Matrix Calculus
$\nabla(x^TAx)=(A+A^T)x, 2Ax$
$\nabla(x^TAy)=A^Ty, \nabla(y^TAx)=A^Ty$
$\nabla(x^Tx)=2x, \|x\|_2^2=x^Tx$
$\|Ax\|_2^2=x^TA^TAx, \|y-Ax\|_2^2=(y-Ax)^T(y-Ax)$
$=y^Ty-y^TAx-x^TA^Ty+x^TA^TAx=$
$y^Ty-2x^TA^Ty+x^TA^TAx$

Generative Models: $P(X,Y)=P(Y)P(X|Y)$. Estimate $P(Y)$, $P(X|Y)$, use Bayes.
Discriminative: $P(X,Y)=P(X)P(Y|X)$. Estimate $P(Y|X)$, use in [illegible] plus $P(Y=1|X)$

Linear Regression: $RSS(\beta)=\frac{1}{2}\|X\beta-Y\|^2$
$\nabla_\beta RSS=\nabla(\frac{1}{2}\beta^TX^TX\beta-y^TX\beta+\frac{1}{2}y^Ty)=0$
$X'X\beta-X'y=0 \Rightarrow \hat{\beta}=(X'X)^{-1}X'y$. Normal.
$E(\hat{\beta})=\beta$. $Var(\hat{\beta})=Cov\hat{\beta}=\sigma^2(X'X)^{-1}$
Ridge Regression: $\hat{\beta}=\min_\beta(\|y-X\beta\|^2+\lambda\|\beta\|^2)$
$\hat{\beta}_{ridge}=(X'X+\lambda I)^{-1}X'y$. L2 norm. Gaussian prior
Lasso: $\hat{\beta}=\min(\|y-X\beta\|^2+\lambda\|\beta\|)$. L1 norm
Sparse coefficients (set to 0). Laplace prior $=\exp(-\lambda\|\beta\|)$

$J(\theta)=\Sigma_i(-y_i(\theta\cdot x_i))_+$
$\nabla J_i(\theta)=-y_ix_i$ if $(y_ix_i)<0$
Algorithm: $\theta=\theta+y_ix_i$ for misclassified
Kernels Replace inner prod of feature vectors $\Phi(x)$ w/ Kernel $K(x_i,x)$. Examples
Polynomial: $K_m(x,\tilde{x})=(1+x\cdot\tilde{x})^m$
RBF: $K_{rbf}(x,\tilde{x})=\exp(-\gamma\|x-\tilde{x}\|^2)$
MLE = map w/ uniform prior = $P(x_1,x_2,\ldots x_n|\theta)$.
MAP $=P(x_1,\ldots x_n|\theta)\pi(\theta)$
π = Gaussian = **Ridge**
π = Laplace $=\exp(-\lambda\|\beta\|)$ = Lasso
Spectral Theorem: Symm., real, $A\in R^{n\times n}$, n orthonormal eigvec, $(v_1,\ldots v_n)$, n eigenvalues $(\lambda_1,\ldots\lambda_n)$.
Can write $AU=U\Lambda$, $A=U\Lambda U'$
$U=[v_1\ v_2\ldots v_n]$, $\Lambda=diag[(\lambda_1,\ldots\lambda_n)]$
Gaussian MLE: $\hat{\mu}=\frac{1}{n}\Sigma_ix_i$.
$\hat{\Sigma}=\frac{1}{n}\Sigma_i(x_i-\hat{\mu})(x_i-\hat{\mu})'$. $\hat{\sigma}^2=\frac{1}{n}\Sigma_i(x_i-\mu)^2$
Subset selection. 2^p diff subsets of features. Greedy search using CV error, forward/backward stepwise

