SUPERVISED LEARNING

+ LINEAR REGRESSION
+ LINEAR CLASSIFICATION (PERCEPTRON)
+ LOGISTIC REGRESSION
+ NEAREST NEIGHBOUR
+ DECISION TREES
+ MULTICLASS SOFTMAX
+ NAIVE BAYES
+ GENERATIVE MULTICLASS GAUSSIAN.
+ SUPPORT VECTOR MACHINES

LINEAR REGRESSION

+ INPUT DATA: $(x^1, y^1), (x^2, y^2), \ldots, (x^N, y^N)$
  where $x^i = (x_1, x_2, \ldots, x_d)$. and y^i = dependent variable
  $x^i \in \mathbb{R}^d$ and $y^i \in \mathbb{R}$ [NOTE, CONTINUOUS OUTPUT]

+ OUTPUT MECHANISM: + In a linear regression, we have some LOSS FUNCTION e.g. $L_2(y,\hat{y}) = (y-\hat{y})^2$, $L_1(y,\hat{y}) = |y-\hat{y}|$ and a PREDICTION MECHANISM e.g. $f_w(x) = w_0 + w_1 x_1 + \cdots + w_d x_d$ where $f_w(x) = \hat{y}$.
  + We can use BASIS FXNS $\phi(x_1), \ldots, \phi(x_d)$ b/c linear in params not inputs.
  + Need to MINIMIZE THE LOSS. which gives us $\hat{y}$ closest to y as possible. Can do this analytically. Goal is to find w^* minimizing L_1/L_2. LOSS FXN IN VARIABLE OF w. This is "training" linreg.
  + When done, w's optimized. Can output $\hat{y}$ closest to y. Model of data.

+ KEY MATHS/ALGOS:
  + Analytical derivation of w^* minimizing loss:

Define $\hat{y} = Xw$ $(n \times d \cdot d \times 1 = n \times 1$ preds).
Then, the L_2 loss is: $L_2(w) = \|y - Xw\|^2$
Tre, $L_2(w) = (y - Xw)^T(y - Xw) = (y^T - (Xw)^T)(y - Xw) = (y^T - w^T X^T)(y - Xw)$
Tha, $(y^T - w^T X^T)(y - Xw) = y^T y - y^T Xw - w^T X^T y + w^T X^T Xw$ } equiv
$= y^T y - w^T X^T y - w^T X^T y + w^T X^T Xw$
$= y^T y - 2w^T X^T y + w^T X^T Xw$.

Then, $\partial_w L(w) = -2X^T y + 2X^T Xw \rightarrow$ set to 0: $0 = 2X^T Xw - 2X^T y$
$\rightarrow w = (X^T X)^{-1} X^T y$. NOTE. IF features NOT LIN DEP, $(X^T X)^{-1}$ exists.
(use solvers: conj. grad, Cholesky decomp)

With REGULARIZATION: $w = (X^T X + \lambda I)^{-1} X^T y$.

+ KEY INFORMATION.

+ KEY INFO.
  + Requires initially designing good feats, outside of scope of algo. Can be difficult!
  + L_2 LOSS: $L_2(y,\hat{y}) = (y-\hat{y})^2$
    Optimal prediction is the CONDITIONAL MEAN $E[y|x]$
    Assumes GAUSSIAN NOISE
  + L_1 LOSS: $L_1(y,\hat{y}) = |y-\hat{y}|$
    Optimal prediction is the CONDITIONAL MEDIAN.
    Assumes LAPLACE NOISE
  + L_1 LOSS VS L_2 LOSS

L_2 LOSS	L_1 LOSS
+ Not resistant to data outliers (low robustness, punishes mistakes)	+ Is robust
+ Stable soln	+ Unstable soln (for a small horizontal mvmt of data pt, reg. line may jump a LOT)
+ Always one soln	+ Multiple solns possible - (euclid vs. taxicabs)

  + L_2 REG: $R_2(w) = \frac{\lambda}{2}\|w\|^2 = \frac{\lambda}{2}w^Tw = \frac{\lambda}{2}\sum w_j^2 \sim$ GAUSSIAN PRIOR
    + Bias NOT regularized
    + Find good $\lambda \rightarrow$ VALIDATION/CV
  + L_1 REG: $R_1(w) = \lambda\|w\| = \lambda\sum|w_j| \sim$ LAPLACE PRIOR.
  + L_1 REG VS L_2 REG

L_2 REG	L_1 REG
+ Analytic soln: comp. efficient	+ Comp. inefficient bc non-sparsity
+ Non-sparse outs	+ Sparse outs - "only few matrix entries non ZERO"
+ No feat. sel.	+ Feat sel. - b/c $L_1 \rightarrow$ sparse coefficients

