1 General

P-Norm: $||x||_p = (\Sigma_{i=1}^n |x_i|^p)^{\frac{1}{p}}$

Frobenius Norm: $||A||_F = \sqrt{\Sigma_{i,j} a_{ij}^2}$

Derivation rules: Chain rule: $D(f(g(x))) = Df(g(x)) * Dg(x)$

positive definiteness: A is p.s.d., then A is a real symmetric matrix and $x^T A x \geq 0$ for all x

Joint distribution: X, Y are RVs $F_{X,Y}(x,y) = \mathbb{P}(X \leq x, Y \leq y)$

Joint density: $f_{X,Y}(x,y) = \frac{\delta^2 F}{\delta x \delta y}(x,y)$

Conditional Probability: $\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$

Law of total probability: $\mathbb{P}(B) = \Sigma_{i=1}^n \mathbb{P}(B|A_i)\mathbb{P}(A_i)$

Bayes rule: $\mathbb{P}(A|B) = \mathbb{P}(B|A)\frac{\mathbb{P}(A)}{\mathbb{P}(B)}$

Variance: $Var(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2 \geq 0$

Convexity: A twice differentiable function $f: \mathbb{R}^d \to \mathbb{R}$ is convex iff for any $x \in \mathbb{R}^d$ its Hessian is p.s.d

Convex functions are closed under addition

2 Regression: Predict real valued labels

Linear Regression: Goal: Measure distance between predicted and target values $f(x) = w_1 x_1 + \cdots + w_d x_d + w_0 = \widetilde{w}^T \widetilde{x}$ with $\widetilde{w} = [w_1 \cdots w_d, \ w_0]$ and $\widetilde{x} = [x_1 \cdots x_d, \ 1]$

Residual: $r_i = y_i - w^T x_i, \ x_i \in \mathbb{R}^d, \ y_i \in \mathbb{R}$

Cost / Objective function (is convex): $\hat{R}(w) = \sum_{i=1}^n r_i^2 = \sum_{i=1}^n (y_i - w^T x_i)^2$

Optimal weights: $w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^n (y_i - w^T x_i)^2$

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

Gradient: $\nabla_w \hat{R}(w) = [\frac{\delta}{\delta w_1}\hat{R}(w) \cdots \frac{\delta}{\delta w_d}\hat{R}(w)] = -2\sum_{i=1}^n r_i x_i^T$

Non-linear functions: $f(x) = \sum_{i=1}^D w_i \phi_i(x)$

Fisher consistency: Given a surrogate loss function $\psi : Y \times S \to \mathbb{R}$, the surrogate is said to be consistent with respect to the loss $L : Y \times S \to \mathbb{R}$, if every minimizer f of the surrogate risk function $R_\psi(f)$ is also a minimizer of the risk function $R_L(f)$. E.g. the hinge and the logistic losses are consistent with respect to the 0-1 loss.

Classification losses: $L_{perceptron} : \{-1, 1\} \times \mathbb{R} \to \mathbb{R} : y, f(x) \to \max(0, -yf(x))$

Find the best separation hyperplane

$L_{hinge} : \{-1, 1\} \times \mathbb{R} \to \mathbb{R} : y, f(x) \to \max(0, 1 - yf(x))$

Find large separation margin

$L_{perceptron} : \{-1, 1\} \times \mathbb{R} \to \mathbb{R} : y, f(x) \to \max log(1 + exp(-yf(x))$

Link to cross entropy and probabilistic interpretation

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

Recall/Sensitivity/True positive rate/TPR: $\frac{TP}{TP+FN}$

Specify or True negative rate/TNR: $\frac{TN}{TN+FP}$

F1 score: $2 * \frac{Precision * Recall}{Precision + Recall}$

Convex function: $f : \mathbb{R}^d \to \mathbb{R}$ is convex $\Leftrightarrow x_1, x_2 \in \mathbb{R}^d, \lambda \in [0,1]$:
$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$

Gradient Descent: 1. Start at an arbitrary $w_0 \in \mathbb{R}^d$
2. For $t = 1, 2, ...$ do $w_{t+1} = w_t - \eta_t \nabla \hat{R}(w_t)$

Gaussian/Normal Distribution: $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$

Multivariate Gaussian: $f(x) = \frac{1}{2\pi\sqrt{|\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$

$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{pmatrix}, \ \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$

Empirical risk minimization: Assumption: Data set generated iid from unknown distribution P: $(x_i, y_i) \sim P(X, Y)$.