SVM hard margin: $\min_\theta\|\theta\|^2$ s.t. $y_i\theta^Tx_i\geq1$ → margin width $1/\|\theta\|$
Softmargin: $\min_\theta \|\theta\|^2+(\Sigma(1-y_i\theta^Tx_i)_+)$
Decision Theory. **Loss**: $l(\hat{y},y)$ = cost of [illegible]
Risk: Expected loss. $R(f)=E[l(f(x),y)]$
Bayes Decision Rule $f^*(x)=$ [illegible] $P(Y=1|x)>P(Y=0|x)$
Two class Risk: $R(f)=E[1[f(x)=0]P(Y=1|x)+1[f(x)=1]P(Y=0|x)]$
Regression Risk: $E[E(f(x)-Y)^2|X]$; **Optimal Risk** R^*
Excess Risk: $R(f)-R^*=E[1[f(X)\neq f^*(X)]|2P(Y=1|X)-1|]$
Bias-Variance $R(f)=E((f(X)-E(Y|X))^2)+E(E(Y|X)-Y)^2)$
Gaussian Generative $P(x|y_k)=N(\mu_k,\Sigma)\Rightarrow$
Logistic Discriminative $P(y_1|x)=(1+\exp(-\theta^Tx-\theta_0))^{-1}$
Two class case: $\theta=\frac{\mu_1-\mu_0}{\sigma^2}$, $\theta_0=\frac{\mu_0^2-\mu_1^2}{2\sigma^2}-\log(\frac{P(0)}{P(1)})$
MV Gaussian: $\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$
$X+Y\sim N(\mu_x+\mu_y,\Sigma_x+\Sigma_y)$; $Y=AX+b$, $Y\sim N(A\mu+b, A\Sigma A^T)$
Covariance Matrix: $\Sigma=E(X-\mu)(X-\mu)'$. Symm, PSD
$Ax=\lambda x$, λ = eigenval, x = eigenvec. Solve $\det(A-\lambda I)=0$ to get λ's, solve $(A-\lambda I)[\,]=0$.
For diagonal covariance, $(Cov(X_i,X_j)=0, i\neq j)$,
Superlevel sets $\Sigma_r=\{x\in R^d:(x-\mu)'\Sigma^{-1}(x-\mu)\leq r^2\}$
$\Leftrightarrow\Sigma_{i=1}^d\frac{(x_i-\mu_i)^2}{\sigma_i^2}\leq r^2$ correspond to axis-aligned ellipsoids
For non-diagonal, Σ_Y is rotated version of diagonal Σ_X: $\Sigma_Y=Q\Sigma_XQ'$. **Volume of ellipsoids** proportional to $\Pi_{i=1}^d\sigma_i^2=\sqrt{|\Sigma|}$

