# Regression

Solve $w^*=\text{argmin } \hat{R}(w)+\lambda C(w)$

## Linear Regression

$\hat{R}(w)=\sum_{i=1}^n (y_i-w^Tx_i)^2=||y-Xw||_2^2$
$\nabla_w \hat{R}(w)=2X^T(X\hat{w}-y)$
$w^*=(X^TX)^{-1}X^Ty \quad (d\le n)$
$w^*=X^T(XX^T)^{-1}y \quad (d\ge n)$ (inf solutions, this min $||w||_2$ with full rank x.
$\hat{R}(w)$: convex matrix, $\hat{w}$ global min

## Regularization

L2: $\hat{R}(w)=\sum_{i=1}^n (y_i-w^Tx_i)^2+\lambda||w||_2^2$
$\nabla_w \hat{R}(w)=2X^T(X\hat{w}-y)+2\lambda w$
$w^*=(X^TX+\lambda I)^{-1}X^Ty$ Bayes with Gaussian prior
L1: $\hat{R}(w)=\sum_{i=1}^n (y_i-w^Tx_i)^2+\lambda||w||_1$
(MAP/Bayes): $P(w|x,y)=p(w)p(y|x,w)$
Ridge: Gaussian prior, $\lambda=\frac{\sigma^2}{\beta^2}$ $\sigma$:noise var, $\beta$:$w$ var
Lasso: $p(w)\ Laplace(0,s)$, $\lambda=\frac{2\sigma^2}{s}$

# Optimization

## Big O

$M=XX^T=O(nd^2)$, $M^{-1}=O(d^3)$ $\hat{w}:O(nd^2+d^3)$

## Gradient Descent

1. Pick arbitrary $w_0\in\mathbb{R}^d$
2. $w_{t+1}=w_t-\eta_t\nabla\hat{R}(w_t)$

differentiable L (bounded below) + careful step $\rightarrow$ converge to *stationary* point of L.
**linReg**: $X^TX$ full rank + $\eta<2/\lambda_{max}(X^TX)\rightarrow$ converge *linearly* to global min $\hat{w}$
full rank $=\lambda_{min}(X^TX)>0$ for PSD matrix
linear: $||w^t-\hat{w}||\sim C\rho^t$, $C$: constant
$\rho=\max\{1-\eta\lambda_{min}(X^TX),\quad \lambda_{max}(X^TX)-1\}$
$\eta_{opt}=\frac{2}{\lambda_{min}+\lambda_{max}}$, $\rho_{min}=\frac{\lambda_{max}-\lambda_{min}}{\lambda_{min}+\lambda_{max}}$

## Other Methods

Momentum-based method: $w^{t+1}=w^t+\alpha(w^t-w^{t-1}-\nabla L(w^t)$, $\alpha=\beta\eta$ (scaled step)
Adaptive method: adjust step-size based on coordinate change: $\frac{\eta}{\sqrt{(w^t-w^{t-1})^2+r}}$
SGD: each time choose a random subset $S$ and use average intra-subset gradient as gradient.
SGD: L may increase at some iteration but eventually converges. $S$ can have repeated elements.
epoch: partition training set into d subsets(*minibatch*). 1 epoch = going through all data once. batch size $\uparrow$: Var (to real $\nabla L$) $\downarrow$, O(n) $\uparrow$.

## Convexity

1st order: $f(y)\ge f(x)+<\nabla f(x),y-x>$
2nd order: Hessian $D^2f$ is PSD $\forall x$

$\alpha,\beta>0$, then $\alpha f+\beta g$ is convex.
$f\circ g$: $f$ is convex, $g$ is affine (linear combo). Or $f$ is non-decrease, $g$ is convex;
point-wise max of 2 convex is convex.

# Classification

Assuming: all data iid, training and test set comes from same distribution.

