Reminder: Classification

- Given examples of a discrete class label $y$ and some features $x$.
- Goal: compute label ($y$) for new inputs $x$.
- Two approaches:
  - Generative: model $p(x, y) = p(y)p(x|y)$; use Bayes’ rule to infer conditional $p(y|x)$.
  - Discriminative: model discriminants $f(y|x)$ directly and take max.
- Generative approach is related to conditional density estimation while discriminative approach is closer to regression.

Probabilistic Classification: Bayes Classifiers

- Generative model: $p(x, y) = p(y)p(x|y)$.
  $p(y)$ are called class priors.
  $p(x|y)$ are called class conditional feature distributions.
- For the prior we use a Bernoulli or multinomial: $p(y = k|\pi) = \pi_k$ with $\sum_k \pi_k = 1$.
- Classification rules:
  - ML: $\text{argmax}_y p(x|y)$ (can behave badly if skewed priors)
  - MAP: $\text{argmax}_y p(y|x) = \text{argmax}_y \log p(x|y) + \log p(y)$ (safer)
- Fitting: maximize $\sum_n \log p(x^n, y^n) = \sum_n \log p(x^n|y^n) + \log p(y^n)$
  1) Sort data into batches by class label.
  2) Estimate $p(y)$ by counting size of batches (plus regularization).
  3) Estimate $p(x|y)$ separately within each batch using ML.
    (also with regularization).

Three Key Regularization Ideas

- To avoid overfitting, we can put priors on the parameters of the class and class conditional feature distributions.
- We can also tie some parameters together so that fewer of them are estimated using more data.
- Finally, we can make factorization or independence assumptions about the distributions. In particular, for the class conditional distributions we can assume the features are fully dependent, partly dependent, or independent (!).
Gaussian Class-Conditional Distributions

- If all features are continuous, a popular choice is a Gaussian class-conditional.
  \[ p(x|y = k; \theta) = |2\pi \Sigma|^{-1/2} \exp \left\{ -\frac{1}{2}(x - \mu_k)\Sigma^{-1}(x - \mu_k) \right\} \]

- Fitting: use the following amazing and useful fact.
  The maximum likelihood fit of a Gaussian to some data is the Gaussian whose mean is equal to the data mean and whose covariance is equal to the sample covariance.
  [Try to prove this as an exercise in understanding likelihood, algebra, and calculus all at once!]

- Seems easy. And works amazingly well. But we can do even better with some simple regularization...

Regularized Gaussians

- Idea 1: assume all the covariances are the same (tie parameters). This is exactly Fisher’s linear discriminant analysis.

- Idea 2: Make independence assumptions to get diagonal or identity-multiple covariances. (Or sparse inverse covariances.) More on this in a few minutes...

- Idea 3: add a bit of the identity matrix to each sample covariance. This “fattens it up” in directions where there wasn’t enough data. Equivalent to using a “Wishart prior” on the covariance matrix.

Gaussian Bayes Classifier

- Maximum likelihood estimates for parameters:
  - priors \( \pi_k \): use observed frequencies of classes (plus smoothing)
  - means \( \mu_k \): use class means
  - covariance \( \Sigma \): use data from single class or pooled data \( (x_m - \mu_{y_m}) \) to estimate full/diagonal covariances

- Compute the posterior via Bayes’ rule:
  \[
  p(y = k|x, \theta) = \frac{p(x|y = k, \theta)p(y = k|\pi)}{\sum_j p(x|y = j, \theta)p(y = j|\pi)} = \frac{\exp\{\mu_k^T \Sigma^{-1}x - \mu_k^T \Sigma^{-1} \mu_k / 2 + \log \pi_k\}}{\sum_j \exp\{\mu_j^T \Sigma^{-1}x - \mu_j^T \Sigma^{-1} \mu_j / 2 + \log \pi_j\}} = \frac{e^{\beta_k^T x}}{\sum_j e^{\beta_j^T x}} = \exp\{\beta_k^T x\}/Z
  \]
  where \( \beta_k = [\Sigma^{-1} \mu_k; (\mu_k^T \Sigma^{-1} \mu_k + \log \pi_k)] \) and we have augmented \( x \) with a constant component always equal to 1 (bias term).