Logistic Regression: For Gaussian Class Conditionals: $P(X|Y=1)=N(\mu_1,\Sigma)$.
Posterior is logistic: $P(Y=1|x)=(\exp(-\theta^Tx-\theta_0)+1)^{-1}$
Because **log odds** $\log(\frac{P(Y=1|x)}{P(Y=0|x)})=\log\frac{P(x|Y=1)P(Y=1)}{P(x|Y=0)P(Y=0)}$.
$\frac{1}{2}[(x-\mu_0)'\Sigma^{-1}(x-\mu_0)-(x-\mu_1)'\Sigma^{-1}(x-\mu_1)]+\log(\frac{P(Y=1)}{P(Y=0)})$
$=\frac{1}{2}(\mu_0'\Sigma^{-1}\mu_0-\mu_1'\Sigma^{-1}\mu_1]+\log(\frac{P(Y=1)}{P(Y=0)})+(\mu_1-\mu_0)'\Sigma^{-1}x$
$=\beta_0+\beta'x$. $\beta=\Sigma^{-1}(\mu_1-\mu_0)$. $\beta_0=\frac{1}{2}(\mu_0'\Sigma^{-1}\mu_0-\mu_1'\Sigma^{-1}\mu_1)+\log(\frac{P(Y=1)}{P(Y=0)})$. $\log\frac{p}{1-p}=\log\frac{P(Y=1|x)}{P(Y=0|x)}$
Linear Discriminant Analysis: choose k-class that maximizes $\delta_k(x)$ - classes are distributed MV Gaussian w/ common Σ.
$\hat{\pi}_k=Pr(Y=k)=\frac{n_k}{n}$ $\hat{\mu}_k=E[X|Y=k]=\frac{1}{n_k}\Sigma_{i:y_i=k}x_i$.
$\hat{\Sigma}=\frac{1}{n}\Sigma_k\Sigma_{i:y_i=k}(x_i-\mu_k)(x_i-\mu_k)'=Var[X|Y=k]$
LD Function: $\delta_k(x)=\hat{\mu}_k'\hat{\Sigma}^{-1}x-\frac{1}{2}\hat{\mu}_k'\hat{\Sigma}^{-1}\hat{\mu}_k+\log\hat{\pi}_k$
KKT: ① **Primal feasibility**: $f_i(x)\leq0$ feasible
② **Dual feasibility**: $\lambda\geq0$ feasible.
③ **Complementary Slackness**: $\lambda_if_i(x)=0$. This means: $\lambda_i\geq0$, $f_i(x)=0$ or $f_i(x)\leq0$, $\lambda_i=0$.
④ **Stationarity**: $\nabla f_0(x)+\Sigma_i\lambda_i\nabla_i f_i(x)=0$

MLE for Logistic: $l(\beta)=\log P(y_1,\ldots y_n|x_1,\ldots x_n;\beta)$
$=\Sigma_iy_i\log\mu_i(\beta)+(1-y_i)(1-\mu_i(\beta))$, $\mu_i(\beta)=P(Y=1|X=x,\beta)$
$=(1+\exp(-\beta'x_i))$. $\nabla_\beta\mu_i(\beta)=\mu_i(\beta)(1-\mu_i(\beta))x_i$
First Deriv $\nabla_\beta l(\beta)=\Sigma_i((\frac{y_i}{\mu_i(\beta)}-\frac{1-y_i}{(1-\mu_i(\beta))})\nabla_\beta(\mu_i(\beta))=$
$\Sigma_i(y_i-\mu_i(\beta))x_i$. **Second**: $\nabla_\beta^2l(\beta)=\Sigma_i-\mu_i(\beta)(1-\mu_i(\beta))x_ix_i'$
Set $\nabla_\beta l(\beta)=0\Rightarrow\Sigma_iy_ix_i=\Sigma_i\mu_i(\beta)x_i$. No closed form.
Gradient ascent $\beta^{t+1}=\beta^t+\eta(\nabla_\beta l(\beta^{(t)}))=$ $O(np)$ per iter.
$\beta^t+\eta\Sigma_i(y_i-\mu_i(\beta^t))$.
Stochastic Gradient: Use single value, $O(p)$/iter
Linear Reg: $\nabla_\beta RSS(\beta)=X'X\beta-X'y=\Sigma_ix_i(x_i'\beta-y_i)\Rightarrow$
$\beta^{(t+1)}=\beta^t+\eta x_{i,t}(y_{i,t}-x_{i,t}'\beta^t)$.
Logistic Reg. $\beta^{t+1}=\beta^t+\eta(y_{i,t}-\mu_{i,t}(\beta^t))x_{i,t}$.
Lagrange Multipliers: min/max $f_0(x)$ s.t. $f_1(x)\geq0$
Replace w/: $L(x,\lambda)=f_0(x)+\lambda f_1(x)$. Take PD's w.r.t. x_i's, λ, set to 0, solve.
Optimization: **primal** $=p^*=\min_x\max_{\lambda\geq0}L(x,\lambda)$
$g(\lambda)=\min_xL(x,\lambda)$. **dual** $=d^*=\max_{\lambda\geq0}\min_xL(x,\lambda)$
SVM hard margin dual: $\max_\alpha\Sigma_i\alpha_i-\frac{1}{2}\Sigma_{i,j}\alpha_i\alpha_jy_iy_jx_i^Tx_j$: $0\leq\alpha_i$
SVM soft margin dual: $\max_\alpha\Sigma_i\alpha_i-\frac{1}{2}\Sigma_{i,j}\alpha_i\alpha_jy_iy_jx_i^Tx_j$: $0\leq\alpha_i\leq\frac{C}{n}$
corresponding primal: $\min_{\theta,\xi}\frac{1}{2}\|\theta\|_2^2+\frac{C}{n}\Sigma_i\xi_i$: $1-y_i\theta^Tx_i-\xi_i\leq0$, $\xi\geq0$