True risk: $R(w) = \int P(x,y)(y - w^T x)^2 dx dy = \mathbb{E}_{x,y}[(y - w^T x)^2]$

Empirical risk: $\hat{R}_D(w) = \frac{1}{|D|}\sum_{(x,y)\in D}(y - w^T x)^2$

Generalization error: $|R(w) - \hat{R}_D(w)|$

Uniform convergence: $\sup_w |R(w) - \hat{R}_D(w)| \to 0$ as $|D| \to 0$

In general, it holds that: $\mathbb{E}_D[\hat{R}_D(\hat{w}_D)] \leq \mathbb{E}_D[R(\hat{w}_D)]$, where $\hat{w}_D = \operatorname{argmin}_w \hat{R}_D(w)$.

Cross-validation: For each model m
For i = 1:k
1. Split data: $D = D_{train}^{(i)} \uplus D_{val}^{(i)}$
2. Train model: $\hat{w}_{i,m} = \operatorname{argmin}_w \hat{R}_{train}^{(i)}(w)$
3. Estimate error: $\hat{R}_m^{(i)} = \hat{R}_{val}^{(i)}(\hat{w}_{i,m})$

After all iterations, select model: $\hat{m} = \operatorname{argmin}_w \frac{1}{k}\sum_{i=1}^k \hat{R}_m^{(i)}$

Ridge regression: Regularization(corresponds to MAP estimation): $\min_w \frac{1}{n}\sum_{i=1}^n (y_i - w^T x_i)^2 + \lambda ||w||_2^2 = \operatorname{argmax}_w P(w)\Pi_i P(y_i|x_i w)$

Sparse regression (L1, convex) encourages coefficients to be exactly 0 - automatic feature selection

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

Gradient: $\nabla_w(\frac{1}{n}\sum_{i=1}^n (y_i - w^T x_i)^2 + \lambda ||w||_2^2) = \nabla_w \hat{R}(w) + 2\lambda w$

Standardization: Goal: each feature: $\mu = 0$, $\sigma^2 = 1$: $\tilde{x}_{i,j} = \frac{(x_{i,j} - \hat{\mu}_j)}{\hat{\sigma}_j}$

$\hat{\mu}_j = \frac{1}{n}\sum_{i=1}^n x_{i,j}, \ \hat{\sigma}_j^2 = \frac{1}{n}\sum_{i=1}^n (x_{i,j} - \hat{\mu}_j)^2$

3 Classification

$h(x) = \text{sign}(w^T x)$

Losses: 0/1 loss: $\ell_{0/1}(w; x, y) = [y \neq \text{sign}(w^T x)]$

Perceptron loss: $\ell_p(w; x, y) = \max(0, -yw^T x)$

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

$\nabla_w \ell_p(w, x_i, y_i) = \begin{cases} 0, & \text{if } w^T x_i y_i \geq 0 \text{ (1 if } \ell_H) \\ -y_i x_i & \text{else} \end{cases}$

SGD: GD requires sum over all data, slow for large datasets.
1. Choose random initial $w_0 \in \mathbb{R}^d$
2. For $t = 1, 2, \ldots$ do:
(a) Choose $(x, y) \in D$ u.a.r (w/ replacement)
(b) Set $w_{t+1} = w_t - \eta_t \nabla \ell(w_t; x, y)$

SGD converges if $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$.

Mini-batch: Choose multiple datapoints at random; may converge faster.

Perceptron: SGD with ℓ_p and $\eta = 1$.
$\hat{w} = \arg\min_w \frac{1}{n}\sum_{i=1}^n \ell_p(w; x_i, y_i)$
If data linearly separable finds separator.

SVM: SGD with ℓ_H and regularization.
$\hat{w} = \arg\min_w \frac{1}{n}\sum_{i=1}^n \ell_H(w; x_i, y_i) + \lambda ||w||_2^2$
$w_{t+1} = w_t(1 - 2\eta_t\lambda) + \eta_t x_i y_i [y_i w_T x_i < 1]$
Often $\eta_t = \frac{1}{\lambda t}$. Works on non-linearly separable data, finds best separator w.r.t. ℓ_H.