## loss functions

0-1 loss: $\frac{1}{n}\sum_{i=1}^n \mathbf{1}_{y_i\hat{f}(x_i)<0}$ (non-convex)
hinge loss: $l_H(w;x_i,y_i)=\max(0,1-y_iw^Tx_i)$
logistic loss(binary): $l(z)=log(1+e^{-z})$
cross-entropy loss(multi):
$l(\hat{y},y)=-\log(P(\hat{y}=y|x)=-\log(\frac{\exp(y)}{\sum_{\hat{y}_i}\exp(\hat{y}_i)})$

## Softmax

$P(Y=y_i|x)$: softmax$_\alpha(v)_i=\frac{\exp(\alpha v_i)}{\sum_{j=1}^K\exp(\alpha v_j)}$
**Binary**: $P(\hat{Y}=y|x)=\frac{1}{1+\exp(-y\hat{f}(x))}$, $y=\pm1$
The neg-log transform = logistic loss.
MLE for softmax binary prob = min logistic loss

## Support Vector Machine (SVM)

soft svm: $\min\frac{1}{2}||w||^2+\lambda\sum_{n=1} n\zeta_i$
$s.t.\quad y_i(<w_i,x_i>+b)\ge1-\zeta_i,\quad \zeta>0\quad\forall i$
Sol: $\zeta^*=\max\{0,1-y_i(<w_i,x_i>+b)\}=l_{hinge}(y_i(<w_i,x_i>+b))$
$w^*=\text{argmin } \lambda l_{hinge}(y_i(<w_i,x_i>+b))+\frac{1}{2}||w||_2^2$
aka $l_2$-regularized risk min for hinge loss

# Kernels

efficient, implicit inner products

## Properties of kernel

$k:\mathcal{X}\times\mathcal{X}\rightarrow\mathbb{R}$, $k$ must be some inner product (symmetric, positive-semidefinite, linear). i.e. $k(\mathbf{x},\mathbf{x}')=\langle\varphi(\mathbf{x}),\varphi(\mathbf{x}')\rangle_{\mathcal{V}}\overset{Eucl.}{=}\varphi(\mathbf{x})^T\varphi(\mathbf{x}')$ and $k(\mathbf{x},\mathbf{x}')=k(\mathbf{x}',\mathbf{x})$

## Important kernels

Linear: $k(x,y)=x^Ty$
Polynomial: $k(x,y)=(x^Ty+1)^d$
Gaussian: $k(x,y)=exp(-||x-y||_2^2/(2h^2))$
Laplacian: $k(x,y)=exp(-||x-y||_1/h)$

## Composition rules

Valid kernels $k_1,k_2$, also valid kernels: $k_1+k_2$; $k_1\cdot k_2$; $c\cdot k_1$, $c>0$;
$f(k_1)$ if $f$ polynomial with pos. coeffs. or exponential

## Kernelized perceptron and SVM

$\alpha^Tk_i\rightarrow w^Tx_i$,
$\alpha^TD_yKD_y\alpha\rightarrow||w||_2^2$
$k_i=[y_1k(x_i,x_1),...,y_nk(x_i,x_n)]$, $D_y=\text{diag}(y)$

Prediction: $\hat{y}=\text{sign}(\sum_{i=1}^n\alpha_iy_ik(x_i,\hat{x}))$
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)

## Kernelized linear regression (KLR)

Ansatz: $w^*=\sum_{i=1}^n\alpha_ix$
$\alpha^*=\underset{\alpha}{\text{argmin}}||\alpha^TK-y||_2^2+\lambda\alpha^TK\alpha$
$=(K+\lambda I)^{-1}y$
Prediction: $\hat{y}=\sum_{i=1}^n\alpha_ik(x_i,\hat{x})$
knn: No training needed but depends on all data. Only max. intra-cluster distance.