Newton's Method: iteratively reweighted least squares.
$\nabla f(x)=0\Rightarrow x_{t+1}=x_t-[\nabla^2f(x_t)]^{-1}\nabla f$
Newton Raphson for LogR
$l(\beta)=\Sigma_iy_i\log\mu_i(\beta)+(1-y_i)\log(1-\mu_i)$
$\nabla_\beta l(\beta)=\Sigma_i(y_i-\mu_i(\beta))x_i=X'(y-\mu)$
$\nabla_\beta^2l(\beta)=\Sigma_i-\mu_i(\beta)(1-\mu_i(\beta))x_ix_i'=$
$-X diag(\mu(1-\mu))X$.
$\beta^{t+1}=\beta^t-[\nabla^2l(\beta^t)]^{-1}\nabla l(\beta^t)=$
$\beta^t+[X'diag(\mu(1-\mu))X]^{-1}X'(y$
Using **SVM kernel**, classify point x by $\theta\cdot\Phi(x)=$
$\Sigma_j\alpha_jy_j(\Phi(x_j)\cdot\Phi(x))=\Sigma_j\alpha_jy_jK$
where $\alpha_j\neq0$ only for support
Quad. Kernel features:
$[1,x_1^2,x_2^2,\sqrt{2}x_1,\sqrt{2}x_2,\sqrt{2}$

kNN algorithm: Given x_q, locate k-nn by distance function, take vote among k.
Not good in high-D; if $D>n$:
① Get more data, increase n
② Reduce features, decrease D
③ Use better distance metric
④ set k by CV.
KNN optimality: $1.\ n\to\infty$, $k\to\infty$, $k/n\to0$, $E_{knn}=E^*$
Mahalanobis Distance: $D_M(x,y|\Sigma)=\sqrt{(x-y)^T\Sigma^{-1}(x-y)}$.
Transformation Invariance: Augment dataset w/ slight transformations of training data

k-d tree: Binary tree for organizing points in k-dim space. Each node associated w/ axis-aligned hyperplane splitting into two subtrees (choose dim w/ highest variance first). Position chosen @ median to balance tree. **Best bin first** uses priority queue for branches. **Locality sensitive hashing** uses hash to map similar points to same bin. Project data r^Tx with bits.

Decision Tree Alg. **GrowTree(S)**:
```
-if (y=0 for all <x,y>∈S) return newLeaf(
-if (y=1 for all <x,y>∈S) return newLeaf(
-choose best attribute x_j
  -S_0 = all <x,y>∈S w/ x_j=0
  -S_1 = all <x,y>∈S w/ x_j=1
-return new node(x_j, GrowTree(S_0), Gr
```
ChooseBestAttribute(S): 1-step lookahead
```
-Choose j to minimize J_j:
  -S_0 = all <x,y> w/ x_j=0; S_1 = all <x,y> w/ x_j=1
  -y_0 = most common y in S_0; y_1 = most common y in
  -J_0 = #<x,y> in S_0 y≠y_0; J_1 = #<x,y> in S_1, y≠
  -J_j = J_0 + J_1
-return j
```

