lambda_1 \geq \dots \geq \lambda_d \geq 0$
Ridge regression	Ansatz: $w^* \in \text{span}(X) \Rightarrow w = \sum_{j=1}^n \alpha_j y_j x_j$	ReLU: $\varphi(z) = \max(z, 0)$	New point: $\hat{z} = f(\hat{x}) = \sum_{j=1}^n \alpha_j^{(i)} k(\hat{x}, x_j)$
$\hat{R}(w) = \sum_{i=1}^n (y_i - w^T x_i)^2 + \lambda \ w\ _2^2$	$\alpha^* = \underset{\alpha \in \mathbb{R}^n}{\text{argmin}} \sum_{i=1}^n \max(0, -\sum_{j=1}^n \alpha_j y_i y_j x_i^T x_j)$	Forward Propagation	Autoencoders
$\nabla_w \hat{R}(w) = -2 \sum_{i=1}^n (y_i - w^T x_i) \cdot x_i + 2\lambda w$	Kernelized perceptron and SVM	Input layer: $\mathbf{v}^{(0)} = \mathbf{x}$	Find identity function: $x \approx f(x; \theta)$
$w^* = (X^T X + \lambda I)^{-1} X^T y$	Use $\alpha^T k_i$ instead of $w^T x_i$,	Hidden layers: $\mathbf{z}^{(\ell)} = \mathbf{W}^{(\ell)} \mathbf{v}^{(\ell-1)}$, $\mathbf{v}^{(\ell)} = \phi(\mathbf{z}^{(\ell)})$	$f(x; \theta) = f_{\text{decode}}(f_{\text{encode}}(x; \theta_{\text{encode}}); \theta_{\text{decode}})$
$\mathbf{E}[w^*] = (X^T X + \lambda I)^{-1} (X^T X) w$	use $\alpha^T D_y K D_y \alpha$ instead of $\ w\ _2^2$	Output layer: $f = \mathbf{W}^{(L)} \mathbf{v}^{(L-1)}$	Probabilistic modeling
$\mathbf{V}[w^*] = \sigma^2 (X^T X + \lambda I)^{-1} (X^T X) [(X^T X + \lambda I)^{-1}]^T$	$k_i = [y_1 k(x_i, x_1), \dots, y_n k(x_i, x_n)]$, $D_y = \text{diag}(y)$	SGD for ANNs	Find $h : X \rightarrow Y$ that min. pred. error:
L1-regularized regression (Lasso)	Prediction: $\hat{y} = \text{sign}(\sum_{i=1}^n \alpha_i y_i k(x_i, \hat{x}))$ SGD update: $\alpha_{t+1} = \alpha_t$, if mispredicted: $\alpha_{t+1,i} = \alpha_{t,i} + \eta_t$ (c.f. updating weights towards mispredicted point)		$R(h) = \int P(x, y) l(y; h(x)) \partial y x \partial y = \mathbb{E}_{x,y} [l(y; h(x))]$
$\hat{R}(w) = \sum_{i=1}^n (y_i - w^T x_i)^2 + \lambda \ w\ _1$			For least squares regression
Classification			Best h : $h^*(x) = \mathbb{E}[Y X=x]$
Solve $w^* = \underset{w}{\text{argmin}}_w l(w; x_i, y_i)$; loss function l		For random (\mathbf{x}, \mathbf{y}) , $\mathbf{W}_{t+1} = \mathbf{W}_t - \eta_t \nabla_{\mathbf{W}} \ell(\mathbf{W}; \mathbf{x}, \mathbf{y})$	Pred.: $\hat{y} = \hat{\mathbb{E}}[Y X=\hat{x}] = \int \hat{P}(y X=\hat{x}) y \partial y$

Maximum Likelihood Estimation (MLE)

$\theta^* = \operatorname{argmax}_{\theta} \hat{P}(y_1, \dots, y_n | x_1, \dots, x_n, \theta)$
E.g. lin. + Gauss: $y_i = w^T x_i + \varepsilon_i, \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$
i.e. $y_i \sim \mathcal{N}(w^T x_i, \sigma^2)$, With MLE (use
 $\operatorname{argmin} -\log$): $w^* = \operatorname{argmin}_w \sum (y_i - w^T x_i)^2$