# Imbalance

## Cost-Sensitive Classification

Scale loss by cost: $l_{CS}(w;x,y)=c_\pm*l(w;x,y)$

## Metrics

$n=n_++n_-$, $n_+=TP+FN$, $n_-=TN+FP$
Accuracy: $\frac{TP+TN}{n}$, Precision: $\frac{TP}{TP+FP}$
Recall/TPR: $\frac{TP}{n_+}$, FPR: $\frac{FP}{n_-}$
F1 score: $\frac{2TP}{2TP+FP+FN}=\frac{2}{\frac{1}{prec}+\frac{1}{rec}}$
ROC Curve: $y=$TPR, $x=$FPR

# Multi-class

one-vs-all ($c$), one-vs-one ($\frac{c(c-1)}{2}$), encoding

## Multi-class Hinge loss

$l_{MC-H}(w^{(1)},...,w^{(c)};x,y)=$
$\max(0,1+\max_{j\in\{1,\cdots,y-1,y+1,\cdots,c\}}w^{(j)T}x-w^{(y)T}x)$

# Neural networks

Parameterize feature map with $\theta$: $\phi(x,\theta)=\varphi(\theta^Tx)=\varphi(z)$ (activation function $\varphi$)
$\Rightarrow w^*=\underset{w,\theta}{\text{argmin}}\sum_{i=1}^n l(y_i;\sum_{j=1}^m w_j\phi(x_i,\theta_j))$
$f(x;w,\theta_{1:d})=\sum_{j=1}^m w_j\varphi(\theta_j^Tx)=w^T\varphi(\Theta x)$

## Activation functions

Sigmoid: $\frac{1}{1+exp(-z)}$, $\varphi'(z)=(1-\varphi(z))\cdot\varphi(z)$
tanh: $\varphi(z)=tanh(z)=\frac{exp(z)-exp(-z)}{exp(z)+exp(-z)}$
ReLU: $\varphi(z)=\max(z,0)$ No deri at 0

## Predict: forward propagation

Yield loss value, no weight update
$v^{(l)}=\varphi(z^{(l)})$, $z^{(l)}=W^{(l)}v^{(l-1)}$, $f=W^{(L)}v^{(L-1)}$
Predict $f$ for regression, sign($f$) for class.

## Compute gradient: backpropagation

Output layer: $\delta_j=\nabla_fl$, $\nabla_{W^L}=\delta^{(L)}v^{(L-1)T}$
Hidden layer:
$\delta^{(l)}=\varphi'(z^{(l)})\odot(w^{(l+1)T}\delta^{(l+1)})$,
$\nabla_{W^L}=\delta^{(l)}v^{(l-1)T}$

## Weight Initialization

- Avoid graident vanish

2 Sol: Special activation function + constant $v^{(i)}$ magnitude across layers. (BatchNorm)
ReLu + standardized X: initializing $Z=w^Tx$. Assume $w_i$ iid $\mathcal{N}(0,\sigma^2)$
$E(z)=0$; $Var(z)=E(z^2)=d\sigma^2$ (d: no. of par)
$E[v^2]=E[\max\{0,z\}^2]=\frac{1}{2}d\sigma^2=1$(Keeping size constant). $\rightarrow\sigma^2=2/d$

## Regularization on nn

- Avoid overfit:

Adding panelty term (weight decay);
early stopping (before convergence);
Drop-out (ignore hidden unit with prob $p$, predict with all unit and weight);
Batch-normalization: scaling and shifting: 1 separate layer before/after non-lin layer

## Convolutional Neural Network

all linear operation.
no.of.para $=mc|k|^d$ $m$: filters, $c$: channels
No. of output: ignore channels: $m\times\frac{n+2p-|k|}{s}+1$
Pooling: Averaging/maximizing - reducing dimension
ResNets: Faster training + adding momentum

# Clustering

## k-mean

$\hat{R}(\mu)=\sum_{i=1}^n\min_{j\in\{1,...k\}}||x_i-\mu_j||_2^2$
non-convex, NP-hard
Algorithm (Lloyd's heuristic): Choose starting centers, assign points to closest center, update centers to mean of each cluster, repeat.