Softmax/Logit

- The squashing function is known as the softmax or logit:
  \[
  \phi_k(z) = \frac{e^{z_k}}{\sum_j e^{z_j}} \quad g(\eta) = \frac{1}{1 + e^{-\eta}}
  \]

- It is invertible (up to a constant):
  \[
  z_k = \log \phi_k + c \quad \eta = \log(g/1 - g)
  \]

- Derivative is easy:
  \[
  \frac{\partial \phi_k}{\partial z_j} = \phi_k (\delta_{kj} - \phi_j) \quad \frac{dg}{d\eta} = g(1 - g)
  \]
Linear Geometry

- Taking the ratio of any two posteriors (the “odds”) shows that the contours of equal pairwise probability are linear surfaces in the feature space:
  \[ \frac{p(y = k|x, \theta)}{p(y = j|x, \theta)} = \exp \{(\beta_k - \beta_j)^\top x\} \]

- The pairwise discrimination contours \( p(y_k) = p(y_j) \) are orthogonal to the differences of the means in feature space when \( \Sigma = \sigma I \).
  For general \( \Sigma \) shared b/w all classes the same is true in the transformed feature space \( w = \Sigma^{-1} x \).

- The priors do not change the geometry, they only shift the operating point on the logit by the log-odds \( \log(\pi_k/\pi_j) \).
- Thus, for equal class-covariances, we obtain a linear classifier.
- If we use difference covariances, the decision surfaces are conic sections and we have a quadratic classifier.

Discrete Bayesian Classifier

- If the inputs are discrete (categorical), what should we do?
- The simplest class conditional model is a joint multinomial (table):
  \[ p(x_1 = a, x_2 = b, \ldots | y = c) = \eta_{ab}^c \]
- This is conceptually correct, but there's a big practical problem.
- Fitting: ML params are observed counts:
  \[ \eta_{ab}^c = \frac{\sum_n[y_n = c][x_1 = a][x_2 = b][\ldots][\ldots]}{\sum_n[y_n = c]} \]
- Consider the 16x16 digits at 256 gray levels.
- How many entries in the table? How many will be zero? What happens at test time? Doh!
- We obviously need some regularization.
  Smoothing will not help much here. Unless we know about the relationships between inputs beforehand, sharing parameters is hard also. But what about independence?

Exponential Family Class-Conditionals

- Bayes Classifier has the same softmax form whenever the class-conditional densities are any exponential family density:
  \[ p(x|y = k, \eta_k) = h(x) \exp\{\eta_k^\top x - a(\eta_k)\} \]
  \[ p(y = k|x, \eta) = \frac{p(x|y = k; \eta_k)p(y = k|\pi)}{\sum_j p(x|y = j; \eta_j)p(y = j|\pi)} \]
  \[ = \frac{\exp\{\eta_k^\top x - a(\eta_k)\}}{\sum_j \exp\{\eta_j^\top x - a(\eta_j)\}} \]
  \[ = \frac{e^{\beta_k^\top x}}{\sum_j e^{\beta_j^\top x}} \]
  where \( \beta_k = [\eta_k; -a(\eta_k)] \) and we have augmented \( x \) with a constant component always equal to 1 (bias term).
- Resulting classifier is linear in the sufficient statistics.

Naive (Idiot’s) Bayes Classifier

- Assumption: conditioned on class, attributes are independent.
  \[ p(x|y) = \prod_i p(x_i|y) \]
- Sounds crazy right? Right! But it works.
- Algorithm: sort data cases into bins according to \( y_n \).
  Compute marginal probabilities \( p(y = c) \) using frequencies.
- For each class, estimate distribution of \( i^{th} \) variable: \( p(x_i|y = c) \).
- At test time, compute \( \arg\max_c p(c|x) \) using
  \[ c(x) = \arg\max_c p(c|x) = \arg\max_c [\log p(x|c) + \log p(c)] \]
  \[ = \arg\max_c [\log p(c) + \sum_i \log p(x_i|c)] \]
Discrete (Multinomial) Naive Bayes

Discrete features $x_i$, assumed independent given the class label $y$.

$$p(x_i = j | y = k) = \eta_{ijk}$$
$$p(x | y = k, \eta) = \prod_i \prod_j \eta_{ijk}$$

Classification rule:

$$p(y = k | x, \eta) = \frac{\pi_k \prod_i \prod_j \eta_{ijk}[x_i = j]}{\sum_q \pi_q \prod_i \prod_j \eta_{iqj}}$$

$$\beta_k = \log[\eta_{11k} \ldots \eta_{ijk} \ldots \eta_{ijk} \ldots \log \pi_k]$$
$$x = [x_1 = 1; x_1 = 2; \ldots; x_i = j; \ldots; 1]$$

Fitting Discrete Naive Bayes

- ML parameters are class-conditional frequency counts:
  $$\eta^*_{ijk} = \frac{\sum_m [x_i^m = j][y^m = k]}{\sum_m [y^m = k]}$$