Bias/Variance/Noise

Prediction error = $\text{Bias}^2 + \text{Variance} + \text{Noise}$

Maximum a posteriori estimate (MAP)

$$\hat{\theta}_{\text{MAP}}(x) = \operatorname{argmax}_{\theta} g(\theta) \prod_{i=1}^n f(x | \theta)$$

Logistic regression

Link func.: $\sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)}$ (Sigmoid)

$P(y|x, w) = \text{Ber}(y; \sigma(w^T x)) = \frac{1}{1 + \exp(-yw^T x)}$ Classifi-

cation: Use $P(y|x, w)$, predict most likely class label.

MLE: $\operatorname{argmax}_w P(y_{1:n} | w, x_{1:n})$

$\Rightarrow w^* = \operatorname{argmin}_w \sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i))$

SGD update: $w = w + \eta_{t,yx} \hat{P}(Y = -y | w, x)$

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

MAP: Gauss. prior $\Rightarrow \|w\|_2^2$, Lap. p. $\Rightarrow \|w\|_1$

SGD: $w = w(1 - 2\lambda \eta_t) + \eta_{t,yx} \hat{P}(Y = -y | w, x)$

Bayesian decision theory

- Conditional distribution over labels $P(y|x)$

- Set of actions A

- Cost function $C: Y \times A \rightarrow \mathbb{R}$

$a^* = \operatorname{argmin}_{a \in A} \mathbb{E}[C(y, a) | x]$

Calculate \mathbb{E} via sum/integral.

Classification: $C(y, a) = [y \neq a]$; asymmetric:

$$C(y, a) = \begin{cases} c_{FP}, & \text{if } y = -1, a = +1 \\ c_{FN}, & \text{if } y = +1, a = -1 \\ 0, & \text{otherwise} \end{cases}$$

Regression: $C(y, a) = (y - a)^2$; asymmetric: MLE for feature distr.: $\hat{P}(X_i = c | Y = y) = \theta_{c|y}^{(i)}$

$C(y, a) = c_1 \max(y - a, 0) + c_2 \max(a - y, 0)$

E.g. $y \in \{-1, +1\}$, predict + if $c_+ < c_-$, $c_+ = \theta_{c|y}^{(i)} = \frac{\text{Count}(X_i = c, Y = y)}{\text{Count}(Y = y)}$

$\mathbb{E}(C(y, +1) | x) = P(y = 1 | x) \cdot 0 + P(y = -1 | x) \cdot c_{FP}$, Prediction: $y^* = \operatorname{argmax}_y \hat{P}(y | x)$

c_- likewise

Discriminative vs. generative modeling

Discr. estimate $P(y|x)$, generative $P(y, x)$. Approach (generative): $P(x, y) = P(x|y) \cdot P(y)$

Naive Bayes

- Estimate prior on labels $P(y)$

- Estimate cond. distr. $P(x|y)$ for each class y

- Pred. using Bayes: $P(y|x) = \frac{P(y)P(x|y)}{P(x)}$

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

Examples

MLE for $P(y) = p = \frac{n_+}{n}$

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

$$\hat{\mu}_{i,y} = \frac{1}{n_y} \sum_{x \in D_{x_i|y}} x$$

$$\hat{\sigma}_{i,y}^2 = \frac{1}{n_y} \sum_{x \in D_{x_i|y}} (x - \hat{\mu}_{i,y})^2$$

MLE for Poi.: $\lambda = \text{avg}(x_i)$

$$\mathbb{R}^d: P(X = x | Y = y) = \prod_{i=1}^d \text{Pois}(\lambda_y^{(i)}, x^{(i)})$$

Deriving decision rule

$$P(y|x) = \frac{1}{2} P(y) P(x|y), Z = \sum_y P(y) P(x|y)$$

$$y^* = \operatorname{amax}_y P(y|x) = \operatorname{amax}_y P(y) \prod_{i=1}^d P(x_i | y)$$

