Tutorial 1

CS 337 Artificial Intelligence & Machine Learning, Autumn 2019

Week 1, August, 2019

Problem 1. A professor decides she will conduct an in-class quiz exactly on those days when at least j of the N registered students show up to class. Each student independently shows up to class with probability p if it is not raining and with probability q if it is raining outside. Suppose on a given day, the probability that it will rain is r. Provide an expression for the probability that the professor conducts an in-class quiz on that day.

Problem 2. A geometric random variable X with parameter p has the following probability mass function:

$$P(X = i) = p(1-p)^{i-1}, \qquad i = 1, 2, \ldots$$

Suppose X and Y are both independent, identically distributed, geometric random variables with parameter p. Show that

$$P(X = i | X + Y = n) = \frac{1}{n-1}, \qquad i = 1, \ldots, n-1$$

Problem 3. If $\mathbf{u}$ and $\mathbf{v}$ are two orthonormal vectors in $\mathbb{R}^8$, what is the formula for the projection of any vector $\mathbf{a}$ onto the plane spanned by the two vectors $\mathbf{u}$ and $\mathbf{v}$? How does this formula change if $\mathbf{u}$ and $\mathbf{v}$ are unit vectors but not orthogonal to each other?

Problem 4. We discussed in class that maximizing a monotonically increasing function of an objective is more convenient (and at times more intuitive) than maximizing the original objective. Prove that maximizing a strictly monotonically increasing transformation of the objective gives the same optimality point as does maximizing the original objective. (Hint: Prove by contradiction)

Answer: We will prove by contradiction. Let $O(\theta)$ be the objective function being maximized. Let $\theta^* = \underset{\theta}{\text{argmax}}\ O(\theta)$. Let $f(\beta)$ be a monotonically increasing function. Let $\hat{\theta} = \underset{\theta}{\text{argmax}}\ f(O(\theta))$ such that $\hat{\theta} \neq \theta^*$ and $f(O(\hat{\theta})) > f(O(\theta^*))$. Since f is a monotonically increasing function of its arguments, it must be that $O(\hat{\theta}) > O(\theta^*)$. Which is a contradiction, since we had $\theta^* = \underset{\theta}{\text{argmax}}\ O(\theta)$. Thus either, it must be that $\hat{\theta} = \theta^*$ OR $f(O(\hat{\theta})) = f(O(\theta^*))$.

Problem 5. A student of CS 337 measured the height of each student in class along with his/her age, weight and the heights of the child's parents. The student fit a linear regression model to predict the height as a function of the other observations i.e., height = f(age, weight, height of the child's mother, height of the child's father). The student suddenly realizes that she measured the height of each child with his/her shoe on. This meant that the estimated model had to be corrected somehow. It was known that every child in the school wore a shoe of 1.5 mm thickness. What is the simplest correction the ML student can apply to the model (without computing it all over again) to get the same result as she would have obtained by using the correct height (excluding the shoe thickness)? Justify your answer.

Solution:
Consider

1. $\hat{\mathbf{w}_{ML}} = \text{argmin} \sum_{j=1}^{m} (\mathbf{w}^T \phi(\mathbf{x}_j) + b - y_j)^2$

2. $\hat{\mathbf{w}_{ML}} = \text{argmin} \sum_{j=1}^{m} (\mathbf{w}^T \phi(\mathbf{x}_j) + b_{new} - (y_j + 1.5))^2$

Claim is $b_{new} = b + 1.5$, where one can obtain the solution b_{new} to (2) simply by subtracting 1.5 mm from b. If not, then one can show that the solution $b_{new} \neq b + 1.5$ to the second should have yielded a better solution $b_{new} - 1.5 \neq b$ to (1) - which is a contradiction.

Problem 6. Consider a random variable Y that is generated, conditional on X, based on the following process:

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$
$$Y = aX + \epsilon$$

Assume we have a training dataset of m pairs $\{x_i, y_i\}$ for $i = 1..m$, and σ is known. Analytically derive the correct expression for the maximum likelihood estimate of the parameter a in terms of x_i's and y_i's.

Solution:

This is a very special case of maximum likelihood estimation for linear regression with $\phi(\mathbf{x}) \in \Re$ (that is, with $p = 1$). This can be also be solved by using $\mathbf{w} = (\mathbf{\Phi^T \Phi})^{-1}\mathbf{\Phi^T y}$ and substituting $\Phi = [x_1; x_2; ...x_m]$ and $\mathbf{w} = [a]$ to get $a = \frac{\sum_i y_i x_i}{\sum_i x_i^2}$

Another way of proving $a = \frac{\sum_i y_i x_i}{\sum_i x_i^2}$ is from first principles. Here is the second method:

Solve for $\arg\max_a \prod_i exp\big(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\big)$. Equivalently, you could also solve for maximizing the monotonically increasing (log) transformation of the objective

$$\arg\max_a \log\left(\prod_i exp\big(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\big)\right) = \sum_i \left(\big(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\big)\right)$$

Taking partial derivative w.r.t. a we get $\sum_i ax_i^2 - x_i y_i = 0$. That is, $a = \frac{\sum_i y_i x_i}{\sum_i x_i^2}$

Tutorial 2

CS 337 Artificial Intelligence & Machine Learning, Autumn 2019

Week 2, August, 2019

Problem 1. Consider a data set in which each data point y_i is associated with a weighting factor r_i, so that the sum-square error function becomes

$$\frac{1}{2}\sum_{i=1}^{m} r_i(y_i - w^T\phi(x_i))^2$$

Find an expression for the solution w^* that minimizes this error function. The weights r_i's are known before hand. (Exercise 3.3 of Pattern Recognition and Machine Learning, Christopher Bishop).

Solution:

Let $r_{m+1} = 1$ and let R be an $(m+1) \times (m+1)$ diagonal matrix of $r_1, r_2, \ldots, r_{m+1}$.

$$R = \begin{bmatrix} r_1 & 0 & ... & 0 & \\ 0 & r_2 & ... & 0 & \\ ... & ... & ... & ... & 1 \\ 0 & 0 & 0 & ... & r_{m+1} \end{bmatrix}$$

Further, let

$$\Phi = \begin{bmatrix} \phi_1(x_1) & ... & \phi_p(x_1) & 1 \\ ... & ... & ... & 1 \\ \phi_1(x_m) & ... & \phi_p(x_m) & 1 \end{bmatrix}$$

and

$$\widehat{\mathbf{w}} = \begin{bmatrix} w_1 \\ ... \\ w_p \\ b \end{bmatrix}$$

and

$$\mathbf{y} = \begin{bmatrix} y_1 \\ ... \\ y_m \end{bmatrix}$$

The sum-square error function then becomes

$$\frac{1}{2}\sum_{i=1}^{m} r_i(y_i - (\widehat{\mathbf{w}}^T\phi(x_i) + b))^2 = \frac{1}{2}||\sqrt{R}\mathbf{y} - \sqrt{R}\Phi\widehat{\mathbf{w}}||_2^2$$

where $\sqrt{R}$ is a diagonal matrix such that each diagonal element of $\sqrt{R}$ is the square root of the corresponding element of R. This is a convex function being minimized (prove this using techniques similar to what we employed for least squares linear regression) and therefore has a global minimum at $\widehat{\mathbf{w}}_*^{x'}$ where the gradient must become 0. (again work out the steps using techniques similar to what we employed for least squares linear regression). The expression for the solution $\widehat{\mathbf{w}}^*$ that minimizes this error function is therefore

$$\widehat{\mathbf{w}}_*^{x'} = (\Phi^T R\Phi)^{-1}\Phi^T R\mathbf{y}$$

Problem 2. Equivalence between Ridge Regression and Bayesian Linear Regression (with fixed σ^2 and λ): Consider the Bayesian Linear Regression Model

$$\begin{aligned} y = \mathbf{w}^T\phi(\mathbf{x}) + \varepsilon \text{ and } \varepsilon \sim \mathcal{N}(0, \sigma^2) \\ \mathbf{w} \sim \mathcal{N}(0, \alpha I) \text{ and } \mathbf{w} \mid \mathcal{D} \sim \mathcal{N}(\mu_m, \Sigma_m) \\ \mu_m = (\lambda\sigma^2 I + \phi^T\phi)^{-1}\phi^T\mathbf{y} \text{ and } \Sigma_m^{-1} = \lambda I + \phi^T\phi/\sigma^2 \end{aligned}$$

Show that $\mathbf{w}_{MAP} = \underset{\mathbf{w}}{\text{argmax}}\ \Pr(\mathbf{w} \mid \mathcal{D})$ is the same as that of *Regularized Ridge Regression.*

$$\mathbf{w}_{Ridge} = \underset{\mathbf{w}}{\text{argmin}}\ ||\phi\mathbf{w} - \mathbf{y}||_2^2 + \lambda\sigma^2||\mathbf{w}||_2^2$$

In other words, The Bayes and MAP estimates for Linear Regression coincide with that of *Regularized Ridge Regression.*

Solution Sketch: Taking the negative log of the log likelihood we see that maximizing the log of the posterior distribution is equivalent to minimizing the ridge regression objective.

$$\Pr(\mathbf{w} \mid \mathcal{D}) = \mathcal{N}(\mathbf{w} \mid \mu_m, \Sigma_m) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma_m|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{w}-\mu_m)^T\Sigma_m^{-1}(\mathbf{w}-\mu_m)}$$

$$-\log\Pr(\mathbf{w}) = \frac{n}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma_m| + \frac{1}{2}(\mathbf{w} - \mu_m)^T\Sigma_m^{-1}(\mathbf{w} - \mu_m)$$

$$\mathbf{w}_{MAP} = \underset{\mathbf{w}}{\text{argmax}} -\log\Pr(\mathbf{w}) = \underset{\mathbf{w}}{\text{argmax}}\frac{1}{2}\mathbf{w}^T\Sigma_m^{-1}\mathbf{w} - \mathbf{w}^T\Sigma_m^{-1}\mu_m$$

that is,

$$\mathbf{w}_{MAP} = \underset{\mathbf{w}}{\text{argmax}}\frac{1}{2}\mathbf{w}^T\left(\lambda I + \phi^T\phi/\sigma^2\right)\mathbf{w} - \mathbf{w}^T\left(\lambda I + \phi^T\phi/\sigma^2\right)\left((\lambda\sigma^2 I + \phi^T\phi)^{-1}\phi^T\mathbf{y}\right)$$

and after expanding and canceling out redundant terms, and later, after completing squares:

$$\mathbf{w}_{MAP} = \underset{\mathbf{w}}{\text{argmax}} \frac{1}{2\sigma^2} \mathbf{w}^T \left(\phi^T \phi \mathbf{w} - 2\phi^T \mathbf{y}\right) + \lambda \mathbf{w}^T \mathbf{w} = \underset{\mathbf{w}}{\text{argmax}} \frac{1}{2} ||\phi \mathbf{w} - \mathbf{y}||^2 + \sigma^2 \lambda ||\mathbf{w}||^2 = \mathbf{w}_{Ridge}$$

Problem 3. Ridge Regression and Error Minimization:

1. *Prove the following Claim:*
   The sum of squares error on training data using the weights obtained after minimizing ridge regression objective is greater than or equal to the sum of squares error on training data using the weights obtained after minimizing the ordinary least squares (OLS) objective.

   More specifically, if ϕ and $\mathbf{y}$ are defined on the training set $\mathcal{D} = \{(\mathbf{x}_1, y_1)...(\mathbf{x}_m, y_m)\}$ as

$$\phi = \begin{bmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & ..... & \phi_n(\mathbf{x_1}) \\ . & & & \\ . & & & \\ \phi_1(\mathbf{x}_m) & \phi_2(\mathbf{x}_m) & ..... & \phi_n(\mathbf{x}_m) \end{bmatrix} \tag{1}$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ . \\ y_m \end{bmatrix} \tag{2}$$

   and if

$$\mathbf{w}_{Ridge} = \underset{\mathbf{w}}{\text{argmin}} \ ||\phi \mathbf{w} - \mathbf{y}||_2^2 + \lambda ||\mathbf{w}||_2^2$$

   and

$$\mathbf{w}_{OLS} = \underset{\mathbf{w}}{\text{argmin}} \ ||\phi \mathbf{w} - \mathbf{y}||_2^2$$

   then you should prove that

$$||\phi \mathbf{w}_{Ridge} - \mathbf{y}||_2^2 \geq ||\phi \mathbf{w}_{OLS} - \mathbf{y}||_2^2$$

   Solution: If

$$\mathbf{w}_{OLS} = \underset{\mathbf{w}}{\text{argmin}} \ ||\phi \mathbf{w} - \mathbf{y}||_2^2$$

   then by definition of argmin,

$$||\phi \mathbf{w}_{Ridge} - \mathbf{y}||_2^2 \geq ||\phi \mathbf{w}_{OLS} - \mathbf{y}||_2^2$$

   Also, one can reformulate

$$\mathbf{w}_{Ridge} = \underset{\mathbf{w}}{\text{argmin}} \ ||\phi \mathbf{w} - \mathbf{y}||_2^2 + \lambda ||\mathbf{w}||_2^2$$

as

$$\mathbf{w}_{Ridge} = \underset{\mathbf{w}}{\operatorname{argmin}} \ ||\phi\mathbf{w} - \mathbf{y}||_2^2$$

$$such\ that\ ||\mathbf{w}||_2^2 \leq \theta$$

for some θ corresponding to a value of λ. The solution to a constrained minimization problem will always be greater than or equal to its unconstrained counterpart.

2. If it is the case that ridge regression leads to greater error than ordinary least squares regression, then why should one be interested in ridge regression at all?

   Solution: This is still acceptable since ridge regression incorporates prior (as per Bayesian interpretation). The idea is ultimately to do well on unseen (test) data. Therefore, higher training error might be acceptable if test error can be lowered.

Problem 4. Gradient descent is a very helpful algorithm. But it is not guaranteed to converge to global minima always. Give an example of a continuous function and initial point for which gradient descent converges to a value which is not global minima.

Problem 5. In class, we have illustrated Bayesian estimation for the parameter μ of a Normally distributed random variable $X \sim \mathcal{N}(\mu, \sigma^2)$, assuming that σ was known by imposing a Normal (conjugate) prior on μ. Now suppose that the parameter μ is known and we wish to estimate σ^2. What will be the form of the conjugate prior for this estimation procedure? If $\mathcal{D} = X_1, X_2, X_3, ..., X_n$ is a set of independent samples from this distribution, after imposing the conjugate prior, compute the form of the likelihood function $\mathcal{L}(\theta)$, the posterior density $P(\theta \mid \mathcal{D})$ and the posterior probability $P(X \mid \mathcal{D})$. Again, you can ignore normalization factors.

Solution:

- Let $\Pr(X) \sim \mathcal{N}(\mu, \sigma^2)$ and let the data $\mathcal{D} = x_1 ... x_m$
- $\mu_{MLE} = \frac{1}{m} \sum_{i=1}^{m} x_i$ and $\sigma^2_{MLE} = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{MLE})^2$
- Suppose you are told that σ^2 is a random variable is and μ is not.

$$\Pr(x_i|\mu;\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(\frac{-(x_i-\mu)^2}{2\sigma^2}\right)$$

$$\Pr(\mathcal{D}|\mu) = \left(\frac{1}{(2\pi)^{\frac{m}{2}}(\sigma^{\mathbf{2}})^m}\right) exp\left(\frac{-1}{2\sigma^{\mathbf{2}}}\sum_{i=1}^{m}(x_i-\mu)^2\right)$$

Note the positions of $\sigma^{\mathbf{2}}$ in the likelihood above. In order to make the posterior of the same form as the prior, we should make the prior jell seamlessly with the likelihood because $\Pr(\theta|\mathcal{D}) \propto \Pr(\mathcal{D}|\theta)\Pr(\theta)$, where we have $\theta := \sigma^{\mathbf{2}}$. This could mean

$$\Pr(\theta) \propto \frac{1}{\theta^A} exp(\frac{-B}{\theta})$$

One can normalize this distribution to find the proportionality constant.

It is ok if the students get so far in suggesting the prior. It is also ok if the students miss somehow land up only with $\Pr(\theta) \propto \frac{1}{\theta} exp(\frac{-B}{\theta}))$

Part 2 of the question:

$$\Pr(x|D) = \int_{\theta} \Pr(x|\theta)\Pr(\theta|D)d\theta$$

Substituting,

$$\Pr(x|D) = \int_{\sigma^{\mathbf{2}}} \Pr(x|\sigma^{\mathbf{2}})\Pr(\sigma^{\mathbf{2}}|D)d\sigma^{\mathbf{2}} = \int_{\sigma^{\mathbf{2}}} \Pr(x|\sigma^{\mathbf{2}})\Pr(\sigma^{\mathbf{2}}|D)d\sigma^{\mathbf{2}}$$

We can substitute and leave the integral as it is. An approximation is to use the MAP or Bayes estimate in place of integration and

$\Pr(x|D) \approx \Pr(x|\sigma^{\mathbf{2}}_{\mathbf{MAP}})\Pr(\sigma^{\mathbf{2}}_{\mathbf{MAP}}|D)$

No need to give marks to what follows: This is called an inverse-gamma distribution.

$$p(\theta) = \frac{B^{A-1}}{\Gamma(A-1)}\frac{1}{\theta^A} exp(\frac{-B}{\theta})$$

The posterior is also an inverse-gamma distribution with

$$A' = A + n/2$$

$$B' = B + \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^2$$

Marginal likelihood

$$p(\mathcal{D}) = \frac{\frac{1}{\sqrt[2]{2\pi\theta}} exp(\sum_{i=1}^{n}\frac{-(x_i-\mu)^2}{2\theta})\frac{B^{A-1}}{\Gamma(A-1)}\frac{1}{\theta^A}exp(\frac{-B}{\theta})}{\frac{B'^{A'-1}}{\Gamma(A'-1)}\frac{1}{\theta^{A'}}exp(\frac{-B'}{\theta})}$$

Posterior Predictive

$$p(x|\mathcal{D}) = \frac{p(x,\mathcal{D})}{p(\mathcal{D})}$$

Use $\mathcal{D}' = (\mathcal{D}, x)$ to compute $p(\mathcal{D}')$ and substitute back to get

$$p(x|\mathcal{D}) = t_{2A'}(x|\mu, \theta = \frac{B'}{A'})$$

Problem 6. Consider a linear model of the form

$$y(x, \mathbf{w}) = w_0 + \sum_{i=1}^{D} w_i x_i$$

together with a sum-of-squares error function of the form

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

Now suppose that Gaussian noise ϵ_i with zero mean and variance σ^2 is added independently to each of the input variables x_i. By making use of $\mathbb{E}[\epsilon_i] = 0$ and $\mathbb{E}[\epsilon_i \epsilon_j] = \delta_{ij}\sigma^2$ (i.e. $\mathbb{E}[\epsilon_i \epsilon_j] = \sigma^2$ when $i = j$), show that minimizing E_D averaged over the noise distribution is equivalent to minimizing the sum-of-squares error for noise-free input variables with the addition of a weight-decay regularization term, in which the bias parameter w_0 is omitted from the regularizer. (Problem 3.4 from Bishop, PRML)

Solution:
After adding Gaussian noise to each of the input variables, let:

$$\bar{y}_n = w_0 + \sum_{i=1}^{D} w_i (x_{ni} + \epsilon_{ni}) = y_n + \sum_{i=1}^{D} w_i \epsilon_{ni}$$

where $y_n = y(x_n, \mathbf{w})$ and $\epsilon_{ni} \sim \mathcal{N}(0, \sigma^2)$.
The sum-of-squares error function becomes:

$$\begin{aligned} \bar{E} &= \frac{1}{2} \sum_{n=1}^{N} \{\bar{y}_n - t_n\}^2 \\ &= \frac{1}{2} \sum_{n=1}^{N} \left\{ y_n^2 + 2y_n \sum_{i=1}^{D} w_i \epsilon_{ni} + \left(\sum_{i=1}^{D} w_i \epsilon_{ni} \right)^2 - 2t_n y_n - 2t_n \sum_{i=1}^{D} w_i \epsilon_{ni} + t_n^2 \right\} \end{aligned} \tag{3}$$

Taking the expectation of $\bar{E}$ under ϵ_{ni}, the second and fifth terms in Equation 3 disappear (because $\mathbb{E}[\epsilon_{ni}] = 0$) and for the third term we get $\mathbb{E}\left[\left(\sum_{i=1}^{D} w_i \epsilon_{ni} \right)^2 \right] = \sum_{i=1}^{D} w_i^2 \sigma^2$ (because ϵ_{ni} are all independent with variance σ^2). Thus we have:

$$\mathbb{E}[\bar{E}] = E_D + \frac{1}{2} \sum_{i=1}^{D} w_i^2 \sigma^2$$

which is what was asked for.

Tutorial 3

CS 337 Artificial Intelligence & Machine Learning, Autumn 2019

Week 3, August, 2019

Problem 1. Given a set of data points $\{x_n\}$, we can define the convex hull to be the set of all points x given by

$$x = \sum_n \alpha_n x_n$$

where $\alpha_n \geq 0$ and $\sum_n \alpha_n = 1$. Consider a second set of points $\{y_m\}$ together with their corresponding convex hull. By definition, the two sets of points will be linearly separable if there exists a vector $\hat{w}$ and a scalar w_0 such that $\hat{w} \cdot x_n + w_0 > 0$ for all x_n and $\hat{w} \cdot x_n + w_0 < 0$ for all y_m. Show that if their convex hulls intersect, the two sets of points cannot be linearly separable, and conversely that if they are linearly separable, their convex hulls do not intersect. [From PRML 2006, Chapter 4.]