4 ???orthogonal distance

: Let $w \in \mathbb{R}^d$ and $H = \{x \in \mathbb{R}^d | \langle w, z \rangle = 0\}$ be a hyperplane. The orthogonal distance of a point z $\in \mathbb{R}^d$ to H can be computed as $\frac{|\langle w, z \rangle|}{||w||}$. Specifically, if w is a unit vector, the inner product $\langle w, z \rangle$ directly gives the distance of z to H.

5 Feature selection

Naive: try all subsets, and pick best (via cross-validation)

Greedy: Greedily add (or remove) features to maximize cross-validated prediction accuracy w.r.t. cost $c : V \Rightarrow \mathbb{R}$ of using features in subset of V. Forward (start with empty set) is faster, but backward is more resilient to "dependent" features. Applies to any method, but slow b/c trains many models and can be suboptimal.

L1-Regularization: regularize loss with $\|w\|_1$, automatic feature selection. Can be used for regression (Lasso), classification (L1-SVM) by replacing $||w||_2^2$ with $||w||_1$. Only works for linear models, but is fast.

6 Class imbalance

Downsample loses data but fast, upsample random pertubation maybe unsafe. Use cost-sensitive metrics controlling tradeoff:
$\ell_{CS} = c_y \ell(w; x, y)$

		True label		
Pred. lab.		Positive	Negative	Σ
	Positive	TP	FP	p_+
	Negative	FN	TN	p_-
	Σ	n_+	n_-	

Accuracy bad metric for imbalanced data.

Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Precision	
TPR, Recall	$\frac{TP}{TP+FN}$	F1 score	
FPR	$\frac{FP}{TN+FP}$		

Precision Recall Curve: Precision (y-axis) vs. Recall (x-axis).
Precision = 1 and Recall = 1 is optimal. Area under curve (AUC) can be used for comparison of algos.

Receiver Operator Characteristic (ROC) Curve: TPR (y-axis) vs. FPR (x-axis).
Random guessing achieves TPR = FPR line. TPR > FPR is better than random guessing. TPR = 1 and FPR = 0 is optimal. Area under curve (AUC) can be used for comparison of algos.
Theorem: Alg 1 dominates Alg 2 in terms of ROC curve $\Leftrightarrow$ Alg 1 dominates Alg 2 in terms of Precision Recall curve.

One-vs-all: c classifiers, one for each class, pick highest confidence. Class may not be lin. sep. from all others. Note Scaling + Imbalance.

One-vs-one $c(c-1)/2$ classifiers, voting scheme with highest number of positive prediction wins: no confidence needed.

Ideally $\mathcal{O}(\log c)$ classifiers, theoretical optimum.

Multi-class SVM: Maintains c weight vectors, want $w^{(y)T}x \geq \max\{w^{(i)T}x\} + 1$ for correct label y.
$\ell_{MC-H}(w^{(1)}, ..., w^{(c)}; x, y) = \max(0, 1 + \max_{j \neq y} w^{(j)T}x - w^{(y)T}x)$

7 Kernels

Kernel trick: 1. Express problem s.t. it only depends on inner products $x_j^T x_i$.
2. Replace inner products by kernels.

Reformulate problem: Fundamental insight: Optimal separating hyperplane lives in the span of data: $\hat{w} = \sum_{i=1}^n \alpha_i y_i x_i$

Perceptron example: $\hat{w} = \arg\min_w \frac{1}{n}\sum_{i=1}^n \max(0, -y_i w^T x_i)$

$\hat{a} = \arg\min_\alpha \frac{1}{n}\sum_{i=1}^n \max(0, -y_i \left(\sum_{j=1}^n \alpha_j y_j\right.$

$\hat{a} = \arg\min_\alpha \frac{1}{n}\sum_{i=1}^n \max(0, -\sum_{j=1}^n \alpha_j y_i y_j x$

Replace inner products: For some feature transform $\phi : x \mapsto \phi(x)$, kernels solve $\phi(x)^T \phi(x')$ efficiently as $k(x, x')$.

Perceptron example (Training):
1. Initialize $\alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$
2. For $t = 1, 2, \ldots$
   (a) Pick $(x_i, y_i) \in D$ u.a.r.