Surprise: $S(V=v) = -\log(V=v)$
$P(v)=0$: ∞ surprise; $P(v)=1$: 0 surprise
Entropy: average surprise
$H = \sum_i -p_i \log p_i$. For group w/ all one class, entropy is 0.
For group w/ 50% split, entropy is
$H = (-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}) = \underline{1}$.
Information Gain: entropy(parent) - [average entropy(children)].
Want to make split that maximizes info gain.
Choose Best Attr: pick based on IG.
Classification tree: $X \to \{0,1\}$
Regression Tree: $X \to y \in R$
Instead of minimizing entropy, minimize sum of variance after split: $\sum_{x_i:R_1}(y_i-\hat{y}_{R_1})^2 + \sum_{x_i:R_2}(y_i-\hat{y}_{R_2})^2$

To prevent overfitting:
1. Impose stopping criteria:
- Max tree depth
- Min # points @ node
- Tree complexity penalty
- Validation error monitoring
Stop splitting earlier & make leaf node:
- avg/majority of y's @ leaf
2. Reduced Error Pruning:
Split into train & validation data.
- Do until further pruning harmful:
1. Evaluate impact on validation set at pruning node & those below
2. Greedily remove node w/ most improvement

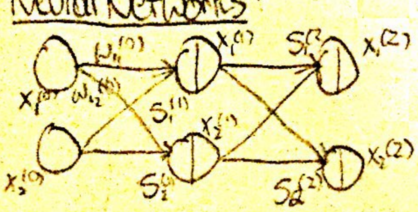
Ensemble Methods
Combine several models
- Bagging Averaging, Random Forest, Boosting
Bagging: Generate bootstrap replicates of training set by sampling w/ replacement. Learn one model on each. Combine by uniform voting.
Random Forest: Bagging on dataset, also random subset features for each tree

Boosting: Build strong learner as combo of weak learners.
$L(x) = a_1 h_1(x) + \dots + a_n h_n(x)$
ADABOOST: (W, D, K)

```
1. $d^{(0)} = [\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N}]$ // uniform importance weights on examples
2. for $k = 1, 2, \dots K$ do
    3. $f^{(k)} = W(D, d^{(k-1)})$ // train kth classifier on weighted data
    4. $\hat{y}_n = f^{(k)}(x_n)\ \forall n$ // make predictions
    5. $\hat{e}^{(k)} = \sum_n d_n^{(k-1)}[y_n \neq \hat{y}_n]$ // compute weighted training error
    6. $\alpha^{(k)} = \frac{1}{2}\log((1-\hat{e}^{(k)})/\hat{e}^{(k)})$ // compute adaptive parameter
    7. $d_n^{(k)} = \frac{1}{Z} d_n^{(k-1)} \exp[-\alpha^{(k)} y_n \hat{y}_n],\ \forall n$ // reweight examples & normalize
8. return $f(\hat{x}) = \mathrm{sgn}[\sum_k \alpha^{(k)} f^{(k)}(\hat{x})]$ // return reweighted classifier
```

ADABoost: Adapt weights for each tree. As error grows, "importance" of tree (adaptive weight) goes down. As error decreases, tree is more important. For each example, boost weight if incorrectly predicted, else decrease.
$\hat{e}$ = sum of misclassifications. $\alpha = \frac{1}{2}\ln(\frac{1-\hat{e}}{\hat{e}})$.
$D_n^{(k)} = \frac{1}{Z} \times D_n^{(k-1)} \times \begin{cases} e^{-\alpha_t} & \hat{y}_n = y_n \\ e^{\alpha_t} & \hat{y}_n \neq y_n \end{cases}$. Z is normalization factor

Neural Networks:



$g(s) = \frac{1}{1+e^{-s}}$

$s_i^{(k)} = \sum_j x_j^{(k-1)} w_{ij}$
$x_i^{(k)} = g(s_i^{(k)})$
$J(w) = \frac{1}{2}\sum_i (x_i^{(2)} - y_i)^2$