Problem 2. Let X have a uniform distribution over integers in an interval $[0, \theta)$:

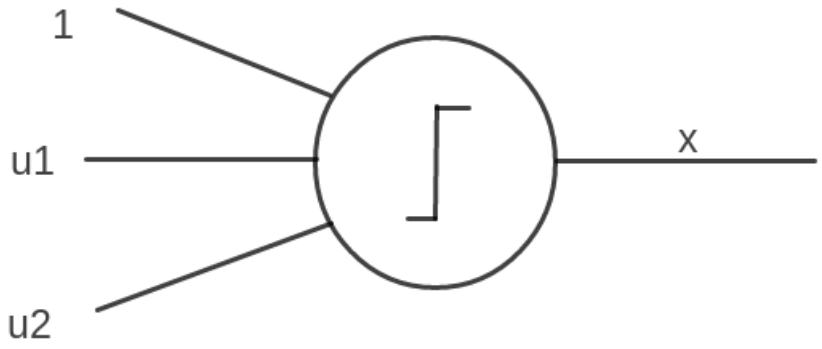
$$p(X = x; \theta) = \begin{cases} \frac{1}{\theta} & \text{if } 0 \leq x < \theta \\ 0 & \text{otherwise} \end{cases}$$

Suppose n samples $x_1, \ldots, x_n$ are drawn i.i.d based on $p(x; \theta)$. What is the MLE estimate of θ?

Problem 3. Computing power of perceptrons. Perceptrons can only separate Linearly separable data as discussed in class. Given n variables we can have 2^{2^n} boolean functions, but not all of these can be represented by a perceptron. For example when n=2 the XOR and XNOR cannot be represented by a perceptron. Given n boolean variables how many of 2^{2^n} boolean functions can be represented by a perceptron?

Problem 4. Consider a perceptron for which $u \in R^2$ and

$$f(a) = \begin{cases} 1 & a > 0 \\ 0 & a = 0 \\ -1 & a < 0 \end{cases}$$



Let the desired output be 1 when elements of class A = {(1,2),(2,4),(3,3),(4,4)} is applied as input and let it be -1 for the class B = {(0,0),(2,3),(3,0),(4,2)}. Let the initial connection weights $w_0(0) = +1, w_1(0) = -2, w_2(0) = +1$ and learning rate be h = 0.5.

This perceptron is to be trained by perceptron convergence procedure, for which the weight update formula is $w(t+1) = w(t) + \eta(y^k - x^k(t))u^k$

1. (a) Mark the elements belonging to class A with x and those belonging to class B with o on input space.
   (b) Draw the line represented by the perceptron considering the initial connection weights w(0).
   (c) Find out the regions for which the perceptron output is +1 and 1
   (d) Which elements of A and B are correctly classified, which elements are misclassified and which are unclassified?
2. If u=(4,4) is applied at input, what will be w(1) ?
3. Repeat a) considering w(1).
4. If u=(4.2) is then applied at input, what will be w(2)?

5. Repeat 1) considering w(2).

6. Do you expect the perceptron convergence procedure to terminate? Why?

Problem 5. In the class, we discussed the probabilistic binary (class) logistic regression classifier. How will you extend logistic regression probabilistic model to multiple (say K) classes? Are their different ways of extending? What is the intuition behind each? Discuss and contrast advantages/disadvantages in each.

Tutorials 4 & 5

CS 337 Artificial Intelligence & Machine Learning, Autumn 2019

September, 2019

Problem 1. Weighted Linear Regression

Consider a data set in which each data point y_i is associated with a weighting factor r_i, so that the sum-square error function becomes

$$\frac{1}{2}\sum_{i=1}^{m} r_i(y_i - w^T\phi(x_i))^2$$

Find an expression for the solution w^* that minimizes this error function. The weights r_i's are known before hand. (Exercise 3.3 of Pattern Recognition and Machine Learning, Christopher Bishop).

Solution: Refer to the solution to problem 2, part 1. The solution to problem 1 is included therein.

Problem 2. Locally Weighted Kernel Regression

In problem 1, we discussed weighted regression. In this problem, we will deal with weighted regression, with the weights obtained using some kernel $K(.,.)$. Given a training set of points $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_n, y_n)\}$, we predict a regression function $f(x') = (\mathbf{w}^\top\phi(x') + b)$ for each test (or query point) x' as follows:

$$(\mathbf{w}', b') = \underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{n} K(x', x_i)\left(y_i - (\mathbf{w}^\top\phi(x_i) + b)\right)^2$$

1. If there is a closed form expression for $(\mathbf{w}', b')$ and therefore for $f(x')$ in terms of the known quantities, derive it.

2. How does this model compare with linear regression and $k-$nearest neighbor regression? What are the relative advantages and disadvantages of this model?

3. In the one dimensional case (that is when $\phi(x) \in \Re$), graphically try and interpret what this regression model would look like, say when $K(.,.)$ is the linear kernel[1].

Solution:

This problem is directly related to problem 1 and herein we present the solution to both the problems

[1]Hint: What would the regression function look like at each training data point?

1. The weighing factor $r_i^{x'}$ of each training data point $(\mathbf{x}_i, y_i)$ is now also a function of the query or test data point $(\mathbf{x}', ?)$, so that we write it as $r_i^{x'} = K(\mathbf{x}', \mathbf{x}_i)$ for $i = 1, \ldots, m$. Let $r_{m+1}^{x'} = 1$ and let R be an $(m+1) \times (m+1)$ diagonal matrix of $r_1^{x'}, r_2^{x'}, \ldots, r_{m+1}^{x'}$.

$$R = \begin{bmatrix} r_1^{x'} & 0 & ... & 0 & \\ 0 & r_2^{x'} & ... & 0 & \\ ... & ... & ... & ... & 1 \\ 0 & 0 & 0 & ... & r_{m+1}^{x'} \end{bmatrix}$$

Further, let

$$\Phi = \begin{bmatrix} \phi_1(x_1) & ... & \phi_p(x_1) & 1 \\ ... & ... & ... & 1 \\ \phi_1(x_m) & ... & \phi_p(x_m) & 1 \end{bmatrix}$$

and

$$\widehat{\mathbf{w}} = \begin{bmatrix} w_1 \\ ... \\ w_p \\ b \end{bmatrix}$$

and

$$\mathbf{y} = \begin{bmatrix} y_1 \\ ... \\ y_m \end{bmatrix}$$

The sum-square error function then becomes

$$\frac{1}{2} \sum_{i=1}^{m} r_i (y_i - (\widehat{\mathbf{w}}^T \phi(x_i) + b))^2 = \frac{1}{2} ||\sqrt{R}\mathbf{y} - \sqrt{R}\Phi\widehat{\mathbf{w}}||_2^2$$

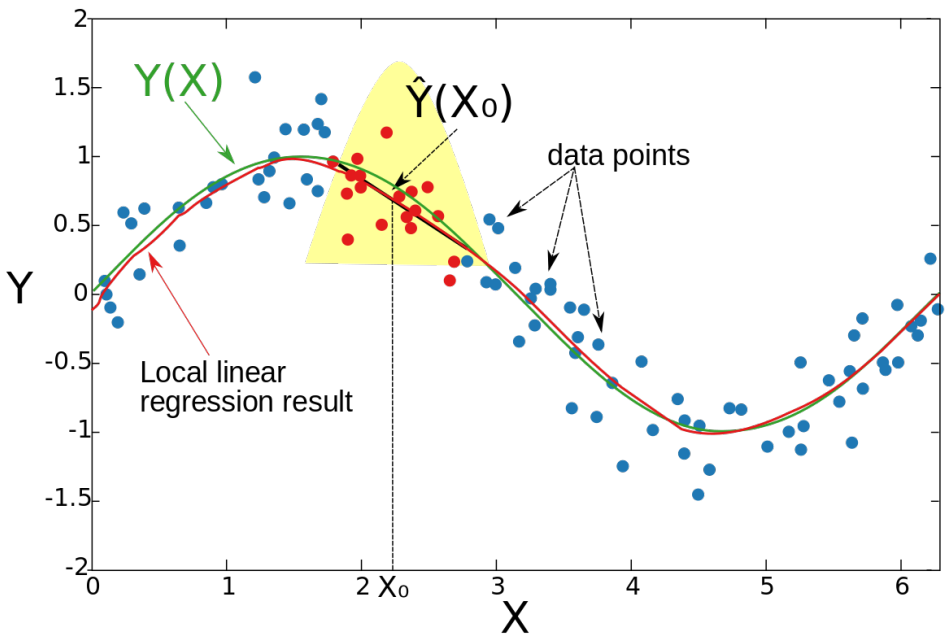
where $\sqrt{R}$ is a diagonal matrix such that each diagonal element of $\sqrt{R}$ is the square root of the corresponding element of R. This is a convex function being minimized (prove this using techniques similar to what we employed for least squares linear regression) and therefore has a global minimum at $\widehat{\mathbf{w}}_*^{x'}$ where the gradient must become 0. (again work out the steps using techniques similar to what we employed for least squares linear regression). The expression for the solution $\widehat{\mathbf{w}}^*$ that minimizes this error function is therefore

$$\widehat{\mathbf{w}}_*^{x'} = (\Phi^T R \Phi)^{-1} \Phi^T R \mathbf{y}$$

2. Let us refer to this model as local linear regression (Section 6.1.1 of Tibshi's book).

   As compared to linear regression, local linear regression gives more importance to points in $\mathcal{D}$ that are closer/similar to $\mathbf{x}'$ and less importance to points that are less similar. Thus, this method can be important if the regression curve is supposed to take different shapes or different parameters in different parts of the space. For example, in two different regions, the ideal regression curve might be linear in each but with different parameters. In this sense, local linear regression comes close to k-nearest neighbor. But unlike k-nearest neighbor, local linear regression gives you a smooth solution since contribution for regression at a point comes from all data points (in proportion to their closeness) and not just the k closest points.

3. Taking clue from the discussion above, one can try and plot this regression curve.



5

Consider a data set in which each data point y_i is associated with a weighting factor r_i, so that the sum-square error function becomes

$$\frac{1}{2}\sum_{i=1}^{m} r_i(y_i - w^T\phi(x_i))^2$$

Find an expression for the solution w^* that minimizes this error function. The weights r_i's are known before hand. (Exercise 3.3 of Pattern Recognition and Machine Learning, Christopher Bishop)

Problem 3. Redoing the Kernel Ridge Regression Problem: Let $\mathcal{D} = \langle(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\rangle$ such that each $y_j \in \Re$. Let $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_n(\mathbf{x})]$ be a vector of basis functions. Consider the linear regression function $f(\mathbf{x}) = \phi^T(\mathbf{x})\mathbf{w}$ with $\mathbf{w}$ obtained either as a least squares or ridge regression estimate. Show that, using either of these estimates for $\mathbf{w}$, the regression function can be written in the (so-called *kernelized*) form $f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}, \mathbf{x}_i) y_i$ where $K(\mathbf{x}, \mathbf{x}_i) = \phi^T(\mathbf{x})\phi(\mathbf{x}_i)$ is a function of $\mathbf{x}$ and $\mathbf{x}_i$ only and independent of any of the $\mathbf{y}_i$'s and $\mathbf{x}_j$ for all $j \neq i$. Each α_i can be a function of the entire dataset $\mathcal{D}$.

Hint: Use the following Matrix Identity that holds for any matrices P, B and R with compatible dimensions such that R and $BPB^T + R$ are invertible:

$$(P^{-1} + B^T R^{-1} B)^{-1} B^T R^{-1} = PB^T(BPB^T + R)^{-1}$$

Solution: The solution to linear (set $\lambda = 0$) and ridge regression can be written as $\mathbf{w} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T\mathbf{y}$ where

- Recall for Ridge Regression: $\mathbf{w} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T y$, where,

$$\mathbf{y} = \begin{bmatrix} y_1 \\ ... \\ y_m \end{bmatrix}$$

$$\Phi = \begin{bmatrix} \phi_1(\mathbf{x}_1) & ... & \phi_p(\mathbf{x}_1) \\ ... & ... & ... \\ \phi_1(\mathbf{x}_m) & ... & \phi_p(\mathbf{x}_m) \end{bmatrix}$$

- **Please note the difference between Φ and $\phi(\mathbf{x})$**

$$\phi(\mathbf{x}_j) = \begin{bmatrix} \phi_1(\mathbf{x}_j) \\ ... \\ \phi_p(\mathbf{x}_j) \end{bmatrix}$$

  Then, the regression function will be

$$f(\mathbf{x}) = \phi^T(\mathbf{x})\mathbf{w} = \phi^T(\mathbf{x})(\Phi^T\Phi + \lambda I)^{-1}\Phi^T\mathbf{y}$$

- $\phi^T(\mathbf{x}_i)\phi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$
- $\left(\Phi^T\Phi\right)_{ij} = \sum_{k=1}^{m} \phi_i(\mathbf{x}_k)\phi_j(\mathbf{x}_k)$
- $\left(\Phi\Phi^T\right)_{ij} = \sum_{k=1}^{p} \phi_k(\mathbf{x}_i)\phi_k(\mathbf{x}_j) = \phi^T(x_i)\phi(x_j) = K(\mathbf{x}_i, \mathbf{x}_j)$

Kernelizing Ridge Regression

- Given $\mathbf{w} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T\mathbf{y}$ and using the identity $(P^{-1} + B^TR^{-1}B)^{-1}B^TR^{-1} = PB^T(BPB^T + R)^{-1}$
  - $\Rightarrow$ by setting $R = I$, $P = \frac{1}{\lambda}I$ and $B = \Phi$,
  - $\Rightarrow \mathbf{w} = \Phi^T(\Phi\Phi^T + \lambda I)^{-1}\mathbf{y} = \sum_{i=1}^{m} \alpha_i\phi(\mathbf{x}_i)$ where $\alpha_i = \left((\Phi\Phi^T + \lambda I)^{-1}\mathbf{y}\right)_i$
  - $\Rightarrow$ the final decision function $f(\mathbf{x}) = \phi^T(\mathbf{x})\mathbf{w} = \sum_{i=1}^{m} \alpha_i\phi^T(\mathbf{x})\phi(\mathbf{x}_i)$

The Kernel function in Ridge Regression

- We call $\phi^\top(x_1)\phi(x_2)$ a **kernel function**:
  $K(x_1, x_2) = \phi^\top(x_1)\phi(x_2)$
- The preceding expression for decision function becomes $f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$
  where $\alpha_i = (([K(\mathbf{x}_i, \mathbf{x}_j)] + \lambda I)^{-1}\mathbf{y})_i$

Problem 4. Equivalent Kernelized Representation (Post-midsem):

Throughout this question, let $0 < p < 1$. Consider a data set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ of m points and a feature function $\phi(\mathbf{x}) \subseteq \Re^n$. Let $f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$. You have seen linear regression with various regularizations of the form:

$$\sum_{i=1}^{m} \left(y_i - \phi^T(\mathbf{x}_i)\mathbf{w}\right)^2 + \lambda \left(\sum_{j=1}^{n} |w_j|^p\right) \tag{1}$$

Now consider a somewhat complementary setting:

$$\sum_{i=1}^{m} \left(y_i - \phi^T(\mathbf{x}_i)\mathbf{w}\right)^p + \lambda \left(\sum_{j=1}^{n} |w_j|^2\right) \tag{2}$$

1. Do these forms have an equivalent kernelized representation: $f(\mathbf{x}) = \phi^T(\mathbf{x})\mathbf{w}^* + b = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}, \mathbf{x}^{(i)})$? How would you prove?
2. Contrast the two descriptions for their capabilities.

Solution:

Solution to part (1):

As per the representer theorem, if $f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$ and $K(\mathbf{x}', \mathbf{x}) = \phi^T(\mathbf{x})\phi(\mathbf{x}')$ then the solution $\mathbf{w}^* \in \Re^n$ to the following problem

$$(\mathbf{w}^*, b^*) = \underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{m} \mathbf{E}\left(f\left(\mathbf{x}^{(i)}\right), y^{(i)}\right) + \Omega(||\mathbf{w}||_2)$$

can be always written as $\phi^T(\mathbf{x})\mathbf{w}^* + b = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}, \mathbf{x}^{(i)})$, provided $\Omega(||\mathbf{w}||_2)$ is a monotonically increasing function of $||\mathbf{w}||_2$. **Optionally recall:** $\Re^n$ is the Hilbert space and $K(., \mathbf{x}) : \mathcal{X} \to \Re$ is the **Reproducing (RKHS) Kernel**

(it is ok if the student does not mention Hilbert space etc).

Thus, it is obvious that the errors of both (5) and (6) linearly decompose across all the examples as expected in the Representer theorem, the regularizer in (5) cannot be written as a monotonically increasing function of $||\mathbf{w}||_2$ which is possible in the case of (6). Thus, only (6) has an equivalent kernelized representation whereas (5) does not have one.

Solution to part (2):

Whereas (5) gives relatively sparser $\mathbf{w}$ than (6), (6) takes care of outliers better than (5) by having a slower rate of growth with respect to the error (recall this discussion from Tutorial 1).

Problem 5. More on Kernel Perceptron:

Recall the proof for convergence of the perceptron update algorithm. Now can this proof be extended to the kernel perceptron?

Recall that Kernelized perceptron is specified as:

$$f(x) = sign\left(\sum_i \alpha_i^* y_i K(x, x_i) + b^*\right)$$

The perceptron update algorithm for the Kernelized version is:

- INITIALIZE: α=zeroes()
- REPEAT: for $< x_i, y_i >$
  - If $sign\left(\sum_j \alpha_j y_j K(x_j, x_j) + b\right) \neq y_i$
  - then, $\alpha_j = \alpha_j + 1$

Solution: Yes, in fact kernel perceptron can be derived from the perceptron update rule as follows:

$$f(x) = sign\left((w^*)^T \phi(x)\right) = sign\left(\sum_i \alpha_i^* y_i K(x, x_i) + b^*\right)$$

- INITIALIZE: $w = [0, 0, \ldots, 0, 1] \Rightarrow f(x) = sign\left((w)^T \phi(x)\right) = sign\left(\sum_i \alpha_i y_i K(x, x_i) + b\right)$ with $\alpha_i = 0$ and $b = 1$
  Note: $\phi^T(\widehat{x})\phi(x)\widehat{y} = \widehat{y}K(\widehat{x}, x) + \widehat{y}$
- REPEAT: for each $< \widehat{x}, \widehat{y} >$
  - If $\widehat{y} w^T \phi(\widehat{x}) < 0$
    $\Rightarrow f(\widehat{x}) = sign\left((w)^T \phi(\widehat{x})\right) = sign\left(\sum_i \alpha_i y_i K(\widehat{x}, x_i) + b\right) \neq \widehat{y}$
  - then, $w' = w + \Phi(\widehat{x}).\widehat{y}$
    $\Rightarrow f(x) = sign\left((w')^T \phi(x)\right) = sign\left(\sum_i (\alpha_i y_i K(x, x_i) + \phi^T(\widehat{x})\phi(x)\widehat{y}) + b\right) =$
    $sign\left(\sum_i \alpha_i' y_i K(x, x_i) + b'\right)$ where $\alpha_i' = \alpha_i$ for all i except that $\alpha_{\widehat{x}}' = \alpha_{\widehat{x}} + 1$ and $b' = b + \widehat{y}$
  - endif

Thus, $f(x) = sign\left((w^*)^T \phi(x)\right) = sign\left(\sum_i^* \alpha_i y_i K(x, x_i)\right)$

Problem 6. Kernel Logistic Regression: Intuition (and optional Rigorous proof)
Recall the Regularized (Logistic) Cross-Entropy Loss function (minimized wrt $\mathbf{w} \in \Re^p$):

$$E(\mathbf{w}) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} \log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] + \frac{\lambda}{2m}\|\mathbf{w}\|_2^2 \quad (3)$$

Now intuitively show that minimizing the following dual kernelized objective[2] (minimized wrt $\alpha \in \Re^m$) is equivalent to minimizing the regularized cross-entropy loss function:

$$E_D(\alpha) = \left[\sum_{i=1}^{m}\left(\sum_{j=1}^{m} -y^{(i)} K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)\alpha_j + \frac{\lambda}{2}\alpha_i K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)\alpha_j\right) + \log\left(1 + \exp\sum_{j=1}^{m}\alpha_j K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)\right)\right] \quad (4)$$

[2] http://perso.telecom-paristech.fr/~clemenco/Projets_ENPC_files/kernel-log-regression-svm-boosting.pdf

where, decision function $f_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1+\exp\left(\sum_{\mathbf{j=1}}^{\mathbf{m}} \alpha_{\mathbf{j}} \mathbf{K}\left(\mathbf{x}, \mathbf{x}^{(\mathbf{j})}\right)\right)}$ How would you prove this very rigorously (**optional**)?