(b) Predict $\hat{y} = \text{sign}\left(\sum_{j=1}^n \alpha_j y_j k(x_j, x_i)\right)$

(c) If $\hat{y} \neq y_i$ set $\alpha_i \Leftarrow \alpha_i + \eta_t$

Kernel properties: $k : X \times X \Rightarrow R$ must be *symmetric*: $k(x, x') = k(x', x)$
Gram matrix K must be p.s.d. ($\forall x. x^T K x \geq 0$) for any n, any set $\{x_1, \ldots, x_n\} \subseteq X$. All p.s.d. matrices are some kernel and all kernels have Gram matrix.

$$K = \begin{bmatrix} k(x_1, x_1) & \ldots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \ldots & k(x_n, x_n) \end{bmatrix}$$

For k_1, k_2 kernel, $c > 0$ and f polyn. with pos. coef. or exp. $k_1 + k_2$, $k_1 \cdot k_2$, $c \cdot k_1$ and $f(k_1(x, x'))$ are kernels.

Poly. degree $= d$	$(x^T x')^d$
Poly. degree $\leq d$	$(x^T x' + 1)^d$
Gaussian (RBF)	$\exp(-\lVert x - x' \rVert_2^2 / (2h^2))$
Lapacian	$\exp(-\lVert x - x' \rVert_1 / h)$

Note: $h > 0$ is bandwidth, $h \to 0$ overfits.

k-NN: $y = \text{sign}\left(\sum_{i=1}^n y_i [x_i \text{ among k-NN of } x]\right)$
No training, but depends on all data.

$$y = \text{sign}\left(\sum_{i=1}^n y_i \alpha_i k(x_i, x)\right)$$

Can use kernel as *similarity function*: Improved performance, depends only on wrongly classified data, can capture global trends, but requires training.

Parametric vs. nonparametric learning: *Parametric* have finite set of parameters (regression, perceptron), while *nonparametric* increase complexity with size of data (kernelized perceptron, k-NN).

Can kernelize other tasks, such as SVM. Let $k_i = [y_1 k(x_1, x_i) \quad \ldots \quad y_n k(x_n, x_i)]$:
$\arg\min_\alpha \frac{1}{n} \sum_{i=1}^n \max\{0, 1 - y_i \alpha^T k_i\} + \lambda \alpha^T D_y K D_y \alpha$

Lin. reg. (training): $\hat{\alpha} = \arg\min_\alpha \frac{1}{n} \lVert \alpha^T K - y \rVert_2^2 + \lambda \alpha^T K \alpha$

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

Lin. reg. (prediction): $\hat{y} = \sum_{i=1}^n \hat{\alpha}_i k(x_i, x)$

Est. kernel parameters via CV. Choosing kernels requires domain knowledge. Deal w/ overfit by regularization.

8 Neural Networks

Parametrized feature maps + regression. Apply nonlinear activation function φ after weighted $\sum$ of inputs. Can learn nonlinear features. *Forward propagation* in layer ℓ: ($v^{(0)} = x$): $v^{(\ell)} = \varphi(z^{(\ell)}) = \varphi(W^{(\ell)} v^{(\ell-1)})$

Output layer: $y = f = W^{(L-1)} v^{(L-1)}$. ($f$ can be vector)

Weight optimization: Apply loss $\ell(y - f(x, W))$, optimize weights to minimize loss. For multi-outputs sum losses.

Sigmoid	$\frac{1}{1+\exp(-z)}$	diff. everywhere, ≈ 0 far from 0
Tanh	$\frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$	diff. everywhere, ≈ 0 far from 0
ReLu	$\max(0, z)$	not diff. at 0, $\varphi' = 1$ if $z > 0$

Back propagation: $\delta^{(L)} = l'(f) = [l'(f_1) \quad \ldots \quad l'(f_p)]$, and $\nabla_{W^{(L)}} \ell(W; y, x) = \delta^{(L)} v^{(L-1)T}$. For $\ell < L$: $\delta^{(\ell)} = \varphi'(z^{(\ell)}) \odot (W^{(\ell+1)T} \delta^{(\ell+1)})$ and $\nabla_{W^\ell} \ell(W; y, x) = \delta^{(\ell)} v^{(\ell-1)T}$

Nonconvex optimization, initialization matters! Random works well.