Gaussian Bayes Classifier

$$\hat{P}(x|y) = \mathcal{N}(x; \hat{\mu}_y, \hat{\Sigma}_y)$$

$$\hat{P}(Y = y) = \hat{p}_y = \frac{n_y}{n}$$

$$\hat{\mu}_y = \frac{1}{n_y} \sum_{i: y_i = y} x_i \in \mathbb{R}^d$$

$$\hat{\Sigma}_y = \frac{1}{n_y} \sum_{i: y_i = y} (x_i - \hat{\mu}_y)(x_i - \hat{\mu}_y)^T \in \mathbb{R}^{d \times d}$$

Fisher's LDA (c=2)

Assume: $p = 0.5$; $\hat{\Sigma}_- = \hat{\Sigma}_+ = \hat{\Sigma}$

discriminant function: $f(x) = \log \frac{p}{1-p} +$

$$\frac{1}{2} [\log \frac{|\hat{\Sigma}_-|}{|\hat{\Sigma}_+|} + ((x - \hat{\mu}_-)^T \hat{\Sigma}_-^{-1} (x - \hat{\mu}_-) -$$

$$((x - \hat{\mu}_+)^T \hat{\Sigma}_+^{-1} (x - \hat{\mu}_+))]$$

Predict: $y = \operatorname{sign}(f(x)) = \operatorname{sign}(w^T x + w_0)$

$$w = \hat{\Sigma}^{-1} (\hat{\mu}_+ - \hat{\mu}_-);$$

$$w_0 = \frac{1}{2} (\hat{\mu}_-^T \hat{\Sigma}^{-1} \hat{\mu}_- - \hat{\mu}_+^T \hat{\Sigma}^{-1} \hat{\mu}_+)$$

Outlier Detection

$$P(x) \leq \tau$$

Categorical Naive Bayes Classifier

MLE for feature distr.: $\hat{P}(X_i = c | Y = y) = \theta_{c|y}^{(i)}$

$$\theta_{c|y}^{(i)} = \frac{\text{Count}(X_i = c, Y = y)}{\text{Count}(Y = y)}$$

Prediction: $y^* = \operatorname{argmax}_y \hat{P}(y | x)$

Missing data

Mixture modeling

Model each c. as probability distr. $P(x | \theta_j)$

$$P(D | \theta) = \prod_{i=1}^n \sum_{j=1}^k w_j P(x_i | \theta_j)$$

$$L(w, \theta) = -\sum_{i=1}^n \log \sum_{j=1}^k w_j P(x_i | \theta_j)$$

Gaussian-Mixture Bayes classifiers

Estimate prior $P(y)$; Est. cond. distr. for each class: $\nabla_x ||x||_2^2 = 2x$

$$P(x|y) = \sum_{j=1}^{k_y} w_j^{(y)} \mathcal{N}(x; \mu_j^{(y)}, \Sigma_j^{(y)})$$

Hard-EM algorithm

Initialize parameters $\theta^{(0)}$

E-step: Predict most likely class for each point:

$$z_i^{(t)} = \operatorname{argmax}_z P(z | x_i, \theta^{(t-1)})$$

$$= \operatorname{argmax}_z P(z | \theta^{(t-1)}) P(x_i | z, \theta^{(t-1)});$$

M-step: Compute the MLE:

$$\theta^{(t)} = \operatorname{argmax}_{\theta} P(D^{(t)} | \theta), \text{ i.e. } \mu_j^{(t)} = \frac{1}{n_j} \sum_{i: z_i = j} x_i$$

Soft-EM algorithm

E-step: Calc p for each point and cls.: $\gamma_j^{(t)}(x_i)$

M-step: Fit clusters to weighted data points:

$$w_j^{(t)} = \frac{1}{n} \sum_{i=1}^n \gamma_j^{(t)}(x_i); \mu_j^{(t)} = \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i) x_i}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)}$$

$$\sigma_j^{(t)} = \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i) (x_i - \mu_j^{(t)})^T (x_i - \mu_j^{(t)})}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)}$$

$$\theta^{(t)} = \operatorname{argmax}_{\theta} \sum_{i=1}^n \sum_{z_i=1}^k \gamma_{z_i}(x_i) \log P(x_i, z_i | \theta)$$