Solution:

We will prove this result and in the process, also motivate (and somewhat prove - **optional**) the more general **Representer Theorem** atleast for Logistic Regression

1. **Some preliminary steps:**

   Recall another form of the regularized cross entropy equivalent to (3)

$$E(\mathbf{w}) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - \log\left(1+\exp\left(\mathbf{w}^T\phi(\mathbf{x}^{(i)})\right)\right)\right)\right] + \frac{\lambda}{2m}||\mathbf{w}||^2 \tag{5}$$

   First of all, we will drop the common term $\frac{1}{m}$ from the primal optimization problem and equivalently minimize the unscaled version

$$E(w) = -\left[\sum_{i=1}^{m}\left(y^{(i)}\log f_w\left(\mathbf{x}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \tag{6}$$

$$\nabla E(\mathbf{w}) = \left[\sum_{i=1}^{m}\left(y^{(i)}\nabla\log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1-y^{(i)}\right)\nabla\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] + \lambda\mathbf{w} \tag{7}$$

2. $\nabla\log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) = \phi(\mathbf{x}^{(i)})e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}\left(\frac{1}{1+e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}}\right)^2$ and

   $\nabla\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right) = -\phi(\mathbf{x}^{(i)})\left(\frac{1}{1+e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}}\right)^2$

3. $\Rightarrow$

$$\nabla E(\mathbf{w}) = \left[\sum_{i=1}^{m}\left(y^{(i)} - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\phi(\mathbf{x}^{(i)})\right] + \lambda\mathbf{w} \tag{8}$$

   At optimality, a necessary condition is that $\nabla E(\mathbf{w}) = 0$ and therefore,

$$\mathbf{w} = \frac{1}{\lambda}\left[\sum_{i=1}^{m}\left(y^{(i)} - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\phi(\mathbf{x}^{(i)})\right] \tag{9}$$

4. **The main idea:**

In summary, the main optimization objective is

$$E(\mathbf{w}) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - \log\left(1+\exp\left(\mathbf{w}^T\phi(\mathbf{x}^{(i)})\right)\right)\right)\right] + \frac{\lambda}{2m}||\mathbf{w}||^2 \quad (10)$$

and an expression for $\mathbf{w}$ at optimality is

$$\mathbf{w} = \frac{1}{\lambda}\left[\sum_{i=1}^{m}\left(y^{(i)} - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\phi(\mathbf{x}^{(i)})\right] \quad (11)$$

5. Recall from the representer theorem that in the optimization problem (10), $\mathbf{w^T}\phi(\mathbf{x^{(i)}})$ can be equivalently expressed as $\sum_{\mathbf{j=1}}^{\mathbf{m}}\alpha_\mathbf{j}\mathbf{K}\left(\mathbf{x},\mathbf{x^{(j)}}\right)$, as a result of which we will obtain the following terms of (4):

$$\left[\sum_{i=1}^{m}\left(\sum_{j=1}^{m} -y^{(i)}K\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)\alpha_j\right) + \log\left(1+\exp\sum_{j=1}^{m}\alpha_j K\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)\right)\right] \quad (12)$$

Substituting from (11) into one of the $\mathbf{w}$ in $\frac{\lambda}{2m}||\mathbf{w}||^2 = \frac{\lambda}{2m}\mathbf{w}^T\mathbf{w}$ term of (10) we will get the regularizer into the form

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\frac{\lambda}{2}\alpha_i K\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)\alpha_j$$

which helps form the remaining term of (4)

EXTRA and COMPLETELY OPTIONAL: Proof of Representer Theorem specifically for Logistic Regression:

To completely prove this specific case of KLR, let $\mathcal{X}$ be the space of examples such that $\left\{\mathbf{x}^{(1)},\mathbf{x}^{(2)},\ldots,\mathbf{x}^{(m)}\right\} \subseteq \mathcal{X}$ and for any $\mathbf{x} \in \mathcal{X}$, $K(.,\mathbf{x}) : \mathcal{X} \to \Re$ be a function such that $K(\mathbf{x}',\mathbf{x}) = \phi^T(\mathbf{x})\phi(\mathbf{x}')$. Recall that $\phi(\mathbf{x}) \in \Re^n$ and

$$f_{\mathbf{w}}(\mathbf{x}) = p(Y=1|\phi(\mathbf{x})) = \frac{1}{1+\exp\left(-\mathbf{w^T}\phi(\mathbf{x})\right)}$$

For the rest of the discussion, we are interested in viewing $-\mathbf{w}^T\phi(\mathbf{x})$ as a function $h(\mathbf{x})$

$$f_{\mathbf{w}}(\mathbf{x}) = p(Y=1|\phi(\mathbf{x})) = \frac{1}{1+\exp\left(\mathbf{h}(\mathbf{x})\right)}$$

We will prove that for the optimization problem (10), $\mathbf{h}(\mathbf{x})$ can be equivalently expressed as $\sum_{\mathbf{j=1}}^{\mathbf{m}}\alpha_\mathbf{j}\mathbf{K}\left(\mathbf{x},\mathbf{x^{(j)}}\right)$, as a result of which we will obtain the following terms of (4):

$$\left[\sum_{i=1}^{m}\left(\sum_{j=1}^{m} -y^{(i)}K\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)\alpha_j\right) + \log\left(1+\sum_{j=1}^{m}\alpha_j K\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)\right)\right] \quad (13)$$

Substituting (11) into $\frac{\lambda}{2m}||\mathbf{w}||^2$ term of (10) we will get the regularizer into the form

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\frac{\lambda}{2}\alpha_i K\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)\alpha_j$$

which forms the remaining term of (4)

1. Consider the set of functions $\mathcal{K}=\{K(.,\mathbf{x})\mid \mathbf{x}\in\mathcal{X}\}$ and let $\mathcal{H}$ be the set of all functions that are **finite** linear combinations of functions in $\mathcal{K}$. That is, any function $h\in\mathcal{H}$ can be written as $\mathbf{h(.)}=\sum_{t=1}^{T}\alpha_t K(.,\mathbf{x}_t)$ for some T and $\mathbf{x}_t\in\mathcal{X},\alpha_t\in\Re$. One can easily verify that $\mathcal{H}$ is a vector space[3]

   Note that, in the special case when $f(\mathbf{x}')=K(\mathbf{x}',\mathbf{x})$, then $T=m$ and

   $$f(\mathbf{x}')=K(\mathbf{x}',\mathbf{x})=\sum_{i=1}^{n}\phi_i(\mathbf{x}')K(\mathbf{e}_i,\mathbf{x})$$

   where $\mathbf{e}_i$ is such that $\phi(\mathbf{e}_i)=\mathbf{u}_i\in\Re^n$, the unit vector along the i^{th} direction.

   Also, by the same token, if $\mathbf{w}\in\Re^n$ is in the search space of the regularized cross-entropy loss function (3), then

   $$\phi^{\mathbf{T}}(\mathbf{x}')\mathbf{w}=\sum_{i=1}^{n}w_i K(\mathbf{e}_i,\mathbf{x})$$

   Thus, the solution to (3) is an $h\in\mathcal{H}$.

2. **Inner Product over $\mathcal{H}$:** For any $g(.)=\sum_{t=1}^{S}\beta_s K(.,\mathbf{x}'_s)\in\mathcal{H}$ and $h(.)=\sum_{t=1}^{T}\alpha_t K(.,\mathbf{x}_t)\in\mathcal{H}$, define the inner product[4]

   $$\langle h,g\rangle=\sum_{s=1}^{S}\beta_s\sum_{t=1}^{T}\alpha_t K(\mathbf{x}'_s,\mathbf{x}_t) \tag{14}$$

   Further simplifying (14),

   $$\langle h,g\rangle=\sum_{s=1}^{S}\beta_s\sum_{t=1}^{T}\alpha_t K(\mathbf{x}'_s,\mathbf{x}_t)=\sum_{s=1}^{S}\beta_s f(\mathbf{x}_s) \tag{15}$$

   One immediately observes that in the special case that $g()=K(.,\mathbf{x})$,

   $$\langle h,K(.,\mathbf{x})\rangle=h(\mathbf{x}) \tag{16}$$

[3]Try it yourself. Prove that $\mathcal{H}$ is closed under vector addition and (real) scalar multiplication.

[4]Again, you can verify that $\langle f,g\rangle$ is indeed an inner product following properties such as symmetry, linearity in the first argument and positive-definiteness: `https://en.wikipedia.org/wiki/Inner_product_space`

3. **Orthogonal Decomposition:** Since $\left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}\right\} \subseteq \mathcal{X}$ and $\mathcal{K} = \{K(., \mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}$ with $\mathcal{H}$ being the set of all finite linear combinations of function in $\mathcal{K}$, we also have that

$$lin\_span\left\{K(., \mathbf{x}^{(1)}), K(.\mathbf{x}^{(2)}), \ldots, K(., \mathbf{x}^{(m)})\right\} \subseteq \mathcal{H}$$

Thus, we can use orthogonal projection to decompose any $h \in \mathcal{H}$ into a sum of two functions, one lying in $lin\_span\left\{K(., \mathbf{x}^{(1)}), K(.\mathbf{x}^{(2)}), \ldots, K(., \mathbf{x}^{(m)})\right\}$, and the other lying in the orthogonal complement:

$$h = h^{\parallel} + h^{\perp} = \sum_{i=1}^{m} \alpha_i K(., \mathbf{x}^{(i)}) + h^{\perp} \tag{17}$$

where $\langle K(., \mathbf{x}^{(i)}), h^{\perp} \rangle = 0$, for each $i = [1..m]$.

For a specific training point $\mathbf{x}^{(j)}$, substituting from (17) into (16) for any $h \in \mathcal{H}$, using the fact that $\langle K(., \mathbf{x}^{(i)}), h^{\perp} \rangle = 0$

$$h(\mathbf{x}^{(j)}) = \langle \sum_{i=1}^{m} \alpha_i K(., \mathbf{x}^{(i)}) + h^{\perp}, K(., \mathbf{x}^{(j)}) \rangle = \sum_{i=1}^{m} \alpha_i \langle K(., \mathbf{x}^{(i)}), K(., \mathbf{x}^{(j)}) \rangle = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \tag{18}$$

which we observe is independent of $h^{\perp}$.

4. **Analysis of the Regularized Cross-Entropy Logistic Loss:**

The Regularized Cross-Entropy Logistic Loss (10), has two parts (after ignoring the common $\frac{1}{m}$ factor), *viz.*, the **empirical risk**

$$-\left[\sum_{i=1}^{m} \left(y^{(i)} \mathbf{w}^T \phi(\mathbf{x}^{(i)}) - \log\left(1 + \exp\left(\mathbf{w}^T \mathbf{x}^{(i)}\right)\right)\right)\right] \tag{19}$$

Since the **empirical risk** in (19) is only a function of $h(\mathbf{x}^{(i)}) = \mathbf{w}^T \phi(\mathbf{x}^{(i)})$ for $i = [1..m]$, based on (18) we note that the value of the **empirical risk** in (19) will therefore be independent of $h^{\perp}$ and therefore one only needs to equivalently solve the following **empirical risk** by substituting from (18) *i.e.*, $h(\mathbf{x}^{(j)}) = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$:

$$\left[\sum_{\mathbf{i=1}}^{\mathbf{m}} \left(\sum_{\mathbf{j=1}}^{\mathbf{m}} -\mathbf{y^{(i)}} \mathbf{K}\left(\mathbf{x^{(i)}}, \mathbf{x^{(j)}}\right) \alpha_{\mathbf{j}}\right) + \log\left(\mathbf{1} + \sum_{\mathbf{j=1}}^{\mathbf{m}} \alpha_{\mathbf{j}} \mathbf{K}\left(\mathbf{x^{(i)}}, \mathbf{x^{(j)}}\right)\right)\right]$$

5. **Safe with Regularizer?**

Consider the regularizer function $||\mathbf{w}||_2^2$ which is a strictly monotonically increasing function of $||\mathbf{w}||$. Substituting $\mathbf{w} = \frac{1}{\lambda}\left[\sum_{i=1}^{m} \left(y^{(i)} - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right) \phi(\mathbf{x}^{(i)})\right]$ from (9), one can view $\Omega(||h||)$ as a strictly monotonic function of $||h||$.

$$\Omega(||h||) = \Omega\left(||\sum_{i=1}^{m} \alpha_i K(., \mathbf{x}^{(i)}) + h^{\perp}||\right) = \Omega\left(\sqrt{||\sum_{i=1}^{m} \alpha_i K(., \mathbf{x}^{(i)})||^2 + ||h^{\perp}||^2}\right)$$

and therefore,

$$\Omega(||h||) = \Omega\left(\sqrt{||\sum_{i=1}^{m} \alpha_i K(., \mathbf{x}^{(i)})||^2 + ||h^{\perp}||^2}\right) \geq \Omega\left(\sqrt{||\sum_{i=1}^{m} \alpha_i K(., \mathbf{x}^{(i)})||^2}\right)$$

That is, setting $h^{\perp} = 0$ does not affect the first term of (10) while strictly increasing the second term. That is, any minimizer must have optimal $h^*(.)$ with $h^{\perp} = 0$. That is,

$$h(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}^{(i)}, \mathbf{x})$$

Problem 7. Effect of increasing λ in Ridge Regression

Consider the ridge regression problem

$$\widehat{\mathbf{w}}_{ridge} = \underset{\mathbf{w}}{\operatorname{argmin}} \; ||\Phi^T \mathbf{w} - \mathbf{y}||^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

for any $\lambda \geq 0$. Recall the purpose for which the regularization term $\frac{\lambda}{2}||\mathbf{w}||^2$ was introduced in linear regression.

Let $\mathbf{w}_1$ be the optimal solution to this problem when $\lambda = \lambda_1$ and let $\mathbf{w}_2$ be the optimal solution to this problem when $\lambda = \lambda_2$. Let $\lambda_2 < \lambda_1$.

Which of the following statements is correct?

1. $||\mathbf{w}_2|| \leq ||\mathbf{w}_1||$ (that is, $||\widehat{\mathbf{w}}_{ridge}||$ will not increase as λ decreases towards 0).
2. $||\mathbf{w}_2|| \geq ||\mathbf{w}_1||$ (that is, $||\widehat{\mathbf{w}}_{ridge}||$ will not decrease as λ decreases towards 0).
3. none of these

Prove your answer. Why does increase in λ reduce the curvature of the solution obtained via ridge regression?

SOLUTION:

Let $\lambda_2 < \lambda_1$ and $f_\lambda(\mathbf{w}) = ||\Phi^T \mathbf{w} - \mathbf{y}||^2 + \frac{\lambda}{2}||\mathbf{w}||^2$ Then:

$$f_{\lambda_1}(\mathbf{w}_2) + f_{\lambda_2}(\mathbf{w}_1) \geq f_{\lambda_1}(\mathbf{w}_1) + f_{\lambda_2}(\mathbf{w}_2)$$

i.e,

$$\lambda_1||\mathbf{w}_2||^2 + \lambda_2||\mathbf{w}_1||^2 \geq \lambda_1||\mathbf{w}_1||^2 + \lambda_2||\mathbf{w}_2||^2$$

i.e

$$(\lambda_1 - \lambda_2)||\mathbf{w}_2||^2 \geq (\lambda_1 - \lambda_2)||\mathbf{w}_1||^2$$

and since $\lambda_2 < \lambda_1$

$$||\mathbf{w}_2||^2 \geq ||\mathbf{w}_1||^2$$

That is, $||\mathbf{w}_2||$ increases as λ decreases

Look at the countours of the objective $||A\mathbf{x}-\mathbf{b}||^2$. The larger is the ratio $\frac{\lambda_{max}(A)}{\lambda_{min}(A)}$, the more skewed are the level curves and more is the time gradient descent will take for convergence. Thus, the matrix A with small value of $\frac{\lambda_{max}(A)}{\lambda_{min}(A)}$ is always desirable.

In general, by the Courant-Fischer min-max Theorem, if A and B are two $n \times n$ symmetric matrices, and suppose the k^{th} largest eigenvalue of matrix X is $\lambda_k(X)$, $k = 1, 2, , n$: $\lambda_1(X) \geq \lambda_2(X) \ldots \geq \lambda_n(X)$ then

$\lambda_k(A) + \lambda_n(B) \leq \lambda_k(A + B) \leq \lambda_k(A) + \lambda_1(B)$

Problem 8. Are these Valid Kernels? Consider the space of all possible subsets A of a given fixed set D. Prove/disprove the following functions are valid Kernels:

1. $K(A_1, A_2) = |A_1 \cap A_2|$
2. $K(A_1, A_2) = 2^{|A_1 \cap A_2|}$

where A_1, A_2 are subsets of D and $|B|$ is the cardinality of $|B|$ or the number of elements in B.

Solution:

Solution to part (a):

Construct a feature vector of size equal to the cardinality of A with 0/1 for an element of set being absent/present respectively in a subset. Taking the product of two feature vectors would give the number of common elements because only the 1's corresponding to elements present in both subsets give a multiplicative contribution.

Solution to part (b):

One way is to prove that if $K(A_1, A_2)$ is a Kernel then $2^{K(A_1,A_2)}$ is also a Kernel. This can be done using the power series expansion of 2^x where coefficient of each degree is positive and then invoking that positive polynomials of Kernels are also Kernels.

Other way is to explicitly come up with a feature vector representation for $2^{|A_1 \cap A_2|}$. Consider feature vector of size $2^{|A|}$ defined by mapping $\phi(A_1)$ where A_1 is a subset of A and each component is representative of a possible subset U of A with entry either 0 or 1 for some subset A_1 as follows:

$$\phi_U(A_1) = 1, \text{if } U \subseteq A_1$$

$$\phi_U(A_1) = 0, \text{otherwise}$$

Tutorial 6

Thursday 10th October, 2019

1 Principal Component Analysis

Let $\mathbf{X}$ be a random vector and $\Gamma = \mathbf{E}[(\mathbf{X} - \mathbf{E}[\mathbf{X}])(\mathbf{X} - \mathbf{E}[\mathbf{X}])^T]$ its covariance matrix. Let $\mathbf{e}_1, \ldots, \mathbf{e}_n$ be the n (normalized) eigenvectors of Γ.

- The n principal components of $\mathbf{X}$ are said to be $\mathbf{e}_1^T\mathbf{X}$, $\mathbf{e}_2^T\mathbf{X}$, ..., $\mathbf{e}_n^T\mathbf{X}$. See `https://arxiv.org/abs/1804.10253`.
- Let $p(X_1) = \mathcal{N}(0, 1)$ and $p(X_2) = \mathcal{N}(0, 1)$ and $cov(X_1, X_2) = \theta$. Find all the principal components of the random vector $\mathbf{X} = [X_1, X_2]^T$.

  Solution:

  1. We first note that the 2×2 matrix
  $$\Gamma = \begin{bmatrix} 1 & \theta \\ \theta & 1 \end{bmatrix}$$
  2. To compute the eigenvalues of Γ from the characteristic polynomial:
  $$\left| \begin{bmatrix} 1-\lambda & \theta \\ \theta & 1-\lambda \end{bmatrix} \right| = (1-\lambda)^2 - \theta^2 = 0$$
  $\Rightarrow \lambda_1 = 1 + \theta$ and $\lambda_2 = 1 - \theta$ are the two solutions/eigenvalues.
  3. For eigenvalue λ_1 we find its eigenvector v_1:
  $$\begin{bmatrix} 1 & \theta \\ \theta & 1 \end{bmatrix} v_1 = (1+\theta)v_1$$
  which is easily solvable as
  $$v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$
  or its normalized version
  $$v_1 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

4. Similarly, for eigenvalue λ_2 we find its eigenvector v_2:

$$\begin{bmatrix} 1 & \theta \\ \theta & 1 \end{bmatrix} v_2 = (1-\theta)v_2$$

which is easily solvable as

$$v_2 = \begin{bmatrix} 1 \\ -1/ \end{bmatrix}$$

or its normalized version

$$v_2 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

5. Thus, the principal components of $\mathbf{X}$ are $X_1/\sqrt{2} + X_2/\sqrt{2}$ and $X_1/\sqrt{2} - X_2/\sqrt{2}$.

- Now, let $\mathbf{Y} = \mathcal{N}(\mathbf{0}, \Sigma) \in \Re^p$ where $\Sigma = \lambda^2 I_{p\times p} + \alpha^2 ones(p,p)$ for any $\lambda, \alpha \in \Re$. Here, $I_{p\times p}$ is a $p \times p$ identity matrix while $ones(p,p)$ is a $p \times p$ matrix of 1's. Find atleast one principal component of $\mathbf{Y}$.

2 How would you Kernelize PCA?

How would you Kernelize PCA? See Section 14.5.4 of the Tibshirani book posted on moodle.

Solution: Here is a proof sketch, that we also provided in the slides

1. Consider the Singular Value Decomposition of $X = ADB^T$. Then $nS = XX^T = AD^2A^T$ and $X^TX = BD^2B^T$
2. $U = AD$ is the matrix of principal component variables.
3. Consider the kernel/gram matrix $\mathcal{K}$ of the data, in place of $\mathbf{X}^T\mathbf{X}$

$$\mathcal{K} = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & ... & K(\mathbf{x}_1, \mathbf{x}_n) \\ ... & K(\mathbf{x}_i, \mathbf{x}_j) & ... \\ K(\mathbf{x}_m, \mathbf{x}_1) & ... & K(\mathbf{x}_m, \mathbf{x}_m) \end{bmatrix}$$

4. Then, $nS = XX^T = AD^2A^T$ and $\mathcal{K} = X^TX = BD^2B^T$ and the projections XU of our data X onto those components $U = AD$ can be computed from the eigendecomposition of $\mathcal{K}$
5. Do eigenvalue decomposition of $\mathcal{K}$. Take the eigenvectors, $\mathbf{v}_1, \ldots, \mathbf{v}_k$.
6. Kernel-PCA computes not the principal components themselves, but the projections of our data onto those components:

- See Extra & Optional slides on Mercer's Theorem and RKHS Kernel for conditions under which $K(\mathbf{x}_i, \mathbf{x}_j)$ will be a **valid Kernel function**
- For more discussion on Kernel PCA, see Section 14.5.4 of the Tibshirani book posted on moodle.