Glorot (tanh):	$w_{i,j} \sim \mathcal{N}(0, 1/n_{in})$
	$w_{i,j} \sim \mathcal{N}(0, 2/(n_{in} + n_{out}))$
He (ReLU):	$w_{i,j} \sim \mathcal{N}(0, 2/(n_{in}))$

Learning rate η_t *decreasing* (e.g. $min(0.1, 100/t)$) to prevent oscillation.
Momentum ($m < 1$, $a \leftarrow m \cdot a + \eta_t \nabla_W \ell(W; y, x); W \leftarrow W - a$) can avoid local minima.
Many parameters $\to$ overfitting! *Early stop* or *regularization* ($\lambda \lVert W \rVert_F^2$) to prevent. Also *dropout*, randomly set weights to 0 with prob. p, set $W = W \odot p$ after training.
Batch normalization, standardize some batch $\{x_{1\ldots m}\}$ and set $y_i = \gamma \hat{x}_i + \beta = BN_{\gamma,\beta}(x_i)$. Then $\varphi(Wx) = \varphi(W(BN_{\gamma,\beta}(x)))$.
Convolutional NN: Apply m $f \times f$ filters to $n \times n$ image, padding p and stride s: Leaves with $\alpha \times \alpha \times m$ output, where $\alpha = \frac{n+2p-f}{s}$.

9 Unsupervised Learning

k-Means clust.: Represent cluster as center, assign point $x_i \in \mathbb{R}^d$ to nearest center $\mu_j \in \mathbb{R}^d$. Squared loss (below) nonconvex.
$\hat{R}(\mu) = \sum_{i=1}^n \min_{j \in [k]} \lVert x_i - \mu_j \rVert_2^2$
Lloyd: Initialize $\mu^{(0)}$. Then assign x_i to closest center $z_i^{(t)} = \arg\min_j \lVert x_i - \mu_j^{(t-1)} \rVert_2^2$. Then update mean: $\mu_j^{(t)} = \frac{1}{|i: z_i^{(t)} = j|} \sum_{i: z_i^{(t)} = j} x_j$.
$\mathcal{O}(nkd)$ per iteration, converges pot. slowly but monotonically to local optimum $\to$ Multiple iter.
k-Means++: Let $\mu^{(0)} = x$ u.a.r. from X. Assign centers $2 \ldots k$ randomly, prop. to sq. dist. to closest sel. cent. Expected cost within $\mathcal{O}(\log k)$ of optimum.
$\Pr[\mu_j = x_i] = \frac{1}{Z} \min_{k \in [j-1]} \lVert \mu_k - x_i \rVert_2^2$
Choosing k difficult. Heuristic: When $k + 1$ yields *diminishing returns*, or *regularization* with λk. Only models circular clusters $\to$ use kernels.

Dimension Reduction: Embed $\{x_1, \ldots, x_n\}$, $x_i \in \mathbb{R}^d$ in $\mathbb{R}^k$ where $k < d$.
PCA: *Center data* $\mu = \frac{1}{n} \sum_{i=1}^n x_i = 0$ and construct empirical cov. matrix: $\Sigma = \frac{1}{n} \sum_{i=1}^n x_i x_i^T$.
PCA problem: $(W, z_1, \ldots, z_n) = \arg\min_{W, z} \sum_{i=1}^n \lVert W z_i - x_i \rVert_2^2$ where W orthogonal, $z_i \in \mathbb{R}^k$ has sol. $z_i = W^T x_i$. The solution for W is given by the k principal eigenvectors of Σ. $W = (v_1 | \ldots | v_k)$ where $\Sigma = \sum_{i=1}^d \lambda_i v_i v_i^T$ with $\lambda_1 \geq \ldots \geq \lambda_d \geq 0$.
Can apply to any matrix $X = USV^T$, first k components of V are first k principal components. Choose k by CV for feature ind., else s.t. variance is explained (see k-Means). Solve nonlinear PCA via kernels.
Kernel PCA: $w = \sum_{j=1}^n \alpha_j \phi(x_j)$, $K = \sum_{i=1}^n \lambda_i v_i v_i^T$
$\arg\max_{\lVert w \rVert_2 = 1} \sum_{i=1}^n (w^T \phi(x_i))^2 = \arg\max_{\alpha^T K \alpha = 1} \alpha^T K^T K \alpha$, Solution: $\alpha^{(i)} = v_i / (\sqrt{\lambda_i})$
New point: $z_i = w^{(i)T} x = \sum_{j=1}^n \alpha_j^{(i)} k(x, x_j)$
Centering a kernel: $K' = K - KE - EK + EKE$ where $E = \frac{1}{n}[1, \ldots, 1][1, \ldots, 1]^T$.
Complexity grows with the number of data points.
Autoencoder: NN where hidden layers usually smaller (k) than in- and output (d). Try to learn identity function. Compression from input to smallest HL, Decompression from smallest HL to output. $f(x; \theta) = f_{dec}(f_{enc}(x; \theta_1); \theta_2)$. If $\varphi(z) = z$ then autoencoder is equivalent to PCA.