+ PROBABILISTIC VIEW: (L_2)
  Assume $p(y^i|x^i) = w^Tx^i + \epsilon_i$, $\epsilon_i \sim \mathcal{N}(0,\sigma^2)$
  $$p(y|x) = \mathcal{N}(w^Tx, \sigma^2) = \frac{\exp\left(-\frac{\|y-w^Tx\|^2}{2\sigma^2}\right)}{\sqrt{2\pi\sigma^2}}$$
  We know w parameterizes a distribution. Which distro to pick? Maximize LIKELIHOOD that we get y's given x's!
  $$p(y|x) = p(y^1|x^1)p(y^2|x^2)\cdots p(y^N|x^N)$$
  $$\log(p(y|x)) = \log(p(y^1|x^1)p(y^2|x^2)\cdots p(y^N|x^N))$$
  $$= \sum_N \log p(y^i|x^i) = \sum_N \log\left(\frac{\exp\left(-\frac{\|y-w^Tx\|^2}{2\sigma^2}\right)}{\sqrt{2\pi\sigma}}\right) =$$
  $$= \sum_N \frac{-\|y-w^Tx\|^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma)$$
+ MAP estimation:
  $w_{MAP} = \arg\max P(w|y,X) = p(y|w,X)p(w)$.

CLASSIFICATION

LINEARLY SEPARABLE: Data of two classes COMPLETELY SEPARABLE BY SINGLE LINE.

+ NON-PERFECT SEPARATION:
  + Simple model
  + Noise
  + Errors in data target (mislabels)
  + Simple features
  + Different feature parameterization.
+ ZERO-ONE LOSS
  + 1 if wrong, 0 if right.
  + SURROGATE LOSS: PIECEWISE SMOOTH + CONVEX.
    + $\ell(y,\hat{y}) \le \hat{\ell}(y,\hat{y}) \rightarrow$ minimizing $\hat{\ell}$ minimizes ℓ.
+ METRICS:
  + PRECISION, RECALL, ROC (FP/TP)

LOGISTIC REGRESSION

+ INPUT: (x^i, y^i), $y^i \in \{0,1\}$.
+ We classify with a linear decision boundary: $\sigma(z) = \frac{1}{1+e^{-z}}$, $z = w^T x$.
  Therefore, if $\sigma(z) \ge 0.5 \rightarrow \hat{y}=1$, $\sigma(z)<0.5 \rightarrow \hat{y}=0$.
+ To learn the weights, we have a probabilistic model,
+ Need to use MAX LIKELIHOOD.

Want to maximize the $p(y|x;w)$. (likelihood). $\max_w L(w) = \max_w \prod_{i=1}^{N} p(y^i|x^i;w)$

We know:

$$p(y|x;w) = p(y^1,\dots,y^N|x^1,\dots,x^N;w)$$
$$= p(y^1|x^1;w)\,p(y^2|x^2;w)\cdots p(y^N|x^N;w). \quad (\text{IID})$$
$$= \left[p(y=1|x^1;w)^{y^1} p(y=0|x^1;w)^{(1-y^1)}\right]\cdots\left[p(y=1|x^N;w)^{y^N} p(y=0|x^N;w)^{(1-y^N)}\right]$$

1) Maximize log-likelihood (neg $\rightarrow$ minimize)

$$L(w) = \prod_{i=1}^{N} p(y^i|x^i;w) \quad \log L(w) = \log\left(\prod_{i=1}^{N} p(y^i|x^i;w)\right) = \sum_{i=1}^{N} \log\left(p(y^i|x^i;w)\right).$$

Then, $\log L(w) = \sum_{i=1}^{N} \log\left(p(y^i|x^i;w)\right) = \sum_{i=1}^{N} \log\left[p(y=1|x^i;w)^{y^i} p(y=0|x^i;w)^{1-y^i}\right]$

$$-\log L(w) = -\sum_{i=1}^{N} \log\left[p(y=1|x^i;w)^{y^i}\left(1-p(y=1|x^i;w)\right)^{1-y^i}\right]$$
$$= -\left(\sum_{i=1}^{N}\left[y^i \log p(y=1|x^i;w) + (1-y^i)\log p(y=0|x^i;w)\right]\right)$$
$$= -\sum_{i=1}^{N} y^i \log p(y=1|x^i;w) - \sum_{i=1}^{N}(1-y^i)\log p(y=0|x^i;w)$$

$$p(y=1|x^i;w) = \left(1+\exp(-w^T x)\right)^{-1} \qquad p(y=0|x^i;w) = 1 - p(y=1|x^i;w)$$

$$-\log L(w) = -\sum_{i=1}^{N} y^i \log\left((1+\exp(-z))^{-1}\right) - \sum_{i=1}^{N} (1-y^i)\log\left(\exp(-z)(1+\exp(-z))^{-1}\right)$$

$$= \sum_{i=1}^{N} y^i \log(1+\exp(-z)) - \sum_{i=1}^{N} (1-y^i)\left[\log(\exp(-z)) + \log\left((1+\exp(-z))^{-1}\right)\right]$$

$$= \sum_{i=1}^{N} y^i \log(1+\exp(-z)) - \sum_{i=1}^{N} (1-y^i)(-z) - (1-y^i)\log(1+\exp(-z))$$