3 EM Algorithm for Mixture of Gaussians (completely optional

Q: Show that the following algorithm for estimating the mean μ_i, the covariance matrix Σ_i and mixture components π_i for a mixture of Gaussians is an instance of the general EM algorithm

ANSWER:

Initialize $\mu_i^{(0)}$ to different random values and $\Sigma_i^{(0)}$ to I. Now iterate between the following **E Step** and **M Steps**:

E Step:

1. For the posterior $p(z_i \mid \phi(x_j), \mu, \Sigma)$

$$p^{(t+1)}(z_i \mid \phi(x_j), \theta) = \frac{\pi_i \mathcal{N}\left(\phi(x); \mu_i^{(t)}, \Sigma_i^{(t)}\right)}{\sum_{l=1}^{K} \pi_l \mathcal{N}\left(\phi(x); \mu_l^{(t)}, \Sigma_l^{(t)}\right)}$$

M Steps:

1. For the prior π_i

$$\pi_i^{(t+1)} = \frac{1}{n} \sum_{j=1}^{n} p^{(t+1)}(z_i \mid \phi(x_j), \theta)$$

2. For μ_i

$$\mu_i^{(t+1)} = \frac{\sum_{j=1}^{n} p^{(t+1)}(z_i \mid \phi(x_j), \theta) \phi(x_j)}{\sum_{j=1}^{n} p^{(t+1)}(z_i \mid \phi(x_j), \theta)}$$

3. For Σ_i

$$\Sigma_i^{(t+1)} = \frac{\sum_{j=1}^{n} p^{(t+1)}(z_i \mid \phi(x_j), \theta) \left(\phi(x_j) - \mu_i^{(t+1)}\right) \left(\phi(x_j) - \mu_i^{(t+1)}\right)^T}{\sum_{j=1}^{n} p^{(t+1)}(z_i \mid \phi(x_j), \theta)}$$

Q: Note that this algorithm is for the Mixture of Gaussians assuming a difference covariance matrix Σ_i for each class C_i. What will be the algorithm like, if we assume a shared covariance matrix Σ across all classes (that is, the Linear Discriminant Analysis discussed in Section 1.2)?

ANSWER: We will simply build on the solution to the Linear Discriminant case from Section 2.1 and simply replace multiple class-specific estimates Σ_i with a single estimate Σ:

4 Convergence of Hard K-Means Algorithm

Prove the following claim: The K-Means Clustering algorithm will converge in a finite number of iterations.

1. **Proof Sketch:** At each iteration, the K-Means algorithm reduces the objective $\sum_{j=1}^{m}\sum_{l=1}^{K} P_{l,j}\|\phi\left(\mathbf{x}^{(j)}\right) - \mu_l\|^2$ and stops when this objective does not reduce any further.

2. Hint1: $P^{(t+1)} = \underset{P}{\operatorname{argmin}} \sum_{j=1}^{m}\sum_{l=1}^{K} P_{l,j}\|\phi\left(\mathbf{x}^{(j)}\right) - \mu_l^{(t)}\|^2$

3. Hint2: $\mu^{(t+1)} = \underset{\mu}{\operatorname{argmin}} \sum_{j=1}^{m}\sum_{l=1}^{K} P_{l,j}^{(t+1)}\|\phi\left(\mathbf{x}^{(j)}\right) - \mu_l\|^2$

4. Hint3: Only a finite number of combinations of $P_{i,j}$ are possible.

Tutorial 7

CS 337 Artificial Intelligence & Machine Learning, Autumn 2019

October, 2019

Problem 1. Design a multilayer perceptron which will learn to recognize various forms of the letters C,L,T placed on a 3x3 grid (as shown in Figure 1) through backpropagation algorithm.

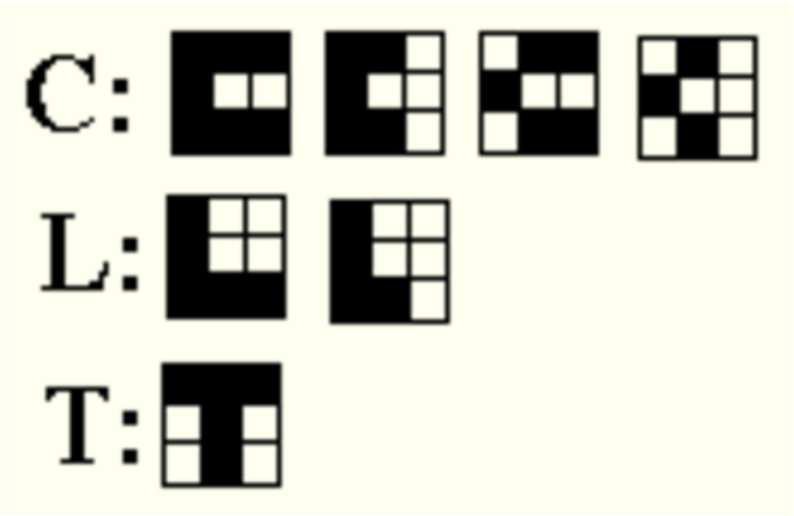


Figure 1: Various forms of the letters C,L,T placed on a 3×3 grid

1. Design a one layer network indicating what should be applied at the input layer and what should be expected at the output layer showing the number of neurons, the connections between them and the neuron's output function.
2. repeat (a) for two layer network by adding a hidden layer

Problem 2. Consider a perceptron (shown in Figure 2) for which $u \in R^2$ and

$$f(a) = \begin{cases} 1 & a > 0 \\ 0 & a = 0 \\ -1 & a < 0 \end{cases}$$

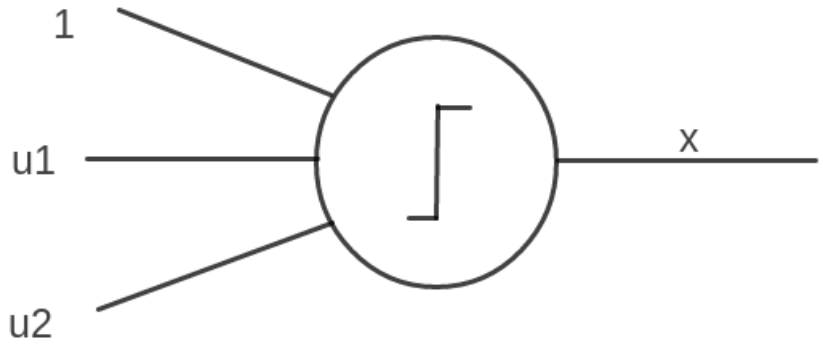


Figure 2: A simple perceptron depicting function f in Problem 2.

Let the desired output be 1 when elements of class A = {(1,2),(2,4),(3,3),(4,4)} is applied as input and let it be -1 for the class B = {(0,0),(2,3),(3,0),(4,2)}. Let the initial connection weights $w_0(0) = +1, w_1(0) = -2, w_2(0) = +1$ and learning rate be h = 0.5.

This perceptron is to be trained by perceptron convergence procedure, for which the weight update formula is $w(t+1) = w(t) + \eta(y^k - x^k(t))u^k$

1. (a) Mark the elements belonging to class A with x and those belonging to class B with o on input space.
   (b) Draw the line represented by the perceptron considering the initial connection weights w(0).
   (c) Find out the regions for which the perceptron output is +1 and –1
   (d) Which elements of A and B are correctly classified, which elements are misclassified and which are unclassified?
2. If u=(4,4) is applied at input, what will be w(1) ?
3. Repeat a) considering w(1).
4. If u=(4.2) is then applied at input, what will be w(2)?
5. Repeat 1) considering w(2).

6. Do you expect the perceptron convergence procedure to terminate? Why?

Problem 3. Problems on Neural Networks

1. We have proved the following for the perceptron update rule:

   If $||\mathbf{w}^*|| = 1$ and if there exists $\theta > 0$ such that for all $i = 1, \ldots, n$, $y_i(\mathbf{w}^*)^T\phi(\mathbf{x}_i) \geq \theta$ and $||\phi(\mathbf{x}_i)||^2 \leq \Gamma^2$ then the perceptron algorithm will make atmost $\frac{\Gamma^2}{\theta^2}$ errors (that is take atmost $\frac{\Gamma^2}{\theta^2}$ iterations to converge)

   Point out specifically the challenge in extending step(s) in the proof to *(show) convergence of backpropagation in Neural Networks?

   Solution:

   When we look at the proof from the point of view of extending this claim or line of thought to neural networks, the first thing we note is that the update rule will change owing to the use of sigmoid activation and with backpropagation. As a result, the first inequality (reproduced below in red) will break.

   We showed the following: We know that $||\mathbf{w}^*||_2^2 = 1$ and $y_i\phi(\mathbf{x}_i)\mathbf{w}^* \geq \theta$ for all i. We assume that $\mathbf{w}^{(0)} = 0$

   Now consider $(\mathbf{w}^*)^T\mathbf{w}^{(k)} = (\mathbf{w}^*)^T(\mathbf{w}^{(k-1)} + y_i\phi(\mathbf{x}_i)) = (\mathbf{w}^*)^T\mathbf{w}^{(k-1)} + y_i(\mathbf{w}^*)^T\phi(\mathbf{x}_i) \geq (\mathbf{w}^*)^T\mathbf{w}^{(k-1)} + \theta \geq (\mathbf{w}^*)^T\mathbf{w}^{(k-2)} + 2\theta \geq (\mathbf{w}^*)^T\mathbf{w}^{(0)} + k\theta = k\theta$

   Thus,

   $$(\mathbf{w}^*)^T\mathbf{w}^{(k)} \geq k\theta$$

   and because

   $$||\mathbf{w}^*||||\mathbf{w}^{(k)}|| = ||\mathbf{w}^{(k)}|| \geq |(\mathbf{w}^*)^T\mathbf{w}^{(k)}|$$

   we must have

   $$||\mathbf{w}^{(k)}|| \geq k\theta$$

   Similarly,

   $||\mathbf{w}^{(k)}||_2^2 = ||\mathbf{w}^{(k-1)} + y_i\phi(\mathbf{x}_i)||_2^2 = ||\mathbf{w}^{(k-1)}||_2^2 + y_i^2||\phi(\mathbf{x}_i)||_2^2 + 2y_i(\mathbf{w}^{(k-1)})^T\phi(\mathbf{x}_i) < ||\mathbf{w}^{(k-1)}||_2^2 + \Gamma^2 < ||\mathbf{w}^{(k-2)}||_2^2 + 2\Gamma^2 < ||\mathbf{w}^{(0)}||_2^2 + k\Gamma^2 = k\Gamma^2$

   since $y_i^2 = 1$ and it must have been that (as per perceptron update rule)

   $y_i(\mathbf{w}^{(k-1)})^T\phi(\mathbf{x}_i) < 0$

   Thus,

   $$||\mathbf{w}^{(k)}||_2^2 < k\Gamma^2$$

   and

   $$||\mathbf{w}^{(k)}||_2^2 \geq k^2\theta^2$$

   which implies

   $$k^2\theta^2 < k\Gamma$$

that is,

$$k < \frac{\Gamma}{\theta^2}$$

which proves our claim.

2. Consider one layer of weights (edges) in a convolutional neural network (CNN) for grayscale images, connecting one layer of units to the next layer of units. Which type of layer has the fewest parameters to be learned during training? Explain/justify your answer.

   (A) A convolutional layer with ten 3×3 filters
   (B) A max-pooling layer that reduces a 10×10 image to 5×5
   (C) A convolutional layer with eight 5×5 filters
   (D) A fully-connected layer from 20 hidden units to 4 output units

   Solution:

   (B). It is based on simple parameter calculation similar to what will be done in class.

Problem 4. Neural logic

Our goal is to build a simple neural network which takes a point $(x_1, x_2) \in [0, 1]^2$ (i.e., a point in the unit square), and outputs $f(x_1, x_2) \in \{0, 1\}$, where f is as shown in the Figure 3 below. (The shaded region, including its boundaries, corresponds to all the points which are mapped to 1.)

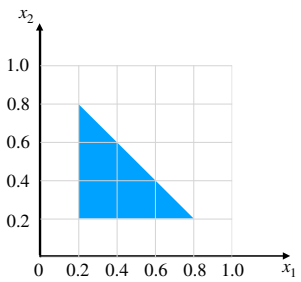


Figure 3: Figure showing function f that takes a point $(x_1, x_2) \in [0, 1]^2$ (i.e., a point in the unit square), and outputs $f(x_1, x_2) \in \{0, 1\}$

Suppose a 2-input neuron is defined as $N(x_1, x_2) = \tau(w_0 + w_1 x_1 + w_2 x_2)$ where $w_0, w_1, w_2 \in \mathbb{R}$ are real-valued weights, and $\tau : \mathbb{R} \to \{0, 1\}$ is defined so that $\tau(x) = 1$ iff $x \geq 0$.

Show a set of weights on a two-layer neural network as shown below in the Figure 4 to implement f. The 3-input neuron on the top layer should correspond to the boolean operator AND (when 0 is interpreted as false and 1 as true). Fill in all the blanks corresponding to the weights in the Figure 4 and justify your answers.

Solution:

F1: $w_0 = -0.2, w_1 = 1, w_2 = 0$
F2: $w_0 = -0.2, w_1 = 0, w_2 = 1$
F3: $w_0 = 1, w_1 = -1, w_2 = -1$
AND: $w_0 = -3, w_1 = 1, w_2 = 1, w_3 = 1$

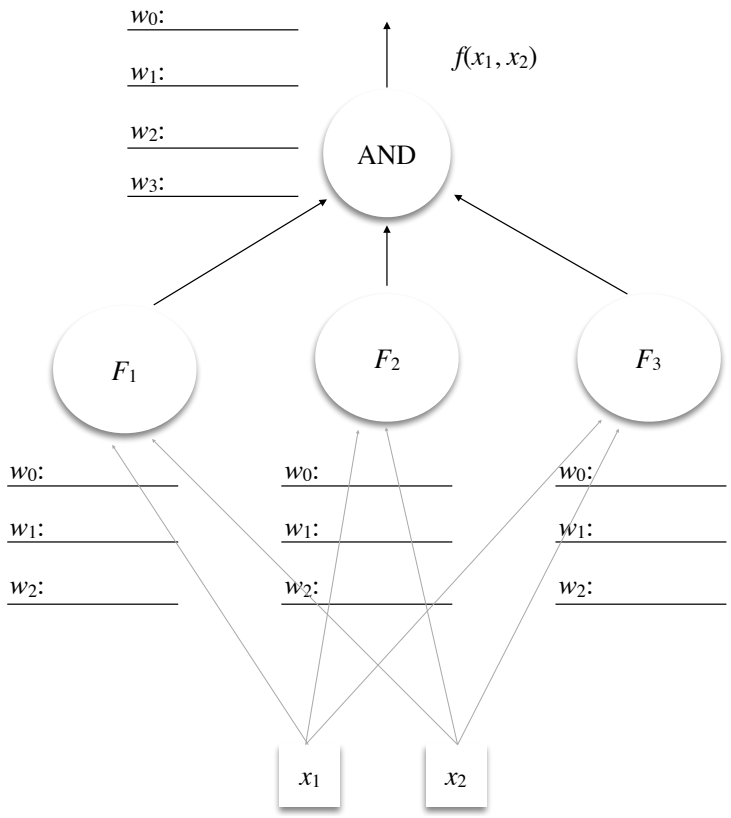


Figure 4: Fill in all the blanks corresponding to the weights in this figure and justify your answers.

Problem 5. In the class, we discussed the probabilistic binary (class) logistic regression classifier. How will you extend logistic regression probabilistic model to multiple (say K) classes? Are their different ways of extending? What is the intuition behind each? Discuss and contrast advantages/disadvantages in each.

Solution:

One might suggest handling multi-class (K) classification via K one-vs-rest probabilistic classifiers. But there is no obvious probabilistic semantics associated with such a classifier (question asked for a probabilistic MODEL for multiple classes).

Basic idea is that each class c can have a different weight vector $[w_{c,1}, w_{c,2}, \ldots, w_{c,k}, \ldots, w_{c,K}]$.

Extension to multi-class logistic

1. Each class $c = 1, 2, \ldots, K-1$ can have a different weight vector $[\mathbf{w}_{c,1}, \mathbf{w}_{c,2}, \ldots, \mathbf{w}_{c,k}, \ldots, \mathbf{w}_{c,K-1}]$ and

$$p(Y = c|\phi(\mathbf{x})) = \frac{e^{-(\mathbf{w}_c)^T \phi(\mathbf{x})}}{1 + \sum_{k=1}^{K-1} e^{-(\mathbf{w}_k)^T \phi(\mathbf{x})}}$$

for $c = 1, \ldots, K-1$ so that

$$p(Y = K|\phi(\mathbf{x})) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{-(\mathbf{w}_k)^T \phi(\mathbf{x})}}$$

Alternative (equivalent) extension to multi-class logistic

1. Each class $c = 1, 2, \ldots, K$ can have a different weight vector $[\mathbf{w}_{c,1}, \mathbf{w}_{c,2} \ldots \mathbf{w}_{c,p}]$ and

$$p(Y = c|\phi(\mathbf{x})) = \frac{e^{-(\mathbf{w}_c)^T \phi(\mathbf{x})}}{\sum_{k=1}^{K} e^{-(\mathbf{w}_k)^T \phi(\mathbf{x})}}$$

for $c = 1, \ldots, K$.

This function is also the called the **softmax**[1] function.

Problem 6. Suppose you are provided a multi-layer neural network for a binary classification problem. Can you convert this network into an equivalent kernel perceptron (single node neural network only)? What will be the exact steps?

Solution:

1. Yes. You are already provided the NN classifier. So you need not worry about separability etc. In fact, the question was only about an equivalent classifier (even if not-separating completely) which can be constructed as follows:

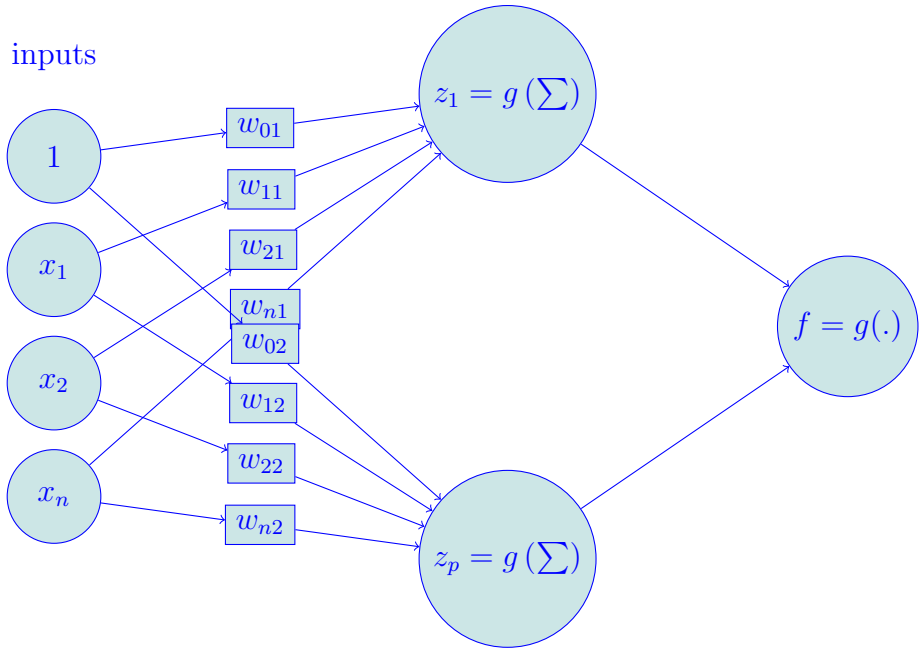
2. Let the neural network be as shown below with p nodes $(z_1, z_2, \ldots, z_p)$ in the last layer. You can develop an equivalent kernel preceptron by specifying

$$\phi_z(x) = [z_1(x), z_2(x), \ldots, z_p(x)]$$

and using the following Kernel:

$$K_z(x, x') = \phi_z^T(x)\phi_z(x')$$

[1] `https://en.wikipedia.org/wiki/Softmax_function`



Problem 7. In class, we saw the detailed derivation of backpropogation update rules when each of the activation units is a sigmoid. You need to derive all the update rules when each activation unit happens to be rectified linear unit (ReLU).

$$\sigma(s) = max(\theta, s)$$

(since we often represent σ).) by $g(.)$, this also means $g(s) = max(\theta, s)$)

Typically, $\theta = 0$. Note that ReLU is differentiable at all points except at $s = \theta$. But by using subgradient $\nabla_s \sigma$ instead of gradient $\nabla \sigma$, we can complete backpropagation as 'subgradient descent'. Note that subgradient is the same as gradient in regions in which the function is differentiable. Thus,

$$\nabla_s \sigma(s) = 1,\ s \in (\theta, \infty)\ ,\ \nabla_s \sigma(s) = 0 \text{ if } s < \theta \text{ and } \nabla_s \sigma(s) \in [0,1] \text{ if } s = \theta$$

The interval $[0, 1]$ is the subdifferential (denoted ∂), which is set of subgradients of σ at θ.