Soft-EM for semi-supervised learning

labeled y_i : $\gamma_j^{(t)}(x_i) = [j = y_i]$, unlabeled:

$$\gamma_j^{(t)}(x_i) = P(Z = j | x_i, \mu^{(t-1)}, \Sigma^{(t-1)}, w^{(t-1)})$$

Useful Math

Calculus

$$F'(x) = f'(g(x))g'(x)$$

Probabilities

$$\mathbb{E}_x[X] = \begin{cases} \int x \cdot p(x) dx & \text{if continuous} \\ \sum x \cdot p(x) & \text{otherwise} \end{cases}$$

$$\text{Var}[X] = \mathbb{E}[(X - \mu_X)^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$$

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}; p(Z|X, \theta) = \frac{p(X, Z | \theta)}{p(X | \theta)}$$

$$P(x, y) = P(y|x) \cdot P(x) = P(x|y) \cdot P(y)$$

$$\mathbb{E}_x[b + cX] = b + c \cdot \mathbb{E}_x[X]$$

$$\mathbb{E}_x[b + CX] = b + C \cdot \mathbb{E}_x[X], \quad C \in \mathbb{R}^{n \times n}$$

$$\mathbb{V}_x[b + cX] = c^2 \mathbb{V}_x[X]$$

$$\mathbb{V}_x[b + CX] = C \mathbb{V}_x[X] C^T, \quad C \in \mathbb{R}^{n \times n}$$

$$\text{Cov}[X, Y] = \mathbb{E}[(X - \mathbb{E}(X))(Y - \mathbb{E}(Y))] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$$

$$\mathbb{V}[X + Y] = \mathbb{V}[X] + \mathbb{V}[Y] + 2\text{Cov}[X, Y]$$

Bayes Rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(B|A)P(A)}{\sum_A P(B|A)P(A)}$$

P-Norm

$$||x||_p = (\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}, 1 \leq p < \infty$$

Some gradients

$$\nabla_x ||x||_2^2 = 2x$$

$$-f(x) = x^T A x; \nabla_x f(x) = (A + A^T)x$$

e.g. $\nabla_w \log(1 + \exp(-yw^T x)) =$

$$\frac{1}{1 + \exp(-yw^T x)} \cdot \exp(-yw^T x) \cdot (-yx) = \frac{1}{1 + \exp(yw^T x)} \cdot (-yx)$$

Invertible/nonsingular Matrices

$A^{m \times m} : A^{-1}A = I_d = AA^{-1}$ only if $\det(A) \neq 0$;
 $Ax = 0$ has only trivial solution $x = 0$.

Orthogonal Matrices

$$A^{m \times m} : A^T A = I_d = AA^T \Leftrightarrow A^T = A^{-1}$$

Symmetric Positive Definite Matrices

Symmetric: $A^{n \times n} : A^T = A$, symmetric positive definite if: $\forall x \setminus \{0\} \in \mathbb{R}^n : x^T A x > 0$ (semi-definite if: ≥ 0) \Leftrightarrow all eigenvalues of A are positive.

Eigendecomposition

$AP = PD \Leftrightarrow A = PDP^{-1}$ iff eigenvectors of A form a basis in \mathbb{R}^n . D diagonal matrix of eigenvalues, Eigenvectors in P . $Ap = \lambda p$

Cholesky decomposition

$A^{n \times n} : A = LL^T$, symmetric and positive definite.

Singular value decomposition

$A = U \Sigma V^T$; $A^{m \times n}$; $U^{m \times m}$, $V^{n \times n} : U, V$ orthogonal and $\Sigma^{m \times n}$ diagonal with singular values

$$\sigma = \sqrt{\lambda(A^T A)} \quad Av = \sigma u$$