$$= \sum_{i=1}^{N} y^i \log(1+\exp(-z)) + \sum_{i=1}^{N} (1-y^i) z + \sum_{i=1}^{N} (1-y^i)\log(1+\exp(-z))$$

$$= \sum_{i=1}^{N} y^i \log(1+\exp(-z)) + \sum_{i=1}^{N} (1-y^i) z + \sum_{i=1}^{N} \log(1+\exp(-z)) - \sum_{i=1}^{N} y^i \log(1+\exp(-z))$$

$$= \sum_{i=1}^{N} (1-y^i) z^i + \sum_{i=1}^{N} \log(1+\exp(-z)).$$

Now, taking the derivative:

$$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_j} = \sum_{i=1}^{N} (1-y^i) x_j^i - \sum_{i=1}^{N} \frac{1}{1+\exp(-z)} \cdot \exp(-z) \cdot x_j =$$

$$= \sum_{i=1}^{N} (1-y^i) x_j - \frac{x_j \exp(-z)}{1+\exp(-z)} = \sum_{i=1}^{N} x_j \left(1 - y^i - \frac{\exp(-z)}{1+\exp(-z)}\right) = \sum_{i=1}^{N} x_j \left(\frac{1}{1+\exp(-z)} - y^i\right)$$

Thus,

$$\frac{\partial L}{\partial w_j} = \sum_{i=1}^{N} x_j \left(p(y=1 \mid x^i; w) - y^i\right)$$

+ LOG-LOSS is more robust to outliers than L_2!
+ MAP estimation: regularization to avoid overfitting:

  MAP: $\max_w \log\left[p(w)\prod_{i=1}^{N} p(y^i \mid x^i; w)\right]$

  $p(w) \sim N(0, \alpha^{-1} I)$ → equiv to $\frac{\lambda}{2}\|w\|^2 = \frac{\lambda}{2}\sum w_j^2$. L_2 regularization

+ New gradient: $w_j^{(t+1)} \leftarrow w_j^{(t)} - \lambda\left(\frac{\partial L(w)}{\partial w_j} + \alpha w_j\right) = w_j^{(t)} - \lambda\frac{\partial L(w)}{\partial w_j} - \lambda\alpha w_j$

+ TUNING HYPERPARAMS:
  + Training and validation set.
  + CAN overfit on validation set
  + LEAVE-p-OUT: $\binom{n}{p}$.
  + K-FOLD: Divide train set into k equal subsamples
    k iterations: k-1 training, k^{th} vs validation.
    k results can be averaged for single hyperparam estimate

+ PROS:
  - class pred prob. view
  - quick train
  - convex loss (log loss)
  - fast classify
  - good acc.
  - resistant to overfit ($N >= 10 \cdot d$)
  - model coeff = feature importance

+ CONS:
  - Linear decision boundary
  - Simple model, assumptions of cond. probs

ST NEIGHBOUR

NONPARAMETRIC: while log reg is a probabilistic classifier, this is a EUCLIDEAN classifier.
GROWING # OF PARAMS
NONLINEAR DECISION BOUNDARIES

+ ALGORITHM:
  1. Take point x^i
  2. Find k points $x^{*1} \dots x^{*k}$ s.t. each is minimal distance to x^i.
  3. Vote on class. $y^{*1} \dots y^{*k}$. Assign $y^i = \text{mode}(y^{*1} \dots y^{*k})$

+ CHOOSING k:
  Generally $k < \text{sqrt}(n)$. Choose k via cross validation.

+ kNN: ISSUES + REMEDIES
  + Attributes w/ larger ranges get weighed more
    ↳ NORMALIZE
  + Irrelevant, correlated attributes add noise
    ↳ ELIM. ATTRIBUTES
  + Nonmetric attributes
    ↳ HAMMING DIST
  + High Dim Data
    ↳ DIM. REDUCTION
  + Expensive at Test Time
    ↳ USE DIM. SUBSETS
    ↳ PRESORT TRAINING EXAMPLES
    ↳ APPROX DIST ONLY
    ↳ REMOVE REDUNDANT DATA

DECISION TREES

+ Each node is a feature, "SPLIT" the universe of values of these features to each other features. Classification → tree descent.
  LEAVES ARE CLASS LABELS.

+ ENTROPY EQUATIONS:

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x).$$

$$H(X,Y) = -\sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 p(x,y).$$

$$H(Y|X=x) = \sum_{y \in Y} p(y|x) \log_2 p(y|x)$$

$$H(Y|X) = \sum_{x \in X} p(x) H(Y|X=x) = -\sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 p(y|x)$$

$$IG(Y|X) = H(Y) - H(Y|X)$$

+ IF X, Y INDEP:
  $H(X|Y) = H(X)$.