Is there a problem in cascading several layers of ReLU? Recall that we invoked subgradients in justifying the *Iterative Soft Thresholding Algorithm* for LASSO. And that LASSO gave sparsity owing to hard thresholding.

Solution:

All the gradients and partial derivatives in the backpropagation algorithm will remain unchanged except for the $\frac{\partial \sigma_{\mathbf{p}}^{\mathbf{l+1}}}{\partial \mathbf{sum}_{\mathbf{p}}^{\mathbf{l+1}}}$ since σ is not differentiable now at all points. So the new

$$\frac{\partial \sigma_p^{l+1}}{\partial sum_p^{l+1}} = 1,\ sum_p^{l+1} \in [\theta, \infty)\ ,\ \frac{\partial \sigma_p^{l+1}}{\partial sum_p^{l+1}} = 0 \text{ if } sum_p^{l+1} < \theta$$

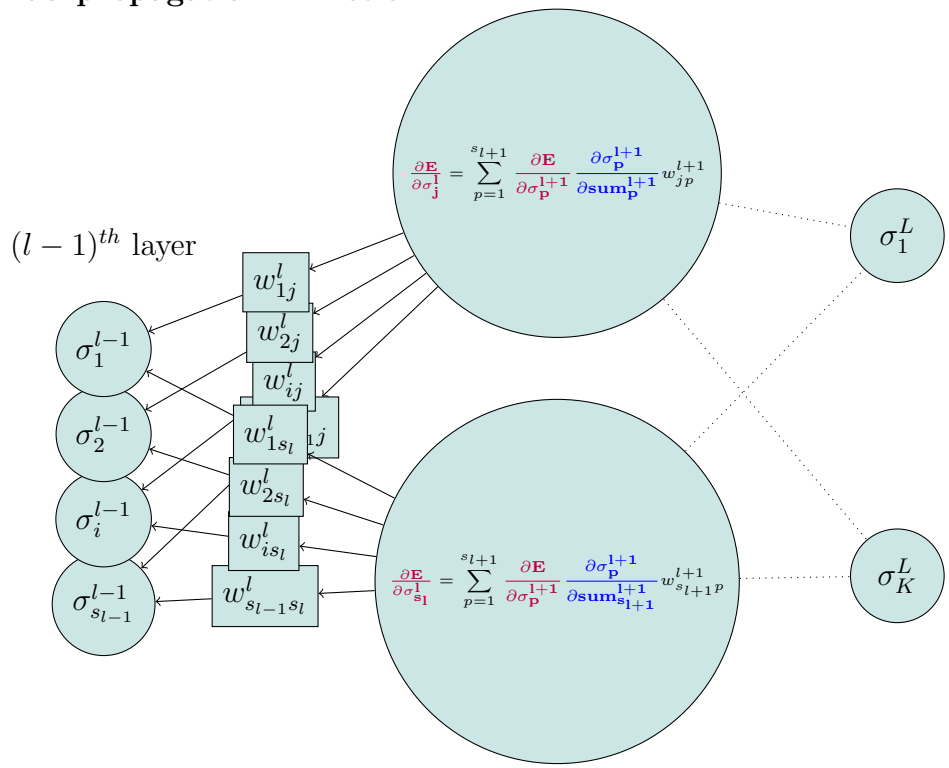
will be one possible choice

- For a single example $(\mathbf{x}, y)$:

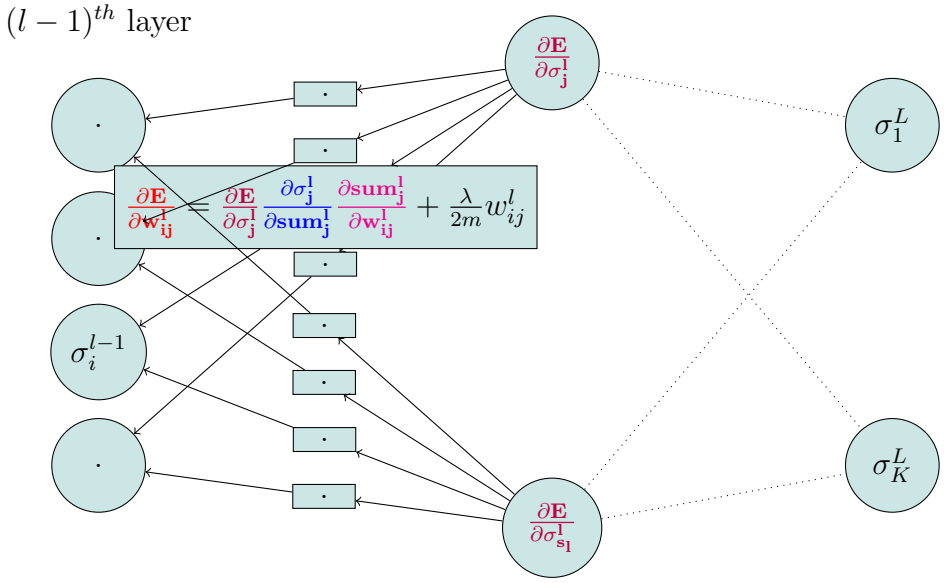
$$-\left[\sum_{k=1}^{K} y_k \log\left(\sigma_k^L(\mathbf{x})\right) + (1-y_k)\log\left(1-\sigma_k^L(\mathbf{x})\right)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L}\sum_{i=1}^{s_{l-1}}\sum_{j=1}^{s_l}\left(w_{ij}^l\right)^2 \tag{1}$$

- $\frac{\partial \mathbf{E}}{\partial \sigma_{\mathbf{j}}^{\mathbf{l}}} = \sum_{p=1}^{s_{l+1}} \frac{\partial E}{\partial sum_p^{l+1}} \frac{\partial sum_p^{l+1}}{\partial \sigma_j^l} = \sum_{p=1}^{s_{l+1}} \frac{\partial \mathbf{E}}{\partial \sigma_{\mathbf{p}}^{\mathbf{l+1}}} \frac{\partial \sigma_{\mathbf{p}}^{\mathbf{l+1}}}{\partial \mathbf{sum}_{\mathbf{p}}^{\mathbf{l+1}}} w_{jp}^{l+1}$ since $\frac{\partial sum_p^{l+1}}{\partial \sigma_j^l} = w_{jp}^{l+1}$
- $\frac{\partial \mathbf{E}}{\partial \sigma_{\mathbf{j}}^{\mathbf{L}}} = -\frac{\mathbf{y_j}}{\sigma_{\mathbf{j}}^{\mathbf{L}}} - \frac{\mathbf{1-y_j}}{\mathbf{1}-\sigma_{\mathbf{j}}^{\mathbf{L}}}$

Backpropagation in Action



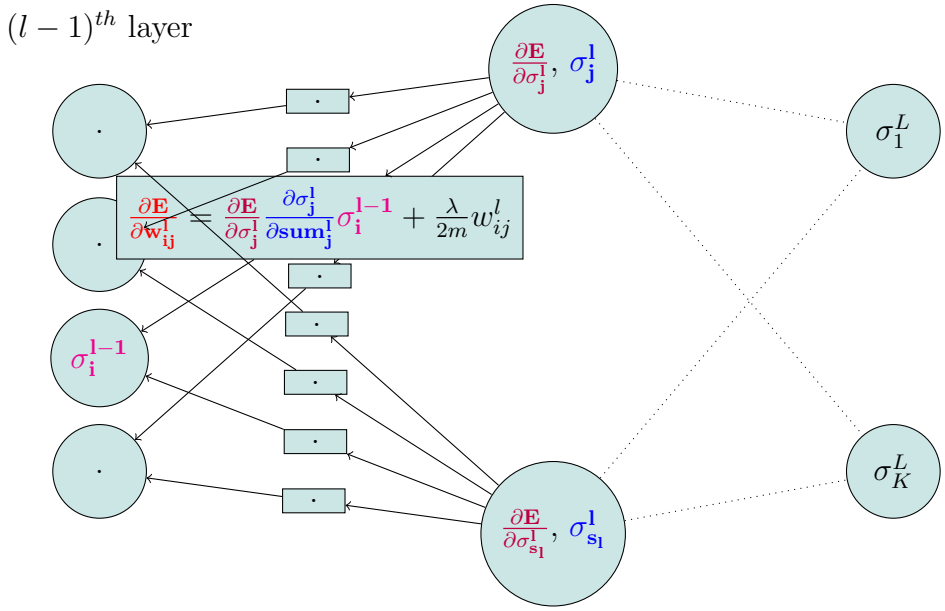
Backpropagation in Action



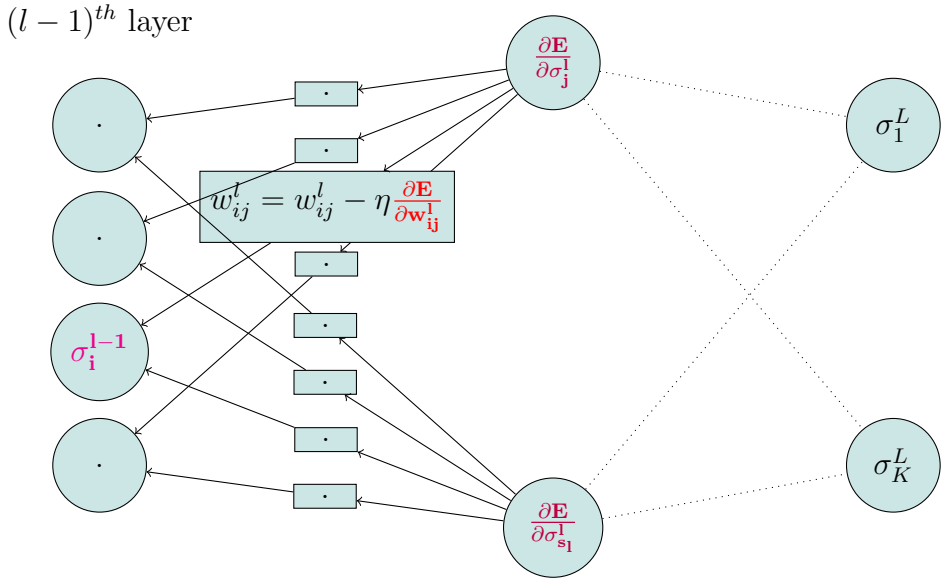
Recall and Substitute

- $sum_j^l = \sum_{k=1}^{s_{l-1}} w_{kj}^l \sigma_k^{l-1}$ and $\sigma_i^l = \frac{1}{1+e^{-sum_i^l}}$
- $\frac{\partial \mathbf{E}}{\partial \mathbf{w}_{\mathbf{ij}}^{\mathbf{l}}} = \frac{\partial \mathbf{E}}{\partial \sigma_{\mathbf{j}}^{\mathbf{l}}} \frac{\partial \sigma_{\mathbf{j}}^{\mathbf{l}}}{\partial \mathbf{sum}_{\mathbf{j}}^{\mathbf{l}}} \frac{\partial \mathbf{sum}_{\mathbf{j}}^{\mathbf{l}}}{\partial \mathbf{w}_{\mathbf{ij}}^{\mathbf{l}}} + \frac{\lambda}{2m} w_{ij}^l$
- $\frac{\partial \sigma_{\mathbf{j}}^{\mathbf{l}}}{\partial \mathbf{sum}_{\mathbf{j}}^{\mathbf{l}}} = \mathbf{1}$, **if** $\mathbf{sum}_{\mathbf{j}}^{\mathbf{l}} \in [\theta, \infty)$, $\frac{\partial \sigma_{\mathbf{j}}^{\mathbf{l}}}{\partial \mathbf{sum}_{\mathbf{j}}^{\mathbf{l}}} = \mathbf{0}$ if $\mathbf{sum}_{\mathbf{j}}^{\mathbf{l}} < \theta$
- $\frac{\partial \mathbf{sum}_{\mathbf{j}}^{\mathbf{l}}}{\partial \mathbf{w}_{\mathbf{ij}}^{\mathbf{l}}} = \sigma_{\mathbf{i}}^{\mathbf{l-1}}$
- $\frac{\partial \mathbf{E}}{\partial \sigma_{\mathbf{j}}^{\mathbf{l}}} = \sum_{p=1}^{s_{l+1}} \frac{\partial \mathbf{E}}{\partial \sigma_{\mathbf{j}}^{\mathbf{l+1}}} \frac{\partial \sigma_{\mathbf{j}}^{\mathbf{l+1}}}{\partial \mathbf{sum}_{\mathbf{j}}^{\mathbf{l+1}}} w_{jp}^{l+1}$
- $\frac{\partial \mathbf{E}}{\partial \sigma_{\mathbf{j}}^{\mathbf{L}}} = -\frac{\mathbf{y_j}}{\sigma_{\mathbf{j}}^{\mathbf{L}}} - \frac{\mathbf{1-y_j}}{\mathbf{1-\sigma_j^L}}$

Backpropagation in Action



Backpropagation in Action



The Backpropagation Algorithm for Training NN

1. Randomly initialize weights w_{ij}^{l} for $l = 1, \ldots, L$, $i = 1, \ldots, s_l$, $j = 1, \ldots, s_{l+1}$.
2. Implement **forward propagation** to get $f_{\mathbf{w}}(\mathbf{x})$ for every $\mathbf{x} \in \mathcal{D}$.
3. Execute **backpropagation** on any misclassified $\mathbf{x} \in \mathcal{D}$ by performing gradient descent to minimize (non-convex) $E(\mathbf{w})$ as a function of parameters $\mathbf{w}$.
4. $\frac{\partial \mathbf{E}}{\partial \sigma_{\mathbf{j}}^{\mathbf{L}}} = -\frac{\mathbf{y_j}}{\sigma_{\mathbf{j}}^{\mathbf{L}}} - \frac{\mathbf{1-y_j}}{\mathbf{1}-\sigma_{\mathbf{j}}^{\mathbf{L}}}$ for $j = 1$ to s_L.
5. For $l = L-1$ down to 2:
   (a) $\frac{\partial \sigma_{\mathbf{j}}^{\mathbf{l}}}{\partial \mathbf{sum}_{\mathbf{j}}^{\mathbf{l}}} = \mathbf{1}$, **if** $\mathbf{sum}_{\mathbf{j}}^{\mathbf{l}} \in [\theta, \infty)$, $\frac{\partial \sigma_{\mathbf{j}}^{\mathbf{l}}}{\partial \mathbf{sum}_{\mathbf{j}}^{\mathbf{l}}} = \mathbf{0}$ if $\mathbf{sum}_{\mathbf{j}}^{\mathbf{l}} < \theta$

(b) $\frac{\partial \mathbf{E}}{\partial \sigma_{\mathbf{j}}^{\mathbf{l}}} = \sum_{p=1}^{s_{l+1}} \frac{\partial \mathbf{E}}{\partial \sigma_{\mathbf{j}}^{\mathbf{l+1}}} \frac{\partial \sigma_{\mathbf{j}}^{\mathbf{l+1}}}{\partial \mathbf{sum}_{\mathbf{j}}^{\mathbf{l+1}}} w_{jp}^{l+1}$

(c) $\frac{\partial \mathbf{E}}{\partial \mathbf{w}_{\mathbf{ij}}^{\mathbf{l}}} = \frac{\partial \mathbf{E}}{\partial \sigma_{\mathbf{j}}^{\mathbf{l}}} \frac{\partial \sigma_{\mathbf{j}}^{\mathbf{l}}}{\partial \mathbf{sum}_{\mathbf{j}}^{\mathbf{l}}} \sigma_{\mathbf{i}}^{\mathbf{l-1}} + \frac{\lambda}{2m} w_{ij}^{l}$

(d) $w_{ij}^{l} = w_{ij}^{l} - \eta \frac{\partial \mathbf{E}}{\partial \mathbf{w}_{\mathbf{ij}}^{\mathbf{l}}}$

6. Keep picking misclassified examples until the cost function $E(\mathbf{w})$ shows significant reduction; else resort to some random perturbation of weights $\mathbf{w}$ and restart a couple of times.

Problem 8. Compute the minimum number of multiplications and additions for a single backpropagation while also estimating the memory required for the minimum number of such multiplications and additions to become possible.

Problem 9. Solve the assignment at `https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/udacity/4_convolutions.ipynb`

Follow the instructions to implement and run each indicated step. Some steps have been implemented for you. This is a self-evaluated assignment. Make sure you are able to solve each problem and answer any posed questions and save the answers/solutions wherever possible.

Problem 8. Extend the backpropagation algorithm, developed on general Neural Networks to Convolutional Neural Networks (with and without Max Pooling).

Advanced and optional: Also extend the backpropagation algorithm to Recurrent Neural Networks and LSTMs.

Problem 10. ConvNetJS (`http://cs.stanford.edu/people/karpathy/convnetjs/`) is a Javascript library for training Deep Learning models (Neural Networks) entirely in your browser. Try different choices of network configurations which include the choice of the stack of convolution, pooling, activation units, number of parallel networks, position of fully connected layers and so on. You can also save some network snapshots as JSON objects. What does the network visualization of the different layers reveal?

Also try out the demo at `http://places.csail.mit.edu/demo.html` to understand the heat maps and their correlations with the structure of the neural network.

Problem 11. Discuss the advantages and disadvantages of different activation functions: tanh, sigmoid, ReLU, softmax. Explain and illustrate when you would choose one activation function in lieu of another in a Neural Network. You can also include any experiences from Problem 5 in your answer.

Tutorial 8

CS 337 Artificial Intelligence & Machine Learning, Autumn 2019

October, 2019

1 Connecting Graphical Models to Propositional Logic

The Indian fruit Jambul[1] is often confused with Blackberries[2]. However, we will use certain regional and seasonal growing patterns of these two fruits along with their physical characteristics to help classify the fruit[3].

Some village folk tell us that the following clauses (together comprising Σ) are good enough for characterizing and discriminating between these fruits.

- blackberry $\leftrightarrow$ north
- blackberry $\leftrightarrow$ winter
- blackberry $\leftrightarrow$ autumn
- blackberry $\leftrightarrow$ medium_in_color
- blackberry $\leftrightarrow$ dark_in_color
- blackberry $\leftrightarrow$ small_in_size
- jambul $\leftrightarrow$ south
- jambul $\leftrightarrow$ spring
- jambul $\leftrightarrow$ summer
- jambul $\leftrightarrow$ light_in_color
- jambul $\leftrightarrow$ big_in_size

[1] `http://en.wikipedia.org/wiki/Jambul`

[2] `http://en.wikipedia.org/wiki/Blackberry`. It is the edible blackberry, not the non-edible smartphone :-)

[3] Note that the data we have is just cooked up.

However, feeling uncomfortable with the fact that blackberries do sometimes grow in the south, that blackberries are sometimes light in color, *etc.*, we decide to probabilistically model[4] the correlations between characteristics and fruits using an undirected[5] graphical model.

We thus start off with the following random variables: $\{S, F, L, C, Z\}$ where S stands for 'seasons' and takes four values, *viz.*, {winter, spring, summer, autumn}, F stands for the fruit and takes on the values {blackberry, jambul}, L stands for the location and assumes two values {north, south}. C stands for the color and takes values {light, medium, dark} while Z stands for the size and takes values {big, small}. We will abbreviate blackberry by b and jambul by j. We consider an undirected graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ with the vertex set $\mathcal{V} = \{S, F, L, C, Z\}$ and edge set $\mathcal{E} = \{(S, F), (L, F), (C, F), (Z, F)\}$.

The potential functions over the maximal cliques (*i.e.*, edges) are given in the following tables:

S	$\phi(S, F = b)$	$\phi(S, F = j)$
winter	0.9	0.1
spring	0.3	0.7
summer	0.4	0.6
autumn	0.8	0.2

L	$\phi(L, F = b)$	$\phi(L, F = j)$
north	0.65	0.35
south	0.25	0.75

F	$\phi(C = light, F)$	$\phi(C = medium, F)$	$\phi(C = dark, F)$
b	0.33	0.33	0.34
j	0.8	0.1	0.1

F	$\phi(Z = big, F)$	$\phi(Z = small, F)$
b	0.4	0.6
j	0.95	0.05

- Suppose you are given a piece of fruit and you find that it is small and has medium color. What season is it now, most likely? What is your probability of being correct?
- As against this, what will be the result of (logical) inference from Σ given the evidences? Is there any difference between the results from logical and probabilistic inference? Explain. What will be the result if logical inference were considered on treating Σ as a "decision list"?

2 Autocompletion, HMM and RNN

All of us know that Whatsapp and other messaging services have an autocomplete feature, wherein, as we type characters, the different possible word completions are prompted. In

[4]Obviously we needed to put more efforts and engaged a CS337 student in the modeling exercise.

[5]Since the implication in the clauses in Σ is a two-sided $\leftrightarrow$, an undirected graph makes more sense.

Figure 1, you can see the different autocomplete options for a message as I was typing on a whatsapp group involving several farmers.