10 Probabilistic modeling

Bayes optimal predictor: $h^*(x) = \mathbb{E}[Y \mid X = x]$, unattainable in pr. Can try to estimate conditional distr. $\hat{P}(Y \mid X)$.
Parametric estimation: Have $\hat{P}(Y \mid X, \theta)$, MLE:
$\theta^* = \arg\max_\theta \hat{P}(Y \mid X, \theta) = \arg\min_\theta - \sum_{i=1}^n \log \hat{P}(y_i \mid x_i, \theta)$
Ex.: Gaussian noise, lin. reg.: $y_i \sim \mathcal{N}(w^T x, \sigma^2)$. Then MLE:
$\arg\min_\theta = \frac{n}{2} \log(2\pi\sigma^2) \sum_{i=1}^n \frac{(y_i - w^T x_i)^2}{2\sigma^2}$
and MLE equivalent to LSQ estimation. In general, MLE with Gaussian noise with constant variance is equivalent to LSQ sol.
Bias Variance Tradeoff: Prediction error = Bias^2 + Variance + Noise.
Bias: Excess risk of best model considered compared to minimal achievable risk knowing $P(X, Y)$ (i.e., given infinite data).
Variance: Risk incurred due to estimating model from limited data.
Noise: Risk incurred by optimal model (i.e., irreducible error).
Trade bias and variance via model selection / regularization

Maximum A Posteriori estimation: Placing assumptions on distribution of parameter. For Gaussian noise and Gaussian prior: MAP = Ridge regression.
Regularized estimation can often be understood as MAP inference:
$\arg\min_w \sum_{i=1}^n \ell(w^T x_i; x_i, y_i) + C(w) = \arg\max_w \prod_i P(y_i | x_i, w) P(w) = \arg\max_w P(w|D)$ where $C(w) = -\log P(w)$ and $\ell(w^T x_i; x_i, y_i) = -\log P(y_i | x_i, w)$.
Bayes optimal classifier: $h^*(x) = \arg\max_y P(Y = y \mid X = x)$
Logistic regression: $P(Y = y|x) = \frac{1}{1+\exp(-y w^T x)}$. Replaces Gaussian noise assumption with Bernoulli noise: $P(y|x, w) = \text{Ber}(y; \sigma(w^T x))$. MLE: $\hat{R}(w) = \sum_{i=1}^n log(1 + exp(-y_i w^T x_i))$, is convex and $\nabla_w \ell_w = \frac{1}{1+exp(-y w^T x)} exp(-y w^T x)(-yx)$, and $exp(-y w^T x) = 1$ if misclassified. Thus $\nabla_w \ell_w = \frac{-yx}{1+exp(y w^T x)}$. With L2 regularizer, take step in direction $w(1 - 2\lambda\eta_t) - \eta_t \nabla_w \ell_w(y, x)$. Nonlinear classification via kernels: $\hat{\alpha} = \arg\min_\alpha \sum_{i=1}^n \log(1 + \exp(-y_i \alpha^T K_i)) + \lambda \alpha^T K \alpha$ and $\hat{P}(y|x, \hat{\alpha}) = \frac{1}{1+\exp(-y \sum_{j=1}^n \alpha_j k(x_j, x))}$ with $w = \sum_i \alpha_i x_i$.
Multi-class: $P(Y = i | x, w_1, \ldots, w_c) = \frac{\exp(w_i^T x)}{\sum_{j=1}^c \exp(w_j^T x)}$
Can obtain class probablities, but dense solutions