+ CHAIN RULE:
  $H(X,Y) = H(X|Y) + H(Y)$
  $= H(Y|X) + H(X)$

+ DECISION TREE ALGORITHM

1) Calculate $H(Y)$ where Y = class targets.
2) Calculate conditional entropy of each feature:
   $H(Y|X_1)$, $H(Y|X_2)$, ..., $H(Y|X_d)$.
3) Calculate information gain for each feature:
   $IG(Y, X_1)$, $IG(Y, X_2)$, ..., $IG(Y, X_d)$.
4) Choose attribute w/ LARGEST IG as DECISION NODE. Assume here it's $X_i = a$.

Recurse on that node:

4.1) Choose $X_i = a$, a particular branch
4.2) Calculate $H(Y|X_i = a)$.
4.3) Calculate $H(Y|X_1, X_i = a)$, ..., $H(Y|X_{i-1}, X_i = a)$, $H(Y|X_{i+1}, X_i = a)$, ..., $H(Y|X_d, X_i = a)$
4.4) Calculate info gain:
   $IG(Y, X_1, X_i = a)$ ... $IG(Y, X_d, X_i = d)$.
4.5) Choose attribute w/ LARGEST IG as DECISION NODE.
4.6) Choose $X_i = b$, the second branch
   ⋮

Recurse on the next layer, repeat above ...

+ GOOD TREE:
  + "Goldilocks size" → too small won't handle subtle distinctions in data
    → too big will be efficient and overfit.
  + Find smallest tree fitting hypothesis
  + INDUCTIVE BIAS: small trees, informative nodes near root

+ ISSUES:
  + less data at lower levels (exponentially)
  + ↑ size tree = overfits
  + Above algo greedy → NOT OPTIMAL b/c NP-Hard Problem.
  + Can REGULARIZE = prune, min gain for split, min node size

+ PROS:
  + Good when fxns are boolean, depends critically on few attributes

+ CONS:
  + Bad on parity and most functions
  + Not good on continuous attributes

CLASS CLASSIFICATION

VR: $R_1 \cap R_2$ region ambiguity, OVO: no-R region ambiguity

K-CLASS DISCRIMINANT:

$$y_k(x) = w_k^T x + w_{k,0} \quad (k \text{ unique } W\text{s})$$

Assign if $y_k(x) \geq y_j(x)$

$$(w_k - w_j)^T x + (w_{k0} - w_{j0}) = 0 \leftarrow \text{line}$$

+ Convex decision regions
+ SOFTMAX:

$$p(C_k|x) = y_k(x) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

+ ONE-HOT ENCODING: $t = [0,1,0,0]^T \rightarrow k=2. \quad = [t_1, t_2, \ldots, t_K]$
+ MULTICLASS LOGISTIC REGRESSION
  + We want to MAXIMIZE THE LIKELIHOOD of the correct class classification.
  + Let T be a matrix of N dims, row i is the t^i
  + Then: WT Maximize:

$$p(T|X; w_1, \ldots, w_K) = \prod_{i=1}^{N} \prod_{k=1}^{K} p(C_k|x^i)^{t_k^i} = \prod_{i=1}^{N} \prod_{k=1}^{K} y_k^i(x^i)^{t_k^i} \quad \text{(IID)}.$$

Recall neg. log likelihood:

$$-\log p(T|X; w_1, \ldots, w_K) = -\log \left[\prod_{i}^{N} \prod_{k}^{K} y_k^i(x^i)^{t_k^i} \right]$$

$$L(w_1, \ldots, w_K) = -\sum_{i}^{N} \sum_{k}^{K} t_k^i \log(y_k^i(x^i)) \leftarrow \text{CROSS-ENTROPY ERROR}$$

+ Obtaining OPTIMIZED WEIGHTS
  + Do GRAD DESCENT. Need derivatives:

$$\frac{\partial y_j^i}{\partial z_k^i} = \frac{\partial}{\partial z_k^i}\left(\frac{\exp(z_j^i)}{\sum_l \exp(z_l^i)}\right) = \underbrace{\delta(k,j)}_{\substack{1 \text{ if} \\ k=j, \\ 0 \text{ else}}} y_j^i - y_j^i y_k^i$$

$$\frac{\partial}{\partial z_k^i}\left(\frac{\exp(z_j^i)}{\sum_l \exp(z_l^i)}\right) = \frac{\partial L}{\partial z_k^i} = y_k^i - t_k^i$$

$$\frac{\partial L}{\partial w_{k,g}} = \sum_{i=1}^{N} \frac{\partial L}{\partial z_k^i} \cdot \frac{\partial z_k^i}{\partial w_{k,g}} = \sum_{i=1}^{N} (y_k^i - t_k^i) x_g^i$$

Note:

$$\frac{\partial L}{\partial w_{k,g}} = \sum_{i=1}^{N} \frac{\partial L}{\partial z_k^i} \cdot \frac{\partial z_k^i}{\partial w_{k,g}} = \sum_{i=1}^{N} \left(\sum_{j=1}^{K} \frac{\partial L}{\partial y_j^i} \cdot \frac{\partial y_j^i}{\partial z_k^i} \right) \cdot \frac{\partial z_k^i}{\partial w_{k,g}}$$

+ K-NNs, DCs can directly handle multiclass!