- Monotonically decreasing average squared distance, converge to local optimal
- No. of iterationa can be exponential
- k is pre-determined, not for abstract cluster shape

## k-mean++

- Start with random data point as center

$P(i_j=i)=\frac{1}{z}\underset{1\le l<j}{min}||x_i-\mu_l||_2^2$; $\mu_j\leftarrow x_{i_j}$
Lower runtime than k-mean practically. But runtime is $O(\log k)$ that of *optimal* k-means solution.
.
.

# Dimension reduction

## PCA

$D=x_{1:n}\subset\mathbb{R}^d$, $\mu=0$, $\Sigma=\frac{1}{n}\sum_{i=1}^n x_ix_i^T$
$W$: mapping $d\Rightarrow k$, $z$: $k$-dim embeddings

$(W, z_{1:n}) = \text{argmin} \sum_{i=1}^{n} ||W z_i - x_i||_2^2 = \text{argmax}(W^T \Sigma W)$,
$W=(v_1|...|v_k) \in \mathbb{R}^{d\times k}$; $z_i = W^T x_i$
$v_i$ are the eigen vectors of $\Sigma$, 1st eigen value has the largerest on-line variance and cloeset to all observations.

### SVD

$X=USV^T$: top $k$ pcs = first $k$ columns of $V$.
$\Sigma = VS^TSV^T$, both $W$ and $V$ are eigen-decomp of $\Sigma$

### Kernel PCA

$W=\sum_{j=1}^{n} \alpha_j \phi(x_j)$, $\alpha^{(1:k)} \in \mathbb{R}^n$,
Obj fun:
$\text{argmax} \sum_{i=1}^{n} (W^T x_i)^2 = \text{argmax} \frac{\alpha^T K^T K \alpha}{\alpha^T K \alpha}$
$\alpha^{(i)} = \frac{1}{\sqrt{\lambda_i}} v_i$, $K=\sum_{i=1}^{n} \lambda_i v_i v_i^T$, $\lambda_1 \geq ... \geq \lambda_d \geq 0$
New point: $\hat{z} = f(\hat{x}) = \sum_{j=1}^{n} \alpha_j^{(i)} k(\hat{x}, x_j)$

### Autoencoders

Find identity function: $x \approx f(x;\theta)$
$f(x;\theta) = f_{decode}(f_{encode}(x;\theta_{encode});\theta_{decode})$
Unsurpervised learning but can use gradient descent.

## Probability modeling

- MLE = MLS when Gaussian iid
- Regularization = Prior

### Big Picture

Principle: $P(w|x,y) = \frac{P(w)P(y|x,w)}{P(y|x)}$
Procedures:
1. estimate likelihood $P(Y|X,w)=\prod$ Gau/T/Log
2. estimate prior $P(w)=$ Gau/Lap
3. MAP: $\hat{w} = \text{argmax} P(w|Y,X)$
4. Choose hyperpara, have $P(Y|X,w)$
5. Cla/Reg: Bayesian Optimal predictor: $f(x) = \text{argmax} p(Y=y|X=x)$

### MLE: Naive Bayesian

$\theta^* = \text{argmax} \hat{P}(y_1,...,y_n|x_1,...,x_n,\theta)$
$y_i$ i.i.d $\sim \mathcal{N}(w^T x_i, \sigma^2)$,
With MLE: $w^* = \underset{w}{\text{argmin}} \sum (y_i - w^T x_i)^2$

### Bias/Variance/Noise

Prediction error = $Bias^2 + Variance + Noise$

### Logistic regression

$P(y|x,w) = (y;\sigma(w^T x)) = \frac{1}{1+exp(-y w^T x)}$
MLE: $\underset{w}{\text{argmax}} P(y_{1:n}|w, x_{1:n}) = \text{argmin} -log(P)$
$\Rightarrow w^* = \text{argmax} \text{sign}(w^T x)$
Kernalized: $w^T x \Rightarrow \alpha^T K_i$
$L = \text{argmin} \log(1+\exp(-y_i \alpha^T K_i)) + \lambda \alpha^T K \alpha$