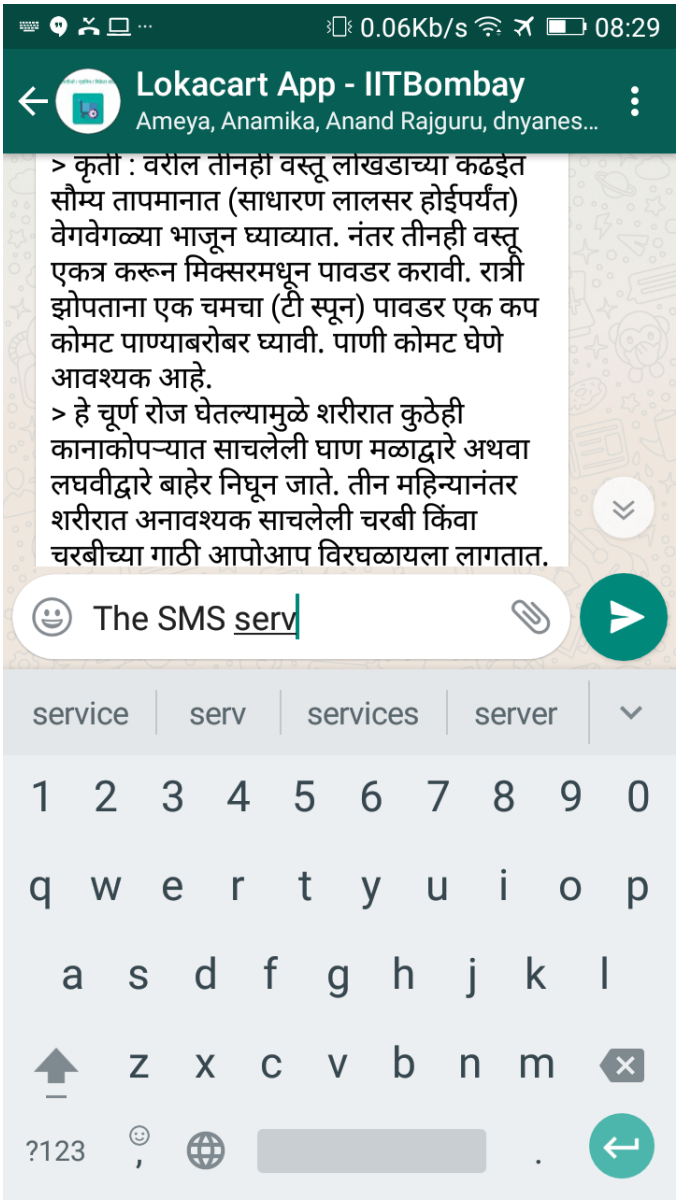


Figure 1: A figure showing the autocomplete feature in Whatsapp

Let us say that traditionally, this autocomplete was enabled by storing a prefix tree over all characters. Each node n_i in the tree represents a prefix (such as node labeled 'ser') and each directed edge e_iij represents the appending of a character to the prefix in n_i (such as appending 'v' to 'ser') to give rise to a longer prefix in n_j (such as 'serv'). We refer to n_j as an immediate **extension** of n_i. Thus, if the characters typed so far are 'serv', the autocompletion options correspond to all the word **extensions** that result from traversing down from the node n_j with prefix 'serv' to every leaf level node (such as 'service', 'services',

'server', etc.). In Figure (2) we have depicted only a small subtree in the larger prefix tree of all strings.

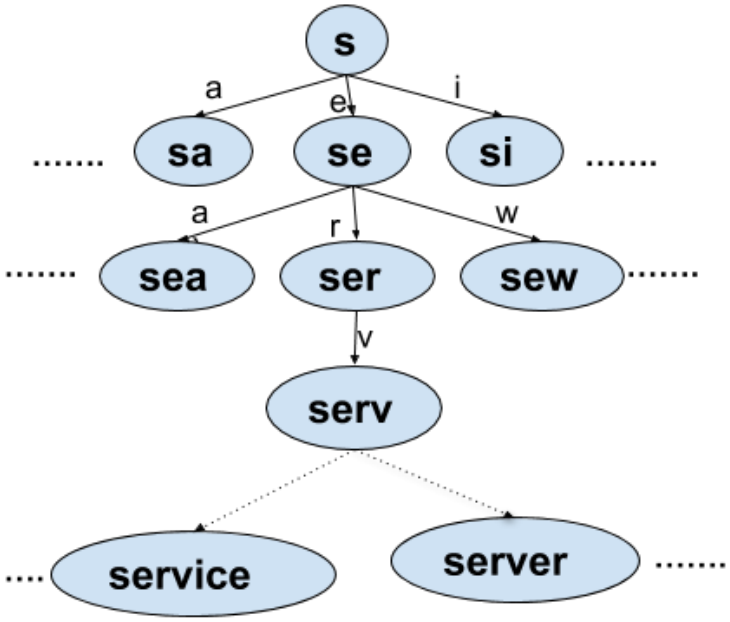


Figure 2: A subtree in the larger prefix tree of all strings

1. Suppose, that in all there are 90 characters that are ever seen on this farmer's Whatsapp group (these include the 26 characters in English and characters in Devanagari etc for also supporting Hindi), what is potentially the number of nodes in this prefix tree? A word can begin with any of the 90 characters. You can answer this question for the average case in which each word is of length 7, and each non-empty prefix has 3 possible immediate **extensions**.

   Solution: Starting from the root, each node has 3 nodes as children. Thus, adding the total number of nodes at each level:$90 \times (1 + 3^2 + ... + 3^6)$.

2. Now you, as our our smart CS337 student believes that prefix trees are outdated and need to be replaced with more relevant and contemporary Machine Learning models for this Autocompletion task. In particular, you need to design an HMM (Hidden Markov Model) and a RNN (Recurrent Neural Network) for this task. You can assume the same statistics stated in part 1 and design the appropriate number of nodes/parameters in the HMM and RNN.

   (a) Describe your HMM as well as your RNN and state the number of parameters you chose inspired by part 1 above.

   (b) Explain the correspondence between the nodes/edges in the prefix tree and the nodes/edges in the HMM and RNN.

Solution: A possible answer is: One hot encoding based on 90 characters in the input. 3 nodes in a hidden layer corresponding to the average fanout of 3 mentioned in part 1. And one hot encoding based output which means 90 nodes in the output layer as well. This amounts to a total of 90×3(input to hidden) $+ 90 \times 3$(hidden to output) $+$ 3(last one for the self loop) $= 543$ parameters. There are exponentially many nodes in the tree of all character strings. In an HMM or RNN, each node is a hidden state vector. The next character must transform this to a new node.

(a) If the nodes are implemented as hidden states in an RNN, different nodes can share structure because they use distributed representations.

(b) The next hidden representation needs to depend on the conjunction of the current character and the current hidden representation.

3. How would you perform MAP inference in your HMM or RNN (assuming parameters were learnt)? When would you use Beam search? When woud you use A* search? And when would you use the Max-product algorithm (Viterbi)? Discuss advantages and disadvantages of using prefix trees vs. your HMM as well as your RNN.

Solution:

(a) Advantages of RNN/Disadvantages of prefix-tree:

i. More compact representation.

ii. Can be trained and even dynamically updated based on choices made by the user as she/he types and select auto-completions.

iii. On the other hand, the prefix tree is a bit dumb and is independent of the statistics of the usage of different words. For example, even if 'servable' is a valid word, it is rarely used and the RNN can be eventually trained to reflect this knowledge. But the prefix tree method would treat 'servable' to be as likely an auto-completion option as service.

(b) Disadvantages of RNN/Advantages of prefix-tree

i. The prefix tree will be deterministic and fast to infer.

ii. RNN will incur overheads in training (and also at test time - though students might not be able to guess this aspect). So we will need to decide how often to retrain the RNN model as we go on collecting auto-completion options from the user(s).

iii. Also, we need to decide when the word will end. So perhaps an end of word character also needs to be included (making the size 91) and predicted.

iv. Extra/Optional: As for inference, the RNN will need to keep track of all the best/most probably sequences starting from the current input. This will require beam search (exact optimization can be expensive).

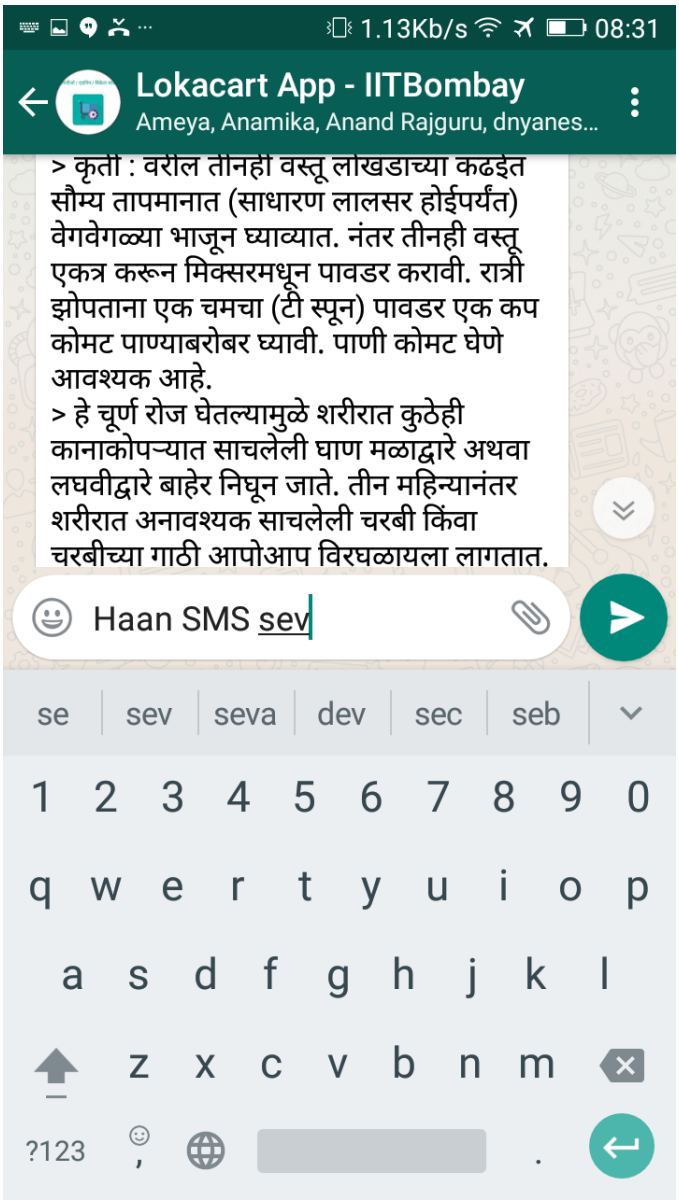


Figure 3: A figure showing the autocomplete feature adopted to Hindi words that might also be typed using the English (latin) alphabet

4. **Slightly Harder - Code mixing:** We realize that in our farmer's WhatsApp group, people often type Hindi words in latin (that is, using English alphabets). This is quite common in India and in (3) we illustrate how the autocomplete feature has been adopted to Hindi words that might also be typed using the English (latin) alphabet. Thus, while 'SMS' is a word purely from English, 'Haan' and 'seva' (as an auto-completion option) are Hindi words typed in English. This is often referred to as **code-mixing**

   (a) What would happen if we simply applied either the Prefix tree model in question 1

or your HMM or RNN model in question 2 in this setting? What are limitations?

Solution: In both cases, the vocabulary of English and the other language Hindi would be completely mixed up and that too in the same script. If the scripts were different, atleast language detection could have been enabled using the script. In the absence of that information, the suggestions would (i) dumbly be from a mixture of two different languages expressed in the same script and (ii) the user might get confused/surprised since she/he starts seeing suggestions from a language she/he is typing in!

(b) Our CS337 student is however determined and wants to extend the model from question 2 to also capture this new **code-mixing** phenomenon. Present an extension/modification of your HMM or RNN model in question 2 to help capture this phenomenon. Justify your choice and present the number of parameters your model has. Compare your new model with the model in part 2. *As an optional hint, you can draw inspiration from Principal Component Analysis (PCA) while proposing your extension.*

Solution:

i. One option is that you can simply add one more hidden layer with two nodes that connect directly to the 90 input nodes (corresponding to implicit language detection). So in this setting, the answer will be: One hot encoding based on 90 characters in the input. 2 nodes in the first hidden layer corresponding to the choice of the language. 3 nodes in the second hidden layer corresponding to the average fanout of 3 mentioned in part 1. And one hot encoding based output which means 90 nodes in the output layer as well. Also, the input nodes can also be (optionally) connected to the second hidden layer, since determining the code-switching might require access to more context. This amounts to a total of 90×2(input to first hidden) $+$ 2×3(first hidden to second hidden) $+ 90 \times 3$(input to second hideen) $+ 90 \times$ 3(second hidden to output) $+ 3 + 2$(last one for the self loops) $= 543 + 180 +$ $6 + 2 = 733$ parameters. i.e., 188 more parameters than the previous model. Other answers are also possible - for example using LSTMs.

ii. Another option is that one does PCA on the 90 dimensional one-hot encoded input as a **pre-processing step** and takes the smaller dimensional PCA output (say eigenvectors corresponding to the largest two eigenvalues) as input to the RNN. So the new RNN will have everything like the previous RNN in part 2 (a) but the number of input nodes will be different.

iii. Other options can also be entertained - such as use of LSTMS as long as the choice is **somewhat justified** and as long as the number of parameters is presented.

3 Noisy-or representation

Let $X_1, X_2, \ldots X_M$ be parents of Y in a bayesian network. Let $Y, X_1, \ldots, X_M \in \{0, 1\}$. The conditional probability table (CPT) for $p(Y|X_1, \ldots, X_M)$ should ordinarily consist of lot of

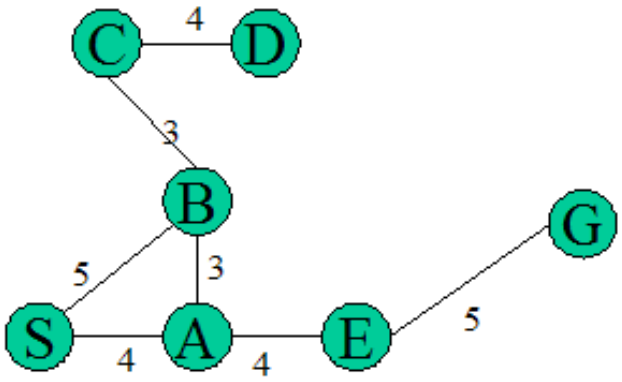


Figure 4: Map of routes. The numbers on the edges are distances.

entries. How many are they?

Now, here is a representation of the same CPT using just M parameters! This representation, called the noisy-or representation, is as follows:

$$p(Y = 1|X_1, \ldots, X_M) = 1 - \prod_{i=1}^{M} (1 - \mu_i)^{X_i}$$

where the parameters μ_i $(i = 1, 2, \ldots, M)$ represent the probabilities $p(X_i = 1)$. Explain why this representation for the CPT for $p(Y|X_1, \ldots, X_M)$ might be called a 'noisy-or' representation.

4 Search: Figure out the A* search

Consider the map of routes as in Figure 4. The numbers on the edges are distances. Consider the problem of organizing search for finding paths from S (start) to G (goal). Your goal is to design the A* algorithm for finding the shorted parth from S (start) to G (goal). We will help you by taking you through the other algorithms.

4.1 Depth First Search

Starting at the source, every time you get a choice for the next step, choose a next step and go ahead. We will have the convention that when we forge ahead, we will take the first choice. Thus, a depth first search on the map in Figure 4 will tread along the path in Figure 5. In practice, this method could yield very complicated solutions.

4.2 Breadth First Search (BFS)

The way BFS organizes its examination of the tree is layer by layer; the algorithm first explores one layer of solutions (A or B), then forges ahead to another layer (B or E or A or C) and so on. Figure 6 illustrates the BFS traversal for the map in Figure 4. In practice,

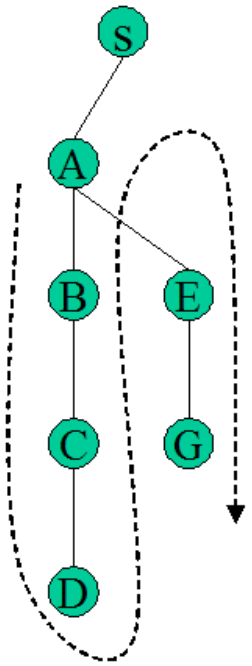


Figure 5: The DFS tree from S to G through the map in Figure 4.

this search could expend a lot of time in useless parts of the graph. But it yields the shortest path in terms of number of number of streets covered.

4.3 Hill Climbing

Both BFS and DFS are completely uninformed about geography; neither information about distance nor direction is exploited. But often it helps if you have outside information about how good a place is to be in. For instance, in the map in Figure 4, between E and D, it is much better to be in E because that gets you closer to the goal. It makes less sense in general to head off to the right if you know that the goal is on the left. It is heuristically good to be closer to the goal, though sometimes it may turn out not to be a good idea. So we could make use of heuristic information of this kind. And this forms the idea behind hill climbing. It is an idea drafted on top of depth first search. When initially at S, you could move to A or B. When you look at A and B, you that one of them is closer to the goal G and you choose that for the next move. And B happens to be closer to G, which you pick for the next move; but this turns out to be a bad idea as we will see. Nevertheless, it looks good from the point of view of the short-sighted hill climbing algorithm. From B, the possible choices are A and C. And A is a natural choice for hill-climbing, owing to its proximity to G. From A onwards, the choices are straightforward - you move to E and then to G. Figure 7 shows the route from S to G as chalked out by hill climbing.

Hill climbing always is greedy because it plans only for one step at a time. The method requires a metric such as distance from the goal.

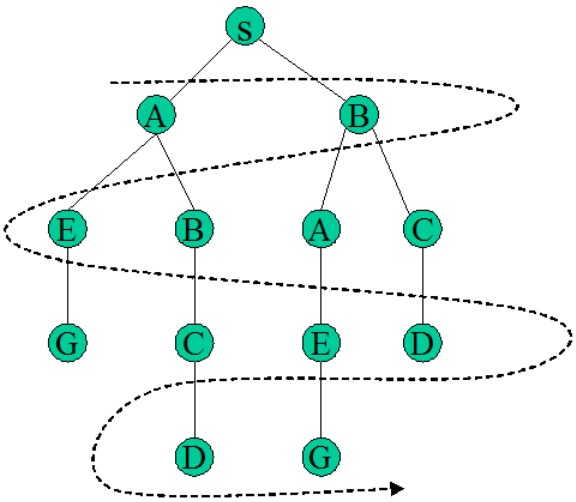


Figure 6: The BFS tree from S to G through the map in Figure 4.

Figure 7: The choices based on hill climbing for the route from S to G through the map in Figure 4.

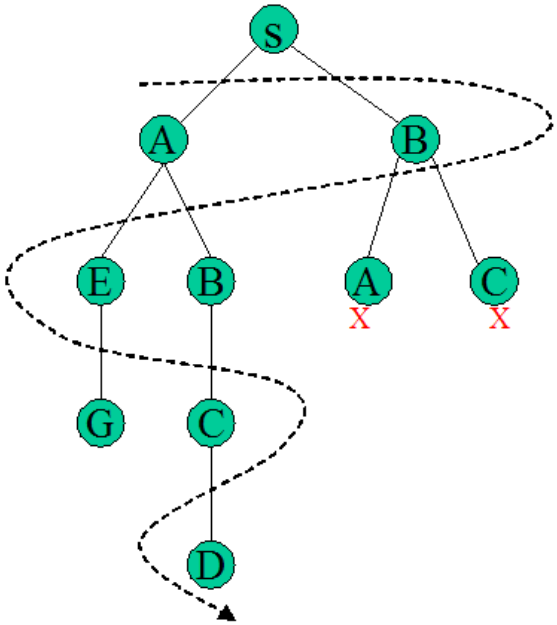


Figure 8: The choices based on beam search for the route from S to G through the map in Figure 4.

4.4 Beam Search

We drafted hill climbing on top of depth first search. Is there a similar thing we could do with breadth first search? And the answer is 'yes'. And this approach is called *beam search* The problem with breadth first search is that it tends to be exponential. But what we could do to work around is to throw away at every level of BFS, all but the 'most promising' of the paths so far. The most promising step is defined as that step which will get us 'closest' to the goal. The only difference from the hill climbing approach is that beam search does not pick up just one path, it picks up some fixed number of paths to carry down. Let us try beam search on the map in Figure 4. For this simple example, let us keep track of two paths at every level. We will refer to the BFS plan in Figure 6. From S, we could move to A or B and we keep track of both possibilities. Further, from A, there are two possibilities, *viz.*, B and E, while from B there are two possibilities in the form of A and C. Of the four paths, which two should we retain? The two best (in terms of the heuristic measure of how far we are from the goal) are B and E. Carrying forward from these points, we arrive at G in a straight-forward manner as shown in Figure 8.

While the vanila breadth first search is exponential in the number of levels, beam search has a fixed width and is therefore a constant in terms of number of levels. Beam search is however not guaranteed to find a solution (though the original BFS is guaranteed to find one). This is a price we pay for saving on time. However, the idea of backing up can be employed here; we could back up to the last unexplored path and try from there.

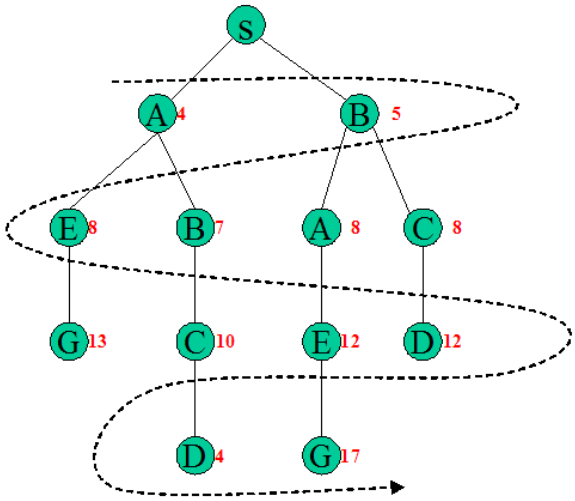


Figure 9: The tree of all possible paths from S to G through the map in Figure 4, with cumulative path lengths marked at each node.