GENERATIVE MODELS

+ Model $p(x,y)$ given $p(y)$, and compute $p(x|y)$. KEY IDEA: The probability of class y! "HOW DOES DATA LOOK LIKE FOR A CLASS?"
+ DISC: How do I separate the classes?
+ GEN: What does each class "look" like?

+ $\text{post} = \frac{\text{class likelihood} \times \text{prior}}{\text{evidence}}$

+ GOAL: $p(C|x) = \frac{p(x|C)\,p(C)}{p(x)}$
+ Simple Bernoulli Binary class situation:
  + $p(x|C) = \text{Ber}(\theta_c) = \theta_c^{x}(1-\theta_c)^{1-x}$
  + FIT DISTRIBUTION TO DATA: Let D_c be data of class C.
  + Want to MAXIMIZE $p(D_c|C)$ → prob. of data of class C, given class C.
  + $p(D_c|C) = \prod_{i=1}^{N} p(x^i|C) = \prod_{i=1}^{N} \theta_c^{x^i}(1-\theta_c)^{1-x^i} = \theta_c^{N_c}(1-\theta_c)^{N-N_c}$.
    (note $x^i \in \{0,1\}$ here)
  + MINIMIZE $-\log$: $-\log p(D_c|C) = -\log(\theta_c^{N_c}(1-\theta_c)^{N-N_c})$
    $= -N_c \log(\theta_c) - (N-N_c)\log(1-\theta_c)$
  + DERIVATIVE FOR SGD:
    $$\frac{\partial \ell_{\text{logloss}}}{\partial \theta_c} = -\frac{N_c}{\theta_c} + \frac{N-N_c}{1-\theta_c} = 0 \rightarrow \boxed{\theta_c = \frac{N_c}{N}}$$
+ What if $N_c = 0$? (Rare Word Case)
  + BETA-BINOMIAL: Need a PRIOR over θ.
    $\text{Beta}(\theta|a,b) = \theta^{a-1}(1-\theta)^{b-1}$
  + $p(\theta_c|D_c) \propto p(D_c|\theta_c)\,p(\theta_c) \rightarrow \theta_c^{N_c}(1-\theta)^{N-N_c} \cdot \theta_c^{a-1}(1-\theta_c)^{b-1}$
    $= \theta^{N_c+a-1}(1-\theta_c)^{N-N_c+b-1}$
  + MAP estimation: $\theta_{c,\text{map}}$: $\frac{N_c+a-1}{N+a+b-2}$

+ NAIVE BAYES
  + Before: IID assumption: $p(y|x;w) = p(y^1|x^1;w)\ldots p(y^d|x^d;w)$
  + Now: $p(x,y;\theta_c) = p(x_1|x_2\ldots x_d;\theta;\theta_c)\,p(x_2|x_3\ldots x_d,y;\theta_c)\ldots p(x_{d-1}|x_d,y;\theta_c)\,p(y)$
  + NEED NAIVE BAYES ASSUMPTION: Dimensions d are INDEPENDENT given class y.
    $$p(x|y=c;\theta_c) = \prod_{i=1}^{d} p(x_i|y=c;\theta_c) = \prod_{i=1}^{d} \theta_{ic}^{x_i}(1-\theta_{ic})^{(1-x_i)}$$
  + First, log it:
    $$\log p(x|y=c;\theta_c) = \sum_{i=1}^{d} x_i \log\theta_{ic} + \sum_{i=1}^{d}(1-x_i)\log(1-\theta_{ic})$$
    $$= \sum_{i=1}^{d} x_i \log\theta_{ic} + \sum_{i=1}^{d} x_i \log(1-\theta_{ic}) + \sum_{i=1}^{d}\log(1-\theta_{ic})$$
    $$= \sum_{i=1}^{d} x_i\left(\log(\theta_{ic}/1-\theta_{ic})\right) + \sum_{i=1}^{d}\log(1-\theta_{ic})$$

$p(x|y=c;\theta_c) = \exp(\log p)$.

define $w_{cj} = \log(\theta_{jc}/1-\theta_{jc})$, $w_{0c} = \sum_{j=1}^{d} \log(1-\theta_{jc})$

$P(x|y=c;\theta_c) = \exp(w_c^T x + w_{0c})$

HOW TO CLASSIFY:

$P(y=c|x) \propto P(x|y=c;\theta_c)\, P(y=c|\theta_c)$.

Thus,

$P(y=c|x) = \exp[w_c^T x + w_{0c} + \log(P(y=c|\theta_c))]$

LINEAR CLASSIFIER!

Really easy to train. NO DERIVATIVES. w_{cj} is just COUNTING $O(d)$.

+ $p(y=c|\theta_c) = \frac{\#\text{ class } c}{N}$ Classify by $\max_c P(y=c|x)$

+ PROS:
  + Fast train, test
  + Less overfit, can be better than logistic on small data
  + Easy to add/remove classes
  + Can handle partial data

+ CONS:
  + Naive IID doesn't always hold!