- How do we know? Write down the likelihood:
  $$\ell(\theta; D) = \sum_m \log p(y^m | \pi) + \sum_{mi} \log p(x_i^m | y^m, \eta)$$

  and optimize it by setting its derivative to zero 
  (careful! enforce normalization with Lagrange multipliers):

  $$\ell(\eta; D) = \sum_{m} \sum_{ik} [x_i^m = j][y^m = k] \log \eta_{ijk} + \sum_{ik} \lambda_{ik}(1 - \sum_j \eta_{ijk})$$

  $$\frac{\partial \ell}{\partial \eta_{ijk}} = \frac{\sum_m [x_i^m = j][y^m = k]}{\eta_{ijk}} - \lambda_{ik}$$

  $$\frac{\partial \ell}{\partial \eta_{ijk}} = 0 \Rightarrow \lambda_{ik} = \sum_m [y^m = k] \Rightarrow \eta^*_{ijk} = \text{above}$$
Logistic/Softmax Regression

- Model: $y$ is a multinomial random variable whose posterior is the softmax of linear functions of any feature vector.

$$p(y = k | x, \theta) = \frac{e^{\theta_k^\top x}}{\sum_j e^{\theta_j^\top x}}$$

- Fitting: now we optimize the conditional likelihood:

$$\ell(\theta; D) = \sum_{mk} [y^m = k] \log p(y = k | x^m, \theta) = \sum_{mk} y^m_k \log p^m_k$$

$$\frac{\partial \ell}{\partial \theta_i} = \sum_{mk} \frac{\partial \ell^m_k}{\partial p^m_k} \frac{\partial p^m_k}{\partial z^m_i} \frac{\partial z^m_i}{\partial \theta_i}$$

$$= \sum_{mk} \frac{y^m_k}{p^m_k} p^m_k (\delta_{ik} - p^m_i) x^m$$

$$= \sum_m (y^m_k - p^m_k) x^m$$

More on Logistic Regression

- Hardest Part: picking the feature vector $x$.
- Amazing fact: the conditional likelihood is (almost) convex in the parameters $\theta$. Still no local minima!
- Gradient is easy to compute; so easy (if slow) to optimize using gradient descent or Newton-Raphson / IRLS.
- Why almost? Consider what happens if there are two features with identical classification patterns in our training data. Logistic Regression can only see the sum of the corresponding weights.
- Solution? Weight decay: add $\epsilon \sum \theta^2$ to the cost function, which subtracts $2\epsilon \theta$ from each gradient.
- Why is this method called logistic regression?
- It should really be called “softmax linear regression”.
- Log odds (logit) between any two classes is linear in parameters.

Joint vs. Conditional Models

- Many of the methods we have seen so far have linear or piecewise linear decision surfaces in some space $x$:
  LDA, perceptron, Gaussian Bayes, Naive Bayes, KNN,...
- But the criteria used to find this hyperplane is different:
  - Naive Bayes is a joint model; it optimizes $p(x, y) = p(x)p(y | x)$.
  - Logistic Regression is conditional: optimizes $p(y | x)$ directly.

Other Models

- Noisy-OR (see slides)
- Classification via Regression (see slides)
- Non-parametric (e.g. K-nearest-neighbor).
- Semiparametric (e.g. kernel classifiers, support vector machines, Gaussian processes).
- Probit regression.
- Complementary log-log.
- Generalized linear models.
- Some return a value for $y$ without a distribution.
Noisy-OR Classifier

- Many probabilistic models can be obtained as noisy versions of formulas from propositional logic.
- Noisy-OR: each input $x_i$ activates output $y$ w/some probability.

\[
p(y = 0|\mathbf{x}, \alpha) = \prod_i \alpha_i^{x_i} = \exp \left\{ \sum_i x_i \log \alpha_i \right\}
\]

- Letting $\theta_i = -\log \alpha_i$ we get yet another linear classifier:

\[
p(y = 1|\mathbf{x}, \theta) = 1 - e^{-\theta^T \mathbf{x}}
\]

Classification via Regression

- Binary case: $p(y = 1|\mathbf{x})$ is also the conditional expectation.
- So we could forget that $y$ was a discrete (categorical) random variable and just attempt to model $p(y|\mathbf{x})$ using regression.
- One idea: do regression to an indicator matrix.
- For two classes, this is equivalent* to LDA. For 3 or more, disaster...
- Very bad idea! Noise models (e.g. Gaussian) for regression are totally inappropriate, and fits are oversensitive to outliers. Furthermore, gives unreasonable predictions $< 0$ and $> 1$. 

\[
\begin{align*}
\text{Graph 1} & \quad \text{Graph 2} \\
0 & \quad 0 \\
0.2 & \quad 0.2 \\
0.4 & \quad 0.4 \\
0.6 & \quad 0.6 \\
0.8 & \quad 0.8 \\
1 & \quad 1 \\
-0.2 & \quad -0.2 \\
0 & \quad 0 \\
0.2 & \quad 0.2 \\
0.4 & \quad 0.4 \\
0.6 & \quad 0.6 \\
0.8 & \quad 0.8 \\
1 & \quad 1 \\
1.2 & \quad 1.2 \\
\end{align*}
\]