4.5 Optimal Search

For optimal search, we will start with the brute force method which is exponential and then slap heuristics on top to reduce the amount of work, while still assuring success.

4.6 Branch and Bound

Suppose the problem is to find the best possible path (in terms of distances) between S and G in the map as in Figure 4. An oracle suggests that the path $SAEG$ is the shortest one (its length is 13). Can we determine if $SAEG$ is indeed the shortest path? One method to answer this question is to verify if every other path is at least that long.

From S, we could go to A or B. The cumulative path length to B is 5. From B, you could either go to A or C. The cumulative path length upto A or C is 8. At this point, from A, you could go to E while from C you could move to D, with cumulative path lengths of 13 each and so on. Figure 9 shows the tree of possible paths from S to G, with cumulative path length (from S) marked at each node. It is evident from the figure that all paths to G have length greater than or equal to 13. We can therefore conclude that the shortest path to G from S is $SAEG$ and has length 13.

Most often we do not have any oracle suggesting the best path. So we have to think how to work without any oracle telling us what the best path is. One way is to find a path to the goal by some search technique (DFS or beam search) and use that as a reference (bound) to check if every other path is longer than that. Of course, our first search may not yield the best path, and hence we might have to change our mind about what the best path (bound) is as we keep trying to verify that the best one we got so far is in fact the best one by extending every other path to be longer than that. The intuition we work with is to always push paths that do not reach the goal until their length is greater than a path that does reach the goal. We might as well work only with shortest paths so far. Eventually, one of those will lead to the goal, with which we will almost be done, because all the other paths will be about that

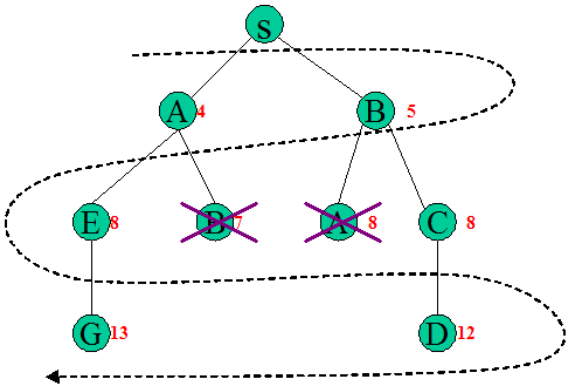


Figure 10: The pruned search tree for branch and bound search.

length too, following which we just have to keep pushing all other paths beyond the goal. This method is called branch and bound.

In the example in Figure 9, the shortest path upto B from S was of length 5 initially, which used as a bound, could eliminate the path SAB and therefore save the computation of the path $SABCD$. Similarly, the initial path to A, SA of length 4, sets a bound of 4 for the shortest path to A and can thus eliminate the consideration of the path SBA beyond A. Figure 10 illustrates the application of branch and bound to prune the BFS search tree of Figure 9. Crossed out nodes indicate search paths that are excluded because they yield paths that are longer than bounds (for the corresponding nodes). This crossing out using bounds is a variation on the theme of the dynamic programming principle.

4.7 A^* Search

The branch and bound algorithm can also be slapped on top of the hill climbing heuristic or the beam search heuristic. In each of these cases, the bound can be computed as the sum of the accumulated distance and the eucledian distance.

When the branch and bound technique is clubbed with shortest distance heuristic and dynamic programming principle, you get what is traditionally known as A^* search. Understanding A^* search is considered the culmination of optimal search. It is guaranteed to find the best possible path and is generally very fast.

5 Gaussian Discriminant Analysis

5.1 Quadratic separating surface

- Consider the following Gaussian Discriminant Classifier discussed in class. Let us say we have K classes with a multivariate Gaussian Model $\mathcal{N}(\mu_i, \Sigma_i)$ fitted for each class:

$$P(\phi(x)|C_1) = \mathcal{N}(\mu_1, \Sigma_1)$$

$$P(\phi(x)|C_i) = \mathcal{N}(\mu_i, \Sigma_i)$$

$$P(\phi(x)|C_K) = \mathcal{N}(\mu_K, \Sigma_i)$$

- Assumption: $\phi(x)$ is generated using **exactly one** $\mathcal{N}(\mu_i, \Sigma_i)$
- In the case of $K = 2$, with $P(C_1)$ and $P(C_2)$ known, separating surface will be $\{\phi(x) \mid P(C_1|\phi(x)) = P(C_2|\phi(x))\}$.

Prove that the separating surface will be **quadratic**.

SOLUTION:

- If $\phi(x) \sim \mathcal{N}(\mu_i, \Sigma_i)$ (where $\phi(x) \in \Re^m$) then

$$p(\phi(x) \mid C_i) = \frac{1}{(2\pi)^{\frac{m}{2}}|\Sigma_i|^{\frac{1}{2}}} exp\frac{-(\phi(x) - \mu_i)^T\Sigma_i^{-1}(\phi(x) - \mu_i)}{2}$$

- So, the separating surface is $\phi(x)$ such that $\{\phi(x) \mid P(C_1|\phi(x)) = P(C_2|\phi(x))\} \Longleftrightarrow \{\phi(x) \mid P(\phi(x) \mid C_1)P(C_1) = P(\phi(x) \mid C_2)P(C_2)\} \Longleftrightarrow$ after taking logs, $\phi(x)$ such that

$$-(\phi(x) - \mu_1)^T\Sigma_1^{-1}(\phi(x) - \mu_1) + (\phi(x) - \mu_2)^T\Sigma_2^{-1}(\phi(x) - \mu_2) = b$$

  where b contains terms independent of $\phi(x)$.
- This is indeed a **QUADRATIC equation**!

5.2 Linear Discriminant Analysis

- Now consider the following variant of the above Gaussian Discriminant Classifier. Let us say we have K classes with a multivariate Gaussian Model $\mathcal{N}(\mu_i, \Sigma)$ fitted for each class. That is, the covariance matrix Σ is now shared across the classes:

$$P(\phi(x)|C_1) = \mathcal{N}(\mu_1, \Sigma)$$

$$P(\phi(x)|C_i) = \mathcal{N}(\mu_i, \Sigma)$$

$$P(\phi(x)|C_K) = \mathcal{N}(\mu_K, \Sigma)$$

- Assumption: $\phi(x)$ is generated using **exactly one** $\mathcal{N}(\mu_i, \Sigma)$.
- As before, in the case of $K = 2$, with $P(C_1)$ and $P(C_2)$ known, separating surface will be $\{\phi(x) \mid P(C_1|\phi(x)) = P(C_2|\phi(x))\}$.

1. Q: Prove that the separating surface will now be **linear**.

   SOLUTION:

   - If $\phi(x) \sim \mathcal{N}(\mu_i, \Sigma)$ (where $\phi(x) \in \Re^m$) then

$$p(\phi(x) \mid C_i) = \frac{1}{(2\pi)^{\frac{m}{2}}|\Sigma|^{\frac{1}{2}}} exp\frac{-(\phi(x) - \mu_i)^T\Sigma^{-1}(\phi(x) - \mu_i)}{2}$$

- So, the separating surface is $\phi(x)$ such that $\{\phi(x) \mid P(C_1|\phi(x)) = P(C_2|\phi(x))\}$ $\Longleftrightarrow$ $\{\phi(x) \mid P(\phi(x) \mid C_1)P(C_1) = P(\phi(x) \mid C_2)P(C_2)\}$ $\Longleftrightarrow$ after taking logs, $\phi(x)$ such that

$$-(\phi(x) - \mu_1)^T\Sigma^{-1}(\phi(x) - \mu_1) + (\phi(x) - \mu_2)^T\Sigma^{-1}(\phi(x) - \mu_2) = b$$

where b contains terms independent of $\phi(x)$
$\Longleftrightarrow$ ...the above expression can actually be simplified by canceling out he terms involving $\phi^T(x)\Sigma^{-1}\phi(x)$

$$-\phi^T(x)\Sigma^{-1}\phi(x) \ldots - \phi^T(x)\Sigma^{-1}\phi(x) = b$$

to finally give

$$2\phi^T(x)\Sigma^{-1}\mu_1 + 2\phi^T(x)\Sigma^{-1}\mu_2 - \mu_1^T(x)\Sigma^{-1}\mu_1 - \mu_2^T(x)\Sigma^{-1}\mu_2 = b$$

that is,

$$2\phi^T(x)\Sigma^{-1}\mu_1 + 2\phi^T(x)\Sigma^{-1}\mu_2 - \mu_1^T(x)\Sigma^{-1}\mu_1 = b'$$

which is a **LINEAR equation**! Here, $b' = \mu_1^T(x)\Sigma^{-1}\mu_1 + \mu_2^T(x)\Sigma^{-1}\mu_2 + b$ is independent of $\phi(x)$.

2. Q: What will be the maximum likelihood estimates for μ_i and Σ in this new case of different means but shared covariance matrix?

   SOLUTION: The Maximum Likelihood estimate for $\hat{\mu}_i$ will be the same as that for the Quadratic Discriminant Analysis, but that for a shared and single covariance estimate $\hat{\Sigma}$ will correspond to the average of covariance matrix estimates across examples from all the classes

$$\hat{\mu}_i = \frac{1}{n_i}\sum_{j=1}^{n_i}\phi(x_j^i)$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{K}\sum_{j=1}^{n_i}(\phi(x_i^j) - \mu_i)(\phi(x_i^j) - \mu_i)^T$$

6 EM Algorithm for Mixture of Gaussians

Q: Show that the following algorithm for estimating the mean μ_i, the covariance matrix Σ_i and mixture components π_i for a mixture of Gaussians is an instance of the general EM algorithm

SOLUTION:

Initialize $\mu_i^{(0)}$ to different random values and $\Sigma_i^{(0)}$ to I. Now iterate between the following **E Step** and **M Steps**:

E Step:

1. For the posterior $p(z_i \mid \phi(x_j), \mu, \Sigma)$

$$p^{(t+1)}(z_i \mid \phi(x_j), \theta) = \frac{\pi_i \mathcal{N}\left(\phi(x); \mu_i^{(t)}, \Sigma_i^{(t)}\right)}{\sum_{l=1}^{K} \pi_l \mathcal{N}\left(\phi(x); \mu_l^{(t)}, \Sigma_l^{(t)}\right)}$$

M Steps:

1. For the prior π_i

$$\pi_i^{(t+1)} = \frac{1}{n} \sum_{j=1}^{n} p^{(t+1)}(z_i \mid \phi(x_j), \theta)$$

2. For μ_i

$$\mu_i^{(t+1)} = \frac{\sum_{j=1}^{n} p^{(t+1)}(z_i \mid \phi(x_j), \theta)\phi(x_j)}{\sum_{j=1}^{n} p^{(t+1)}(z_i \mid \phi(x_j), \theta)}$$

3. For Σ_i

$$\Sigma_i^{(t+1)} = \frac{\sum_{j=1}^{n} p^{(t+1)}(z_i \mid \phi(x_j), \theta)\left(\phi(x_j) - \mu_i^{(t+1)}\right)\left(\phi(x_j) - \mu_i^{(t+1)}\right)^T}{\sum_{j=1}^{n} p^{(t+1)}(z_i \mid \phi(x_j), \theta)}$$

Q: Note that this algorithm is for the Mixture of Gaussians assuming a difference covariance matrix Σ_i for each class C_i. What will be the algorithm like, if we assume a shared covariance matrix Σ across all classes (that is, the Linear Discriminant Analysis discussed in Section 1.2)?

Solution: We will simply build on the solution to the Linear Discriminant case from Section 2.1 and simply replace multiple class-specific estimates Σ_i with a single estimate Σ:

7 (Optional) Bayesian Inference from Multinomial to Naive Bayes

First we summarize the conjugate prior, MLE and Bayesian estimation for Multinomial

- In the case of the Multinomial distribution (extension to the binomial distribution) a variable X could assume one of t possible values $V_1, V_2 \ldots V_t$ with parameters $\Pr(X = V_j) = \mu_j$. Each observation $\mathbf{X}_k$ (for $k \in [1, n]$) is modeled as a vector $\mathbf{X}_k = [X_{k,1} \ldots X_{k,j} \ldots X_{k,t}]$ with $X_{k,j} = 1$ if and only if value of $\mathbf{X}_k$ was observed to be V_j and $X_{k,j} = 0$ otherwise. Eg: In the case of the toss of dice, $t = 6$.

- The maximum likelihood estimate for the mean is given by:

$$\hat{\mu}_j = \frac{\sum_{k=1}^{n} X_{k,j}}{n} = \frac{n_j}{n} \tag{1}$$

  where, given n iid observations of a multinomial random variable X, $n_j = \sum_{k=1}^{n} X_{k,j}$ is the number of times $X = V_j$ was observed,

- Conjugate prior follows $\text{Dir}(\alpha_1 \dots \alpha_n)$

- Posterior is $\text{Dir}(\dots \alpha_l + \sum_{k=1}^{n} X_{k,j} \dots)$

- The expectation of μ under $\text{Dir}(\alpha_1 \dots \alpha_n)$ is given by:

$$E\left[\mu\right]_{Dir(\alpha_1 \dots \alpha_n)} = \left[\frac{\alpha_1}{\sum \alpha_l} \cdots \frac{\alpha_1}{\sum \alpha_l}\right] \tag{2}$$

- The (posterior) expectation of μ under $\text{Dir}(\dots \alpha_j + \sum_{k=1}^{n} X_{k,j} \dots)$ is given by:

$$E\left[\mu\right]_{Dir(\dots \alpha_j + \sum_{k=1}^{n} X_{k,j} \dots)} = \left[\frac{\alpha_1 + \sum_k X_{k,1}}{\sum \alpha_l + n} \cdots \frac{\alpha_j + \sum_k X_{k,j}}{\sum \alpha_l + n} \dots\right] \tag{3}$$

Now recall that we extended single Multinomial to Multinomial Naive Bayes that has class conditioned independent features, each of which is Multinomial. We also discussed Maximum Likelihood estimation of Multinomial Naive Bayes. All of that is summarized below:

- $< X_k, C_i >$: Tuple with example X_k belonging to class C_i. $\Pr(C_i)$ is prior probability of class C_i.

- $\phi_1(X_k), \dots, \phi_m(x_k)$: The feature vector for X_k

- $P(\phi_q(x)|C_i) \sim Mult(\mu^q_{1,i} \dots \mu^q_{t_q,i})$; that is, each feature ϕ_q follows multinomial distribution Bayes

  1. $[V^1_1 \dots V^1_{t_1}] \dots [V^q_1 \dots V^q_{t_q}] \dots [V^m_1 \dots V^m_{t_m}]$: Set of values that could be taken by each of $\phi_1, \phi_2 \dots \phi_m$ respectively
  2. $[\mu^1_{1,i} \dots \mu^1_{t_1,i}] \dots [\mu^q_{1,i} \dots \mu^q_{t_q,i}] \dots [\mu^m_{1,i} \dots \mu^m_{t_m,i}]$: Parameters for each of $\phi_1, \phi_2 \dots \phi_m$ respectively for class C_i

- $P(\phi_1(x) \dots \phi_m(x)|C_i) = \prod_{q=1}^{m} P(\phi_q(x)|C_i)$: Feature are independent given the class

- Maximum Likelihood Estimate for Naive Bayes: Let

$$\begin{aligned}
\#C_i = &\text{ No. of times } c(X_k) = C_i \text{ across all } k\text{'s in the dataset} \\
n^q_{j,i} = &\text{ No. of times } \phi_q(X_k) = V_j \text{ and } c(X_k) = C_i \text{ across all the } k\text{'s} \\
n^q_{j,i} = &\sum_k \delta\big(\phi_q(X_k), V^q_j\big)\delta\big(c(Xk_i), C_i\big)
\end{aligned}$$

then

$$\hat{\mu}^q_{j,i} = \frac{n^q_{j,i}}{\sum_{j'=1}^{n} n^q_{j',i}}$$
$$\widehat{Pr}_{c_i} = \frac{\#C_i}{\sum_{i'} \#C_{i'}}$$

Q: Extend Bayesian Inference using the Dirichlet prior from Multinomial to Naive Bayes

Solution: The setting for Naive Bayes with Dirichlet prior on Multivariate Bernoulli distribution is as follows

- For each data point X_k which belongs to class C_i there are a set of m features given by $\phi_1(X_k) \dots \phi_q(X_k) \dots \phi_m(X_k)|C_i$
- Each parameter (vector) $\mu^q_i = [\mu^q_{1,i} \dots \mu^q_{j,i} \dots \mu^q_{t_q,i}]$ corresponding to a feature $\phi_q(.)$ has a probability distribution given by
  $p(\mu^1_i) \sim Dir(a^1_{1,i} \dots a^1_{j,i} \dots a^1_{t_1,i}) \dots$
  $p(\mu^q_i) \sim Dir(a^q_{1,i} \dots a^q_{j,i} \dots a^q_{t_q,i}) \dots$
  $p(\mu^m_i) \sim Dir(a^m_{1,i} \dots a^m_{j,i} \dots a^m_{t_m,i})$
  where $a^q_{j,i}$ are the multivariate dirichlet prior parameters
- Let $n^q_{j,i}$ be the number of times attribute $\phi_q(.)$ was observed with value V^q_j amongst examples belonging to class C_i.
- Then, from our previous analysis of Bayesian estimation for Multinomial[6],
  $p(\mu^1_i \mid D) \sim Dir(a^1_{1,i} + n^1_{1,i} \dots a^1_{j,i} + n^1_{j,i} \dots a^1_{t_1,i} + n^1_{t_1,i}) \dots$
  $p(\mu^q_i \mid D) \sim Dir(a^q_{1,i} + n^q_{1,i} \dots a^q_{j,i} + n^q_{j,i} \dots a^q_{t_q,i} + n^q_{t_q,i}) \dots$
  $p(\mu^m_i \mid D) \sim Dir(a^m_{1,i} + n^m_{1,i} \dots a^m_{j,i} + n^m_{j,i} \dots a^m_{t_m,i} + n^m_{t_m,i})$.

8 Logistic Regression and Maximum Entropy Classifier

The Logistic Regression classifier also goes under another name called the Maximum Entropy Classifier where the goal is to prefer the most uniform models that also satisfy any given constraints. More specifically, the goal in Maximum Entropy Classification is to find the probability distribution $\Pr(Y = c|\phi(\mathbf{x}))$ for $c = [1...K]$ that maximizes the entropy

$$E(\Pr(.))) = -\left[\frac{1}{m}\sum_{c=1}^{K}\sum_{i=1}^{m} \Pr\left(Y = c \left| \phi\left(\mathbf{x}^{(i)}\right)\right.\right) \log \Pr\left(Y = c \left| \phi\left(\mathbf{x}^{(i)}\right)\right.\right)\right] \tag{4}$$

such that, for every feature ϕ_j for $j = [1...n]$ and every class $c = [1..K]$

$$\sum_{i=1}^{m} \phi_j\left(\mathbf{x}^{(i)}\right) \Pr\left(Y = c \left| \phi\left(\mathbf{x}^{(i)}\right)\right.\right) = \sum_{i=1}^{m} \phi_j\left(\mathbf{x}^{(i)}\right) \delta(y^{(i)}, c) \tag{5}$$

[6]See page 12 of `http://23.253.82.180/course/307/865/2238`.

where $\delta(y^{(i)}, c) = 1$ if and only if $y^{(i)} = c$ and $\delta(y^{(i)}, c) = 0$ otherwise.

Now answer the following questions

1. Interpret the optimization problem (4) and the constraint (5) in plain English words while clearly stating the intuition.

2. Prove that the probability distribution $\Pr(Y = c|\phi(\mathbf{x}))$ of the solution that maximizes the entropy in (4) subject to the constraint set (5) turns out to have the form of logistic regression[7].

3. How can one introduce regularization into the entropy objective (4)? What will be the result of doing so on the form of the resulting probability distribution $\Pr(Y = c|\phi(\mathbf{x}))$ at optimality?