GAUSSIAN MIXTURE MODELS + EM

+ GOOD CLUSTERING → assume data generated by generative model!
  → adjust MODEL PARAMS to maximize probability that the model produces the data we observe.

+ Usually: $p(x,z) = p(x|z)p(z)$, but we don't know z?

So, MIXTURE MODEL:

$$p(x) = \sum_z p(x,z) = \sum_z p(x|z)p(z)$$

+ GMM: K classes, $p(z=u) = \pi_u$, $p(x|z=u) = \mathcal{N}(x|\mu_u, \Sigma_u)$

Then, $p(x) = \sum_{u=1}^{K} \mathcal{N}(x|\mu_u, \Sigma_u)\pi_u$ and $\sum_u \pi_u = 1$

+ UNIVERSAL DENSITY APPROXIMATOR!

+ MAXIMUM LIKELIHOOD:

$$\log p(X|\pi,\mu,\Sigma) = \log\left(\prod_{i=1}^{N} p(x^i)\right) = \sum_{i=1}^{N} \log\left(\sum_{u=1}^{K} \mathcal{N}(x^i|\mu_u,\Sigma_u)\pi_u\right)$$

  + SINGULARITIES: ↑ likelihood when Gaussn. explains single pt.
  + NONCONVEX
  + Soln. INVARIANT to permutes

+ LATENT VARIABLES: z^i for each x^i.

Then,

$$\log p(X|\pi,\mu,\Sigma) = \sum_{i=1}^{N} \log\left(\sum_{u=1}^{K} p(x^i|z^i;\mu,\Sigma)\, p(z^i|\pi)\right)$$

We don't know z^i for every x^i.

(else: $\mu_u = \frac{1}{\delta(z^i,u)} \sum_{i=1}^{N} \delta(z^i,u)x^i$ (mean of all u-x's)

$\Sigma_u = \frac{1}{\delta(z^i,u)} \sum_{i=1}^{N} \delta(z^i,u)(x^i-\mu_u)(x^i-\mu_u)^T$ (covar.)

$\pi_u = \frac{1}{N} \sum_{i=1}^{N} \delta(z^i,u)$. (all u's over N).)

+ IDEA: The EM algorithm to OPTIMIZE PARAMETERS

1) E: COMPUTE POST. PROB OVER z GIVEN CURRENT MODEL PARAMS. (How much do we think each Gaussian gens each datapt)

M: Assume data was generated this way. CHANGE PARAMS of each Gaussian to MAXIMIZE PROB that it would gen. the data it's resp. for

E-STEP:

1) Compute post. prob over z

$\gamma_u^i = p(z^i = u \mid x^i; \pi, \mu, \Sigma)$.

"Which Gaussn. gen'd each dpt?" ARN'T SURE → DISTRIBUTION OVER ALL POSSIBLE GAUSSIANS. (u).

M-STEP:

1) Each Gaussian now has some posterior prob for each data point, which is γ_u^i

2) FIT each GAUSSIAN to the WEIGHTED DPTS.

3) Derive closed-form UPDATES of ALL PARAMS

WHERE DOES EM COME FROM?

+ Let $\theta = \mu, \pi, \Sigma$.

+ $\ell(x, \theta) = \sum_i \log(p(x^i; \theta)) = \sum_i \log\left(\sum_u p(x^i, z^i = u; \theta)\right)$ ← hard! (likelihood)

+ A TRICK IN ML: INTRODUCE NEW DISTRIBUTION q:

$$\ell(x, \theta) = \sum_i \log\left(\sum_u q_u \frac{p(x^i, z = u; \theta)}{q_u}\right)$$

+ JENSEN'S INEQUALITY: (concave fxn like log)

$f(\mathbb{E}[x]) \geq \mathbb{E}[f(x)]$

$f\left(\sum_i p_i x_i\right) \geq \sum_i p_i f(x_i)$

+ Apply JENSEN to the above: (maximize this = lower bd = increase likelihood!)

$$\ell(x, \theta) = \sum_i \log\left(\sum_u q_u \frac{p(x^i, z = u; \theta)}{q_u}\right) \geq \sum_i \sum_u q_u \log\left(\frac{p(x^i, z = u, \theta)}{q_u}\right)$$

+ Pick $q_u = p(z^i = u \mid x^i; \theta^{old})$.

$$Q(\theta) = \sum_i \sum_u p(z^i = u \mid x^i; \theta^{old}) \log(p(x^i, z^i = u; \theta)).$$

$$= \mathbb{E}_{p(z^i = u \mid x^i; \theta^{old})}\left[\log p(x^i, z^i; \theta)\right] \leftarrow$$

+ CONCEPTUALLY: we don't know z^i, so we AVERAGE them given the CURRENT MODEL θ^{old}.