$P(Y|X,\alpha) = \frac{1}{1+\exp(-y \sum_{j=1}^{n} \alpha_j k(x_j, x))}$

## Bayesian decision theory

- Conditional distribution over labels $P(y|x)$
- Set of actions $\mathcal{A}$
- Cost function $C: Y \times \mathcal{A} \to \mathbb{R}$

$a^* = \underset{a\in\mathcal{A}}{\text{argmin}} \mathbb{E}[C(y,a)|x]$
Calculate $\mathbb{E}$ via sum/integral.

### Classification

$C(y,a) = [y \neq a]$;
$\hat{a} = \text{argmax } sign(w^T x_i)$
asymmetric binary classification cost:
$C_+ = p(y=-1|x)c_{FP}$; $C_- = p(y=1|x)c_{FN}$
$\hat{a} = \text{argmin}\{C_+, C_-\}$

### Regression

$C(y,a) = (y-a)^2 : a = E[y|x]$
asymmetric: $C(y,a) = c_1 \max(y-a,0) + c_2 \max(a-y,0)$

## Generative modeling

$p(y)$ (prior) $\to$ $p(x|y)$(likelihood) $\to$ $p(y|x)$ $\to$ prediction

### Naive Bayes Model

assume all features distributions are independent from class.
- $p(y=y_i) = p = \frac{n_y}{n}$
- $p(x|y) = \prod_{i=1}^{d} \mathbf{N}(x_i|\mu_{y_i}, \sigma_{y_i})$
- $p(y|x) = p(y)p(x|y)$
- $y = \text{argmax} p(y|x)$

binary case: discriminant function
$f(x) = log(\frac{p}{1-p}) = \sum_{i=1}^{d} \frac{1}{\sigma_i^2}(\mu_1 - \mu_{-1})x + log(\frac{p}{1-p}) + \sum_{i=1}^{d} \frac{1}{2\sigma^2}(\mu_{-1}^2 - \mu_1^2)$
In this case the posterior prob $p = \frac{1}{1+\exp(-f(x))}$ logistic regression

### Gaussian Bayes Classifier

Accept correlation and different variance across class.
- $\hat{P}(Y=y) = \hat{p}_y = \frac{n_y}{n}$
- $\hat{P}(x|y) = \mathcal{N}(x; \hat{\mu}_y, \hat{\Sigma}_y)$
- $\hat{\mu}_y = \frac{1}{n_y} \sum_{i:y_i=y} x_i \in \mathbb{R}^d$
- $\hat{\Sigma}_y = \frac{1}{n_y} \sum_{i:y_i=y} (x_i - \hat{\mu}_y)(x_i - \hat{\mu}_y)^T \in \mathbb{R}^{d\times d}$

binary case: LDA discriminant fun
Assume: $p=0.5$; $\hat{\Sigma}_- = \hat{\Sigma}_+ = \hat{\Sigma}$
discriminant function: $f(x) = \log\frac{p}{1-p} + \frac{1}{2}[\log\frac{|\hat{\Sigma}_-|}{|\hat{\Sigma}_+|} + \left((x-\hat{\mu}_-)^T \hat{\Sigma}_-^{-1}(x-\hat{\mu}_-)\right) - \left((x-\hat{\mu}_+)^T \hat{\Sigma}_+^{-1}(x-\hat{\mu}_+)\right)]$

Predict: $y = \text{sign}(f(x)) = \text{sign}(w^T x + w_0)$
$w = \hat{\Sigma}^{-1}(\hat{\mu}_+ - \hat{\mu}_-)$;
$w_0 = \frac{1}{2}(\hat{\mu}_-^T \hat{\Sigma}^{-1} \hat{\mu}_- - \hat{\mu}_+^T \hat{\Sigma}^{-1} \hat{\mu}_+)$