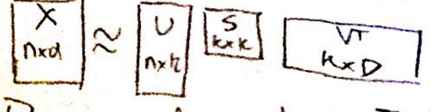
$\frac{\partial J(w)}{\partial w_{ij}^{k}} = \frac{\partial J(w)}{\partial x_j^{(k+1)}} \frac{\partial x_j^{(k+1)}}{\partial s_j^{(k+1)}} \frac{\partial s_j^{(k+1)}}{\partial w_{ij}^{k}}$
$\frac{\partial s_j^{(k+1)}}{\partial w_{ij}^{k}} = x_i^{k}$; $\frac{\partial x_j^{(k+1)}}{\partial s_j^{(k+1)}} = g(s_j^{(k+1)})(1-g(s_j^{(k+1)}))$
$\frac{\partial J(w)}{\partial x_j^{(k+1)}} = (x_j - y_j)$ at output.
$w_{ij} = w_{ij} - \eta \frac{\partial J(w)}{\partial w_{ij}}$

$\frac{\partial J(w)}{\partial x_j^{(k)}} = \sum_i \left(\frac{\partial J(w)}{\partial x_i^{(k+1)}} \frac{\partial x_i^{(k+1)}}{\partial s_i^{(k+1)}}\right.$
$\frac{\partial J(w)}{\partial w_{ij}} = \delta_j x_i$ where
$\delta_j = \frac{\partial J(w)}{\partial x_j}\frac{\partial x_j}{\partial s_j} = \begin{cases}(y_j - x_j)g(s_j)(1-g(s_j)) & \text{@ output} \\ (\sum_i \delta_i w_{ji})g(s_j)(1-g(s_j)) & \text{@ inner}\end{cases}$
where i is nodes connected in next layer

Clustering: Hierarchical: Produce tree/dendrogram
Agglomerative: bottom-up (most used). Start with points, @ each step merge closest two
Divisive: top-down.
Need to define distance function
Flat partitioning: Partition data into k groups, iteratively reallocate observation to closest group to minimize loss.
K-means: Initialize cluster centroids $\mu_1, \dots \mu_k$ randomly.
Repeat until convergence: ① Assign every point to closest cluster k: $C_i = \arg\min_k \|x_i - \mu_k\|^2$. ② For every cluster k, recompute its mean $\mu_k = (\sum_{C_i=k} x_i)/(\sum_{C_i=k} 1)$
Repeat to minimize objective function: $J(C,\mu) = \sum_i \|x_i - \mu_{C_i}\|^2$.
K-means will converge to local minima. Use random restarts/better init
K-means++: Choose one center uniformly @ random from data, compute $D(x)$ = distance to nearest center. Choose one new data point as center using prob. distribution on $D(x)^2$. Repeat to get k centers, then do k-means.
k-means works well if clusters spherical/well separated/similar vol./#pts.
Gaussian mixture model clusters: $P(x) = \sum_i P(c_i)P(x|c_i)$. Gaussian prior

Meanshift:

PCA: Given design matrix X
1. Subtract mean from each point
2. (Sometimes) scale by SD
3. Compute $S = X^TX$ = cov mat = Σ
4. Compute eigenvec. $S = VDV^T$
SVD: Decompose $X = USV^T$
$X^TX = VSSV^T = VDV^T$. $XX^T = USSU^T =$



Dimensionality reduction: Taking k eigenvectors amounts to reducing dimensions of X into k dim w/ highest variance (project data points onto vector). Approximation to
$X_i = U_{i,1}S_{11}v_1 + U_{i,2}S_{22}v_2 + \dots$

XX^T, X^TX share same eigenvalues
To SVD X: Calculate $\det(XX^T - \lambda I)$ to get λ, then $(XX^T - \lambda I)\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$ for U vec, X^TX for V vec