11 Decision Theory

Have $P(y \mid x)$, actions $\mathcal{A}$ and cost $C : \mathcal{Y} \times \mathcal{A} \to R$. Min. Exp. cost: $a^* = \arg\min_a \mathbb{E}_y[C(y, a) \mid x]$
Asymmetric costs: Assume $A = \{-1, +1\}$, $c_{FP}, c_{FN} > 0$. Let $\hat{P}(y \mid x) = p$, then $\mathbb{E}_y[C(y, +1)] = (1 - p) c_{FP}$ and $\mathbb{E}_y[C(y, -1)] = p c_{FN}$. Predict $+1$ when $p > \frac{c_{FP}}{c_{FP} + c_{FN}}$.
Uncertainty sampling: Ask user to label the example that we are most uncertain about: $i_t \in \arg\min_i |0.5 - \hat{P}(Y_i | x_i)|$. Active learning violates i.i.d. assumption.
MAP summary: 1. Choose likelihood function $\to$ loss function
2. Choose prior $\to$ regularizer
3. Optimize for MAP parameters, choose hyperparameters through cross-validation
4. Make predictions via Bayesian Decision Theory

12 Generative Modeling

Estimate *joint distribution* $P(X,Y)$ instead of $P(Y \mid X)$. Cond. distr. can be derived from joint: 1. Estimate $P(Y)$, 2. Estimate $P(x|y)$ for each y, 3. Use Bayes' rule: $P(y|x) = P(y)P(x|y)/P(x)$ where $P(x) = \sum_{y'} P(x,y')$. For $c = 2$ *discriminant function* $f(x) = \log \frac{P(Y=1|x)}{P(Y=-1|x)}$, which is $+1$ if $P(Y = 1 \mid x) > 0.5$.

Naive Bayes: Model class y as categorical, and features conditionally independent given y, e.g. Gaussian NB, assumes $P(x_i|y) = \mathcal{N}(x_i|\mu_{y,i},\sigma_{y,i}^2)$. Produces linear classifier, equiv. to log. reg. if assumptions are met. $f(x) = w^T x + w_0$ where $w_0 = \log \frac{p_+}{1-p_+} + \sum_{i=1}^d \frac{\mu_{-,i}^2 - \mu_{+,i}^2}{2\sigma_i^2}$ and $w_i = \frac{\mu_{+,i}-\mu_{-,i}}{\sigma_i^2}$. Due to conditional independence assumption, predictions can become overconfident. # parameters = $O(cd)$, Complexity (memory + inference) linear in d

Categorical Naive Bayes: $P(X_i = c|Y = y) = \theta_{c|y}^{(i)}$. MLE prior: $\hat{p}_y = \frac{\#y}{n}$, MLE feat. distr.: $\theta_{c|y}^{(i)} = \frac{\text{Count}(X_i=c,Y=y)}{\#y}$. Requires exponentially (in d) many parameters, Fantastic way to overfit.

Gaussian Bayes: $P(x|y) = \mathcal{N}(x;\mu_y,\Sigma_y)$. MLE prior: $\hat{p}_y = \frac{\#y}{n}$, MLE feat. distr.: $\hat{\mu}_k = \frac{1}{\#y}\sum_{i:y_i=y} x_i$ and $\hat{\Sigma}_y = \frac{1}{\#y}\sum_{i:y_i=y}(x_i - \hat{\mu}_y)(x_i - \hat{\mu}_y)^T$. Discriminant $f(x)$: $\log\frac{p}{1-p}+\frac{1}{2}\left[\log\frac{|\hat{\Sigma}_-|}{|\hat{\Sigma}_+|} + \left((x-\hat{\mu}_-)^T\hat{\Sigma}_-^{-1}(x-\hat{\mu}_-)\right) - \left((x-\hat{\mu}_+)^T\hat{\Sigma}_+^{-1}(x-\hat{\mu}_+)\right)\right]$.
Captures correlations among features, avoids overconfidence, # parameters = $O(cd^2)$, complexity quadratic in d.