E-STEP:

calculating POST PROB RESPONSIBILITIES:

$$\gamma_u = p(z=u|x) = \frac{p(x|z=u)\,p(z=u)}{p(x)} = \frac{p(x|z=u)\,p(z=u)}{\sum_{j=1}^{k} p(x|z=j)\,p(z=j)} \quad \left.\right\} \text{Bayes}$$

$$= \frac{\mathcal{N}(x|\mu_u, \Sigma_u)\,\pi_u}{\sum_{j=1}^{u} \mathcal{N}(x|\mu_j, \Sigma_u)\,\pi_j} \quad \left.\right\} \text{Gaussian MM.}$$

+ Now we have $\gamma_u^i = p(z^i=u|x)$
+ Calculating EXPECTED LIKELIHOOD:

$$\mathbb{E}_{P(z^i|x^i)}\left[\sum_i \log\left(P(x^i, z^i|\theta)\right)\right] =$$

$$= \mathbb{E}_{P(z^i|x^i)}\left[\sum_i \log\left(P(x^i|z^i;\theta)\,P(z^i|\theta)\right)\right]$$

$$= \sum_i \sum_u \gamma_u^i \log\left(P(x^i|z^i=u;\theta)\right) + \gamma_u^i \log\left(P(z^i=u|\theta)\right)$$

$$= \sum_i \sum_u \gamma_u^i \left(\log\left(\mathcal{N}(x^i;\mu_u,\Sigma_u)\right) + \log(\pi_u)\right)$$

$$= \sum_u \sum_i \gamma_u^i \log\left(\mathcal{N}(x^i;\mu_u,\Sigma_u)\right) + \sum_u \sum_i \gamma_u^i \log(\pi_u).$$

GMM M-STEP

+ Need to optimize $\sum_u \sum_i \gamma_u^i \log\left(\mathcal{N}(x^i;\mu_u,\Sigma_u)\right) + \sum_u \sum_i \gamma_u^i \log(\pi_u)$
+ Solving for μ_u, Σ_u is like FITTING k SEP GAUSSNS, weighted γ_u^i
+ similar to generative derivation

ALGO:

1) Init π_u, Σ_u, μ_u
2) ITERATE TILL CONVG:
   3) E-STEP: Calculate post probs $\gamma_u^i = p(z^i=u|x)$.
   4) M-STEP: Re-estimate params w/ new responsibilities

$$\mu_u = \frac{1}{N_u} \sum_{i=1}^{N} \gamma_u^i x^i$$

$$\Sigma_u = \frac{1}{N_u} \sum_{i=1}^{N} \gamma_u^i (x^i - \mu_u)(x^i - \mu_u)^T$$

$$\pi_u = \frac{N_u}{N} \qquad N_u = \sum_{i}^{N} \gamma_u^i$$

   5) Eval log-likelihood, check for convergence.

K-MEANS

+ Why does it converge?
  + Sum of all distances between pts and means J always REDUCES, and always start pos, so will conv. to 0
  + CONVERGENCE → assignments do not change
+ Local Minima?
  + J is NONCONVEX. Can GET STUCK in local minima
  + Fix: – multiple random starts
    – split big cluster in 2, merge two nearby clusters
+ k-means ++:
  + Random & choose first mean/center
  + $\forall x^i$ set d^i = dist. closest to center
  + Pick new center to be at x^i w/ prob $d(i)^2$
  + Repeat till k centers.
+ Soft K-Means: ALGO
  1) Set μ_k to rando
  2) ITERATE till conv:
     3) $\forall x^i$, check all μ_k:
     4) Give $r_k = \frac{\exp(-\beta d(m_k, x^i))}{\sum \exp[-\beta d(m_k, x^i)]}$ } SOFTMAX ASSIGNMENT
     5) Update model params
        $m_k = \frac{\sum r_i^i x^i}{\sum r_i^i}$ (like hard k-Means)

LE METHODS

S-VARIANCE DECOMP:

+ regression w/ L_2 loss, $h^*(x) = \mathbb{E}[t|x]$.
+ ML prediction: $y(x;D)$
+ Breaking $\mathbb{E}_{D,x,t}[(t-y(x;D))^2]$ into core components:
  + $\mathbb{E}_{D,x,t}[(t-y(x;D))^2] = \mathbb{E}_{D,x,t}[(t-h^*(x)+h^*(x)-y(x;D))^2]$

$$= \mathbb{E}_t[(t-h^*(x)+h^*(x)-y(x;D))^2|x,D]$$
$$= \mathbb{E}_t[(t-h^*(x))^2|x,D] + \mathbb{E}_t[(h^*(x)-y(x;D))^2|x,D]$$
$$+ 2\,\mathbb{E}_t[(t-h^*(x))(h^*(x)-y(x;D))]$$
$$2\,(\mathbb{E}_t[t|x]-h^*(x))(h^*(x)-y(x;D))$$
$$= 2\,(h^*(x)-h^*(x))(h^*(x)-y(x;D))$$

Thus:

$$\underbrace{\mathbb{E}_{D,x,t}[(t-y(x;D))^2]}_{\text{ERROR}} = \underbrace{\mathbb{E}_t[(t-h^*(x))^2|x,D]}_{\text{noise}} + \mathbb{E}_t[(h^*(x)-y(x;D))^2|x,D]$$