### Outlier Detection

$P(x) = \sum P(y)P(X|Y) \leq \tau$

### Categorical Naive Bayes Classifier

- $P(y) = \frac{(Y=y_j)}{n}$
- $\hat{P}(X_i = c|Y=y) = \theta_{c|y}^{(i)}$

$\theta_{c|y}^{(i)} = \frac{Count(X_i=c, Y=y)}{Count(Y=y)}$
- $y^* = \underset{y}{argmax} \hat{P}(y) \prod_{i=1}^{d} P(X|Y)$

No. of para = $O(2^d)$ overfit

### Regularization

- Prior conjugate pair: estimation of parametrized $p(y)$: $p(\theta|y)$ and $p(y)$
- Generative model tend to be sensitive to outliers!

## GMM

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

### Hard-EM algorithm

Initialize parameters $\theta^{(0)}$
E-step: predict class z: $z_i^{(t)} = \underset{z}{\text{argmax}} P(z|x_i, \theta^{(t-1)})$
$= \underset{z}{\text{argmax}} P(z|\theta^{(t-1)}) P(x_i|z, \theta^{(t-1)})$;
M-step: MLE: $\theta^{(t)} = \underset{\theta}{\text{argmax}} P(D^{(t)}|\theta)$,
i.e. $\mu_j^{(t)} = \frac{1}{n_j} \sum_{i:z_i=j} x_i$
fixed cluster distribution not good for overlapping clusters

### Soft-EM algorithm

E-step: Calc p for each point and class.:
$\gamma_j^{(t)}(x_i) = P(Z|X) = \frac{w_j P(x|\sigma_j, \mu_j)}{\sum_l w_l P(x|\sigma_l, \mu_l)}$
M-step: Fit clusters to weighted data points:
$w_j^{(t)} = \frac{1}{n} \sum_{i=1}^{n} \gamma_j^{(t)}(x_i)$; $\mu_j^{(t)} = \frac{\sum_{i=1}^{n} \gamma_j^{(t)}(x_i) x_i}{\sum_{i=1}^{n} \gamma_j^{(t)}(x_i)}$
$\sigma_j^{(t)} = \frac{\sum_{i=1}^{n} \gamma_j^{(t)}(x_i)(x_i - \mu_j^{(t)})^T (x_i - \mu_j^{(t)})}{\sum_{i=1}^{n} \gamma_j^{(t)}(x_i)}$
spherical covar EM + equal weight $w_j$ = K-mean

Overfit and dgeneracy: Adding $\nu I$ term to covariance mat
EM will monotonically increase likelihood, EM does not have a step
Anomaly detection: same as generative model

### Soft-EM for semi-supervised learning

labeled $y_i$: $\gamma_j^{(t)}(x_i) = [j=y_i]$, unlabeled:
$\gamma_j^{(t)}(x_i) = P(Z=j|x_i, \mu^{(t-1)}, \Sigma^{(t-1)}, w^{(t-1)})$

### GAN

$\min_{W_G} \max_{W_D} E_{X_d} \log(D(x; W_D)) + E_{X_m} \log(1 - D(F, W_G))$
Complicated and no way to evaluate
$D^*(x) = \frac{P_{data}}{P_{data} + P_G}$
Duality gap: Upper bound for model-data divergence (most different they can be)

## Useful math

### 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. $\nabla_w \log(1+\exp(-yw^Tx)) = \frac{-yx}{1+\exp(y w^T x)}$

### Gaussian / Normal Distribution

$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$

### Multivariate Gaussian

$\Sigma$= covariance matrix, $\mu$ = mean
$f(x) = \frac{1}{2\pi\sqrt{|\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$
Empirical: $\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$ (needs centered data points)

### Positive semi-definite matrices

$M \in \mathbb{R}^{n\times n}$ is psd $\Leftrightarrow$
$\forall x \in \mathbb{R}^n : x^T M x \geq 0 \Leftrightarrow$
all eigenvalues of $M$ are positive: $\lambda_i \geq 0$