Fischer's LDA: Assume $p = 0.5$ and $\hat{\Sigma}- = \hat{\Sigma}+ = \hat{\Sigma}$. Then $f(x) = x^T\hat{\Sigma}^{-1}(\hat{\mu}_+ - \hat{\mu}_-) + \frac{1}{2}\left(\hat{\mu}_-^T\hat{\Sigma}^{-1}\hat{\mu}_- - \hat{\mu}_+^T\hat{\Sigma}^{-1}\hat{\mu}_+\right)$.
Produces linear classifier, equiv. to log. reg. if assumptions are met. LDA can be viewed as a projection to a 1-dim. subspace that maximizes ratio of between-class and within-class variances. In constrast, PCA (k=1) maximizes the variance of the resulting 1-dim. projection. Generative model, can be used to detect outliers, not very robust against violation of normality of X assumption.
Can regularize: $\text{Beta}(\theta,\alpha_+,\alpha_-)$ models likelihood of θ given α_+ weight for $y = 1$ and α_- weight for $y = -1$.
Conjugate distr.: Posterior is same family as prior. Ex.: $\text{Beta}(\theta,\alpha_+,\alpha_-)$ and $\text{Beta}(\theta, n_+ + \alpha_+, n_- + \alpha_-)$. MAP estimate: $\hat{\theta} = \frac{\alpha_+ + n_+ - 1}{\alpha_+ + n_+ + \alpha_- + n_- - 2}$.

13 Generalized Mixture Models

Setting: Labels potentially unknown, want to cluster data.
Model $P(x,\theta)$ as *convex combination* of Gaussian distributions: $\sum_{i=1}^c w_i \mathcal{N}(x;\mu_i,\Sigma_i)$ with $w_i > 0$ and $\sum_i w_i = 1$.
$(\mu^*,\Sigma^*,w^*) = \arg\min -\sum_{i=1}^n \log\sum_{j=1}^k w_j\mathcal{N}(x_i;\mu_j,\Sigma_j)$.
Constraints (Σ p.s.d.) hard to maintain in SGD.
Fitting a GMM = Training a GBC without labels.
Hard-EM: E-step: Predict most likely class for all x_i: $z_i^{(t)} = \arg\max_z P(z \mid x_i, \theta^{(t-1)}) = \arg\max_z P(z|\theta^{(t-1)})P(x_i|z,\theta^{(t-1)})$ with $\theta^{(t)} = [w_{1:c}^{(t)}, \mu_{1:c}^{(t)}, \Sigma_{1:c}^{(t)}]$. M-step: Compute $\theta^{(t)}$ as MLE as for the Gaussian Bayes classifier: $\theta^{(t)} = \arg\max_\theta P(D^{(t)} \mid \theta)$.
Too much information extracted from each label, overlapping clusters not detected, fixed label when model uncertain. k-Means Algorithm is equiv to Hard-EM if $w_j = 1/k$ and $\Sigma_j = I\cdot\sigma^2$.
Soft-EM: Let $\gamma_j(x)$ be prob. that x in cluster j: $\gamma_j(x) = P(Z = j \mid x,\Sigma,\mu,w) = \frac{w_j P(x|\Sigma_j,\mu_j)}{\sum_\ell w_\ell P(x|\Sigma_\ell,\mu_\ell)}$
E-Step: Calculate prob $\gamma_j(x_i)$ for all i,j based on $\mu^{(t-1)}$, $\Sigma^{(t-1)}$ and $w^{(t-1)}$.
M-Step: Adjust parameters: $w_j^{(t)} = \frac{1}{n}\sum_{i=1}^n \gamma_j^{(t)}(x_i)$, $\mu_j^{(t)} = \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i)x_i}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)}$ and $\Sigma_j^{(t)} = \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i)(x_i-\mu_j^{(t)})(x_i-\mu_j^{(t)})^T}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)}$.
Initialization sensitive, nonconvex objective. Init. w with uniform distribution, μ by k-means++ and Σ as spherical (according to empirical variance in data). Choose k via CV.
Avoid degeneracy by adding term $\nu^2 I$ to $\Sigma_j^{(t)} \rightarrow$ Wishart-prior.
Semi-supervised learning: For points with label y_i: $\gamma_j^{(t)}(x_i) = [j = y_i]$
GANs: Objective:
$\min_{w_G}\max_{w_D}\mathbb{E}_{x\sim\text{Data}}\log D(x;w_D) + \mathbb{E}_{x\sim\mathcal{N}}\log(1 - D(G(z;w_G);w_D))$, Mode Collapse: generator produces less diverse samples than distribution, Data memorization, Simultaneous training $\rightarrow$ oscillations, Cannot compute likelihood on holdout set.