Breakdown: $\mathbb{E}_D[(h^*(x)-y(x;D))^2|x] = \mathbb{E}_D[(h^*(x)-\mathbb{E}_D[y(x;D)]+\mathbb{E}_D[y(x;D)]-y(x;D))^2|x]$

$$= \mathbb{E}_D[(h^*(x)-\mathbb{E}_D[y(x;D)])^2|x] + \mathbb{E}_D[(\mathbb{E}_D[y(x;D)]-y(x;D))^2|x]$$
$$+ 2\,\mathbb{E}_D[(h^*(x)-\mathbb{E}_D[y(x;D)])(\mathbb{E}_D[y(x;D)]-y(x;D))|x]$$
$$2\,(h^*(x)-\mathbb{E}_D[y(x;D)])(\mathbb{E}_D[y(x;D)]-\mathbb{E}_D[y(x;D)])$$
$$2\,(h^*(x)-\mathbb{E}_D[y(x;D)])\cdot 0$$

Thus,

$$\mathbb{E}_{D,x,t}[\underbrace{(t-y(x;D))^2}_{\text{error}}] = \mathbb{E}_{D,x}[\underbrace{(t-h^*(x))^2}_{\text{noise}}] + \mathbb{E}_x[\underbrace{(h^*(x)-\mathbb{E}_D[y(x;D)])^2}_{(\text{bias})^2}] + \mathbb{E}_{D,x}[(y(x;D)-\mathbb{E}_D[y(x;D)])^2]$$

+ HIGH VARIANCE (OVERFITS) ⇔ LOW BIAS
+ LOW VARIANCE ⇔ HIGH BIAS

+ BAGGING
  + TO REDUCE VARIANCE ~ OVERFITTING
  + BOOTSTRAPPING: sample w/ replacement from dset to get variance estimate

  + ALGO:
    INPUT: D, ML algo A, # bags N.
    1) ITER i : 1 → N :
       2) Get $D_i \subseteq D$ by SAMPLING w/ REPLACEMENT
       3) $f_i = A(D_i)$.
    4) Return $f_1 \dots f_N$
    5) PREDICTION: Return $\frac{1}{N} \sum_i f_i(x)$ or majority vote.

+ RANDOM FOREST
  + Random subset of features on each split → reduce correlation.
  + D-Trees: easily overfit, fast inference, good on relational data.

  + OOB EST: cheaply estimate test loss → can reduce validation need
    + PREDICT each TRAINING EXAMPLE using ALL TREES that DID NOT CONTAIN THIS EXAMPLE in training data.
    + Slightly worse loss than real test error → upper bound.

+ Reduces overfits by avg. predictions
+ DOES NOT REDUCE BIAS. STILL CLASSIFIER CORRELATION,
  + RF soln: MORE RANDOMNESS

STING

Reduces BIAS.

+ Idea: Take M WEAK CLASSIFIERS that does ε better than chance and BOOST IT to get LOW TRAINING ERROR.

+ ADABOOST ALGO:

INPUT: $\{x^i, t^i\}_{i=1}^N$, WeakLearn learning procedure.

INIT: D is an N×M matrix where each column D_m is a set of weights for the N examples. Initialize $D_1^i(x) = 1/N$. for all 1 to N. i's.

$$\begin{bmatrix} 1/N & 0 & \cdots & 0 \\ 1/N & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 1/N & 0 & \cdots & 0 \end{bmatrix}$$

1) ITER: $m = 1 \rightarrow M$

2) $h_m(x) = \text{WeakLearn}(\{x\}, t, w)$. Get weak classifier

3) Minimize:

$$J_m = D_m^1[h_m(x^1) \neq t^1] + D_m^2[h_m(x^2) \neq t^2] + \cdots + D_m^N[h_m(x^N) \neq t^N].$$

(idea: J_m is the "score" of misclassifications by the weak classifier, weighted by D_m).

4) Compute WEIGHTED ERROR RATE:

$$\epsilon_m = \frac{J_m}{\sum_i D_m^i} \quad \text{always} < 1/2$$

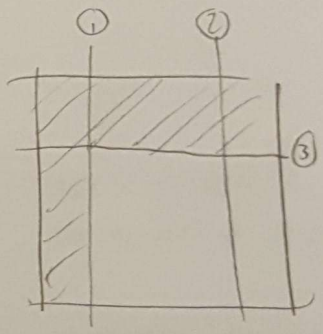
weighted misclass mean

weighted misclass mean.

5) Compute CLASSIFIER COEFF: $\alpha_M = \frac{1}{2} \log \frac{1-\epsilon_m}{\epsilon_m}$

6) UPDATE DATA WEIGHTS:

$$D_{m+1}^i = D_m^i \exp(-t^i \alpha_m h_m(x^i))$$



7) FINAL:

$$H(x) = \text{sign}\left(\sum_{m=1}^{N} \alpha_m h_m(x)\right) = \text{sign}(F(x))$$

+ BOOSTING → greedy optimization of exponential loss