Solution:

1. The main idea behind maximum entropy is that one should prefer the most uniform models that also satisfy the data driven constraints which are captured in (5). For example, consider a four-way text classification task where we are told only that on an average 60% of documents with the word "advisor" in them are in the student class. Thus, given a document with "advisor" in it, we would say it has a 60% chance of being a student document, and a 40% chance for each of the other three classes. If a document does not have"advisor" we would guess the uniform class distribution, that is, 25% each as per (4). This is exactly the maximum entropy model (4) that conforms to our known constraint (5)

2. The key is to understand that the optimization problem in (4) is for the set of variables p_{ci} where $p_{ci} = \Pr\left(Y = c|\phi\left(\mathbf{x}^{(i)}\right)\right)$. Simplfying (4) and (5) in terms of these variables

$$E\left(\Pr(.)\right)) = -\left[\frac{1}{m}\sum_{c=1}^{K}\sum_{i=1}^{m} p_{ci}\log p_{ci}\right] \tag{6}$$

such that, for every feature ϕ_j for $j = [1...n]$ and every class $c = [1..K]$

$$\sum_{i=1}^{m}\phi_j\left(\mathbf{x}^{(i)}\right)p_{ci} = \sum_{i=1}^{m}\phi_j\left(\mathbf{x}^{(i)}\right)\delta(y^{(i)}, c) \tag{7}$$

and for each $i = [1..m]$

$$\sum_{c=1}^{K} p_{ci} = 1 \tag{8}$$

Let w_{cj} be the lagrange multiplier corresponding to (7) for a specific value of c and j and λ be the lagrange multiplier corresponding to (8). We know that a necessary

[7]Optional point of reference: Recall Solution to Problem 3 of Tutorial 7, where we restated the problem of finding optimal solution $\mathbf{w}$ to the regularized cross entropy as that of finding a function from a function space that is smooth enough and minimizes the objective. In this problem, we are similarly seeking a characterization of family of probability distributions that maximize the entropy.

condition for optimality of (6) subject to (7) and (8) is that the gradient (or every partial derivative) of the Langrange with respect to all the p_{ci} should be 0. That is,

$$-\log p_{ci} - 1 + \sum_{j=1}^{m} w_{cj}\phi_j\left(\mathbf{x}^{(i)}\right) + \lambda = 0$$

that is,

$$\log p_{ci} = \lambda - 1 + \sum_{j=1}^{m} w_{cj}\phi_j\left(\mathbf{x}^{(i)}\right)$$

That is,

$$p_{ci} = \exp\left(\lambda - 1 + \sum_{j=1}^{m} w_{cj}\phi_j\left(\mathbf{x}^{(i)}\right)\right) = \exp\left(\lambda - 1\right) \times \exp\left(\sum_{j=1}^{m} w_{cj}\phi_j\left(\mathbf{x}^{(i)}\right)\right)$$

Now from (8),

$$\sum_{k=1}^{K} p_{ki} = \sum_{k=1}^{K} \exp\left(\lambda - 1\right) \times \exp\left(\sum_{j=1}^{m} w_{kj}\phi_j\left(\mathbf{x}^{(i)}\right)\right) = \exp\left(\lambda - 1\right) \times \sum_{k=1}^{K} \exp\left(\sum_{j=1}^{m} w_{kj}\phi_j\left(\mathbf{x}^{(i)}\right)\right) = 1$$

Therefore

$$\exp\left(\lambda - 1\right) = \frac{1}{\sum_{k=1}^{K} \exp\left(\sum_{j=1}^{m} w_{kj}\phi_j\left(\mathbf{x}^{(i)}\right)\right)}$$

Thus,

$$\Pr\left(Y = c | \phi\left(\mathbf{x}^{(i)}\right)\right) = p_{ci} = \frac{\exp\left(\sum_{j=1}^{m} w_{cj}\phi_j\left(\mathbf{x}^{(i)}\right)\right)}{\sum_{k=1}^{K} \exp\left(\sum_{j=1}^{m} w_{kj}\phi_j\left(\mathbf{x}^{(i)}\right)\right)}$$

Also, $\sum_{j=1}^{m} w_{cj}\phi_j\left(\mathbf{x}^{(i)}\right) = \mathbf{w}_c^T \phi\left(\mathbf{x}^{(i)}\right)$. Thus

$$\Pr\left(Y = c | \phi\left(\mathbf{x}^{(i)}\right)\right) = p_{ci} = \frac{\exp\left(\mathbf{w}_c^T \phi\left(\mathbf{x}^{(i)}\right)\right)}{\sum_{k=1}^{K} \exp\left(\mathbf{w}_k^T \phi\left(\mathbf{x}^{(i)}\right)\right)}$$

which is exactly of the form of Logistic regression!

3. Any one of the two options could serve the purpose.

- Note that maximum entropy without any constraints results when the distribution is uniform. Thus, maximizing entropy is the same as minimizing the (probabilistic) distance of the target desired distribution from a uniform prior. One could therefore introduce regularization into the entropy maximization in (4) by changing the objective from maximizing entropy (that is minimizing the distance of the target distribution from the uniform distribution) to **minimizing the distance of the target distribution from some prior distribution** (such as Gaussian), subject to the constrains in (5). Specifically, such one such distance is called the **KL divergence**.
- One could introduce regularization into the entropy maximization via the constraints (5) by adding a small factor η to each constraint on the right hand side. This amounts to the assumption that by default certain additional number of observations η for each feature need to be accounted for, even if the number of such observations in the data are fewer. This allows for sparse features to be accounted for.

9 Car(s) and Vehicle(s)

In this problem, assume that every input image has the same size: 1024×1024**.**

1. **Task 1:** In the class we discussed the Convolutional and pooling layers of a CNN and motivated the kernel (sparse interaction, patches, strides) for classification of images. Suppose our input image contains exactly one vehicle (either a bicycle or a car or a motorbike or any one of N vehicles) and we have trained a CNN-based network to distinguish between images based on the kind of vehicle that the image contains. Specifically, the ouput layer of our network consists of a soft-max layer that can classify an image into one of N vehicular categories. Thus, our CNN-based network is trained to distinguish that the image in the first row, first column of Figure 11 is a car and that the image in the first row, second column of Figure 11 is a motor-bike.

   Roughly depict what the different components/operators of our trained CNN-based network would extract in the intermediate steps to help distinguish between images containing different vehicles, and handle different locations[8] of an object, etc, on each of the 3 rows of Figure 12. Here all the images are actually reproduced from the first row of Figure 11.

[8] that is, give correct results even when the location of an object in the image changes, etc.

Task 1: Detecting Image Containing Single Object

Task 2: Detecting Separated Objects

Task 3: Detecting Overlapping Objects

Figure 11: List of 1024×1024 images for the three tasks in this problem.

Solution:

Expect depiction of the Kernel/filter on the image and description in pictures/words of how (i) local connections help detect, vertical and horizontal edges, parts of cars such as wheels/body/chasis etc.. (i) weight sharing helps detect the same vehicle/car or its parts in different locations (iii) equivariance ensures that moving an object followed by convolution is the same as convolution followed by moving the object and finally (iv) max pooling does down sampling so that large contiguous sections such as the body of the car can have a more compact representation.

Figure 12: Explain by rough sketching how the CNN-based network helps address/correctly classify the images here. Recall that each image is of the same size, viz., 1024×1024

2. **Task 2:** Suppose we now have images that contain a combination of multiple vehicles but well separated from each other within the same image, as illustrated in the images

in the second row of Figure 11. How would you modify your CNN-based network from Task 1 to detect different objects in the same image? Identify any important limitation in your solution. 4

Solution: Naive is to change the output layer (like in RNN) to predict not just one class, but one class for each pixel. Thus, change the output layer to a 1024×1024 matrix of softmax units. Possible problem is that we are making prediction at every pixel - which can be expensive. Optional: A better alternative (not expected as answer) is to predict bounding boxes and restrict the prediction of objects at the level of bounding boxes. So smaller middle layer, upsamping, Bounding box etc is what RCNN and Faster RCNN do.

3. **Task 3:** Now suppose we need to detect overlapping vehicles within the same image. That is, we now have images that contain a combination of multiple vehicles that are NOT well separated from each other within the same image, as illustrated in the images in the third row of Figure 11. This kind of phenomenon is called occlusion. Suggest how you will modify the CNN-based network to handle Task 3. Identify any important limitation in your solution. 3

   Solution: The solution to Task 2 will detect exactly one vehicle at every pixel. So a simple modification can be to replace the 1024×1024 matrix of softmax units with a 1024×1024 matrix, with each cell now being an N dimensional vector of sigmoids so that we can deal with multi-label classification. Extra information: This is technically also called a Mask. Thus, each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters.

10 Bayesian Network: Learning with Missing Data (Completely Optional)

Consider the directed graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{G} \rangle$ where $\mathcal{V} = \{H, A, B, C\}$ and $\mathcal{E} = \{(H, A), (H, B), (H, C)\}$. Consider the dataset $\mathcal{D}$ provided in the table below:

H	A	B	C	w
?	a_0	b_0	c_0	14
?	a_0	b_0	c_1	11
?	a_0	b_1	c_0	20
?	a_0	b_1	c_1	20
?	a_1	b_0	c_0	5
?	a_1	b_0	c_1	5
?	a_1	b_1	c_0	11
?	a_1	b_1	c_1	14

where ? stands for missing observations. Consider the following random initial probabilities for the example execution of the expectation maximization algorithm.

p_H	h_0	h_1
	0.3	0.7

$p_{A\|H}$	h_0	h_1
a_0	0.4	0.6
a_1	0.6	0.4

$p_{B\|H}$	h_0	h_1
b_0	0.7	0.8
b_1	0.3	0.2

$p_{C\|H}$	h_0	h_1
c_0	0.2	0.5
c_1	0.8	0.5

Compute the probabilities (parameters) of the bayesian network after one iteration of the EM algorithm.

Tutorial 9

CS 337 Artificial Intelligence & Machine Learning, Autumn 2019

October, 2019

1 Example: Optimal Action Plan

Lets define a state machine with two states s_1, s_2. Formally, in the example depicted in Figure 1 we have:

time: $T = \{1...N\}$

states: $S = \{s_1, s_2\}$

possible actions in each state: $A_{s_1} = \{a_{11}, a_{12}\}, A_{s_2} = \{a_{21}\}$

reward: $r(s_1, a_{11}) = 5, r(s_1, a_{12}) = 10, r(s_2, a_{21}) = -1$

transition function: $\Pr(s_1|s_1, a_{11}) = 0.5, \Pr(s_2|s_1, a_{11}) = 0.5, \Pr(s_2|s_1, a_{12}) = 1, \Pr(s_2|s_2, a_{22}) = 1$

Now we want to maximize the return. First we compare two deterministic policies:

π_1- always chooses a_{11} when in state s_1.

π_2 - always chooses a_{12} when in state s_1.

Determine the best policy, that is the policy that maximizes the value function. Explain the procedure.

Solution:

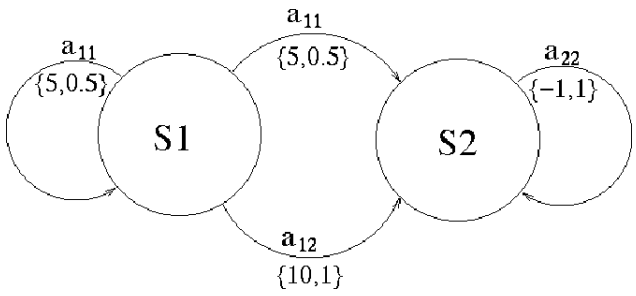


Figure 1: State Diagram

As an aside, recall that $V_\pi(s, N)$ is the expected return of policy Π in N time steps.

$$\begin{aligned} V_{\pi_1}(s_1, 3) &= \frac{1}{2}(5-1+0) + \frac{1}{2}(5+5+0) = 7 \\ V_{\pi_2}(s_1, 3) &= 10 - 1 = 9 \\ V_{\pi_1}(s_1, 5) &= \frac{1}{2}(5-3) + \frac{1}{4}(10-2) + \frac{1}{8}(15-1) + \frac{1}{8}(20) = 7.25 \\ V_{\pi_2}(s_1, 5) &= 10 - 3 = 7 \end{aligned}$$

Notice how in shorter time frames π_2 is the preferable policy, while in longer ones, π_1 brings on the higher returns.

Finding a π that maximizes $V_\pi(s, N)$ is called a finite horizon problem. When $N \to \infty$, we get the infinite horizon problem.

We solve by looking first at the reward gained in the last step, for N=2:

$$\begin{aligned} Q_2(s_1, a_{11}) &= 5 \\ Q_2(s_1, a_{12}) &= 10 \end{aligned}$$

In this case:

$$\begin{aligned} V_2(s_1) &= 10 \\ V_2(s_2) &= -1 \end{aligned}$$

When N=2, The optimal action from s_1 is a_{12}.

For N=3, we use the optimal policy for N=2, after we do one action.

$$\begin{aligned} Q_3(s_1, a_{11}) &= 5 + \frac{1}{2}V_2(s_1) + \frac{1}{2}V_2(s_2) = 5 + \frac{1}{2} * 10 + \frac{1}{2} * (-1) = 9.5 \\ Q_3(s_1, a_{12}) &= 10 + V_2(s_2) = 10 - 1 = 9 \end{aligned}$$

Hence, when N=3 the optimal action from s_1 is a_{11}.

2 MDP and Optimal Policy for the Recruitment Problem

A manager has to recruit a new employee and he can serially interview a finite group of candidates. There is a total order defined on the candidates' fitness to the opening, and there are no two employees with the same skill level. The manager is able to sort the candidates' fitness level after a short interview. After each interview the manager has two alternatives:

- recruit the last interviewed candidate
- continue to the next interview (and give up the chance of recruiting the previous candidate)

The goal is to maximize the probability of recruiting the best candidate. Formulate this as a Markov Decision Process (MDP) and develop its optimal policy. Analyze what happens when N (size of candidate set being interviewed) is 5. What happens when $N \to \infty$?

Solution: We will first construct a corresponsing MDP for the problem, and then develop the optimal policy for it.
Let $A = \{Continue, QuitAndHire\}$ the possible actions, where $Continue$ stands for 'continue' and $QuitAndHire$ for 'quit and recruit'. Let $S = \{MaxSoFar, Other, 1, 0\}$ be the group of states of the MDP, standing for:

- MaxSoFar - the last interviewed candidate was the best so far
- Other - the last interviewed candidate was NOT the best so far
- 1 - the last interviewed candidate was THE best candidate in the entire group and was hired
- 0 - the last interviewed candidate was NOT the best candidate in the entire group and was hired

The resultant MDP, uses the following transition probabilities:

- $q_t = Prob[\max\{x_1, ..., x_t\} = \max\{x_1, ..., x_N\}] = \frac{t}{N}$
- $r_t = Prob[x_{t+1} \geq \max\{x_1, ..., x_t\}] = \frac{1}{t+1}$

where N is the finite time horizon (the size of the candidates group) and t is the current time (the number of candidates interviewed so far).
Writing the optimality equations for this problem we get:

$$U_t^*(1) = 1, \;\; U_t^*(0) = 0, \;\; U_N^*(MaxSoFar) = U_N^*(Other) = 0$$

$$\begin{aligned} U_t^*(Other) &= \max\{0, r_t U_{t+1}^*(MaxSoFar) + (1 - r_t) U_{t+1}^*(Other)\} \\ &= r_t U_{t+1}^*(MaxSoFar) + (1 - r_t) U_{t+1}^*(Other) \\ U_t^*(MaxSoFar) &= \max\{q_t U_{t+1}^*(1), r_t U_{t+1}^*(MaxSoFar) + (1 - r_t) U_{t+1}^*(Other)\} \\ &= \max\{q_t \cdot 1, r_t U_{t+1}^*(MaxSoFar) + (1 - r_t) U_{t+1}^*(Other)\} \\ &= \max\{q_t, U_t^*(Other)\} \end{aligned}$$

Assigning the terms for q_t and r_t we get:

$$U_t^*(Other) = \frac{1}{t+1} U_{t+1}^*(MaxSoFar) + \frac{t}{t+1} U_{t+1}^*(Other) \tag{1}$$

$$U_t^*(MaxSoFar) = \max\{\frac{t}{N}, U_t^*(Other)\} \tag{2}$$

An optimal decision rule has the following properties:

- In $s = Other$, always choose $a_s = Continue$ (otherwise the return is promised to be zero)
- In $s = MaxSoFar$, choose $a_s = Continue$ if $\frac{t}{N} < U_t^*(Other)$ and $a_s = QuitAndHire$ otherwise (in order to maximize $U_t^*(MaxSoFar)$)

The optimal policy is of the form:

- Interview τ candidates, performing $a_t = Continue,\ t \leq \tau$
- Quit and hire the first candidate after time τ, which is the best so far

2.1 When $N = 5$

Let $N = 5$, we get
$\frac{1}{3} + \frac{1}{4} < 1$ and $\frac{1}{2} + \frac{1}{3} + \frac{1}{4} > 1$
and therefore we choose $\tau = 2$.

2.2 When $N \to \infty$

Let $N \to \infty$, we get

$$\sum_{j=\tau}^{N-1} \frac{1}{j} \sim \ln \frac{N}{\tau}.$$

We therefor shearch for τ such that
$\ln \frac{N}{\tau+1} < 1$ and $\ln \frac{N}{\tau} > 1$,
which leads to:

$$\tau \sim \frac{N}{e}$$

OPTIONAL Optimality Proof We will now show that the described policy agrees with the optimality equations.
We start by showing that if there is a time τ such that $\frac{\tau}{N} < U_\tau^*(MaxSoFar)$ then $\forall t, t < \tau$ we get $\frac{t}{N} < U_t^*(MaxSoFar)$. That is, if there exists a time $t = \tau$ in which it is preferred to $Continue$, then for each earlier time it is also preferred to continue. Later on, we will show that such a time, τ, does exist.
Let $\frac{t}{N} < U_t^*(MaxSoFar)$, then according to equation 2 $U_\tau^*(MaxSoFar) = U_\tau^*(Other)$. Performing backword induction steps, and using equations 1 and 2 we show that for $t < \tau$ the inequality remains true:
$U_{\tau-1}^*(Other) = \frac{1}{\tau} U_\tau^*(MaxSoFar) + \frac{\tau-1}{\tau} U_\tau^*(Other) = U_\tau^*(Other) > \frac{\tau}{N}$
$U_{\tau-1}^*(MaxSoFar) = \max\{\frac{\tau-1}{N}, U_{\tau-1}^*(Other)\} > \frac{\tau}{N} > \frac{\tau-1}{N}$
We now show that for $t > \tau$ and $s = MaxSoFar$, it is preferred to $QuitAndHire$.
Let $t > \tau$, then according to the first half of the proof,
$U_t^*(MaxSoFar) = \frac{t}{N}$

and,

$$\begin{aligned} U_t^*(Other) &= \frac{1}{t+1}U_{t+1}^*(MaxSoFar) + \frac{t}{t+1}U_{t+1}^*(Other) \\ &= \frac{1}{N} + \frac{t}{t+1}U_{t+1}^*(Other) \\ &= \frac{1}{N} + \frac{t}{t+1}\frac{1}{N} + \frac{t}{t+2}\frac{1}{N} + ... \\ &= \frac{1}{N}\sum_{i=0}^{N-t}\frac{t}{t+i} \sim \ln(\frac{N}{t}) \end{aligned}$$

We conclude this proof by showing that such a τ exists, that is, for $N \geq 2$ there exists $\tau \geq 1$ that meets the above requiremints. Let us assume $\tau = 0$ we get:
$U_1^*(Other) = \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{N} > \frac{1}{2} \geq \frac{1}{N} = U_1^*(MaxSoFar) \geq U_1^*(Other)$
which is a circular inequality, and thus leads to cnotradiction.
Note that for $N \geq 2$ we always get $\tau \geq 1$:
$U_1^*(Other) = ... = U_\tau^*(Other)$
||
$U_1^*(MaxSoFar) = ... = U_\tau^*(MaxSoFar)$
and for $t > \tau$
$U_t^*(MaxSoFar) = \frac{t}{N}$
$U_t^*(Other) = \frac{t}{N}(\frac{1}{t} + \frac{1}{t+1} + ... + \frac{1}{N-1})$.
We therefore choose *Continue* if $\frac{1}{t} + ... + \frac{1}{N-1} > 1$ and *QuitAndHire* otherwise.

3 Bound concerning Value Iteration

Consider an MDP $(S, A, \Pr, R, \gamma)$, with notations being as usual and discount factor $\gamma \in [0, 1)$. Recall that the Value Iteration algorithm produces a sequence V_0,V_1, V_2, ... each element being a mapping from S to R, which converges to the optimal value function V_*

1. For $t = 0, 1, ...$, write down how V_{t+1} is obtained from V_t

2. Assume that there is a scalar $R_{max} > 0$ such that each individual reward obtained from R lies in $[0, R_{max}]$. Also assume that Value Iteration is initialised with the zero vector: that is, $V_0 = 0$. Show that for $t = 0, 1, ...$ and for $s \in S$:

$$V_*(s) - V_t(s) \leq \frac{\gamma^t R_{max}}{1-\gamma}$$

Solution:

1. For $t = 0, 1, ...$ and $s \in S$:

$$V_{t+1}(s) \Leftarrow \max_{a \in A} \sum_{s' \in S} \Pr(s'|a, s)R(s, a, s') + \gamma V_t(s')$$

2. We prove the result by induction on t. For $t = 0$, we have to show that for $s \in S$,

$$V^*(s) \leq \frac{R^{max}}{1-\gamma}$$

which is evident since $V^*(s)$ is the expected infinite discounted reward obtained by a policy, in this case an optimal policy. Even if each reward is maximum, $V^*(s)$ can at most be $R^{max} + \gamma R^{max} + \gamma^2 R_{max} + ... = \frac{R_{max}}{1-\gamma}$

Assume the result is true for some $t \geq 0$. Take π^* to be any optimal policy. We have, for $s \in S$:

$$\begin{aligned} V^{t+1}(s) &= \max_{a \in A} \sum_{s' \in S} \Pr(s'|a,s)\{R(s,a,s') + \gamma V^t(s')\} \\ &\geq \sum_{s' \in S} \Pr(s'|\pi^*(s),s)\left\{R(s,\pi^*(s),s') + \gamma V^t(s')\right\} \\ &\geq \sum_{s' \in S} \Pr(s'|\pi^*(s),s)\left\{R(s,\pi^*(s),s') + \gamma\left(V^*(s') - \frac{\gamma^t R_{max}}{1-\gamma}\right)\right\} \end{aligned} \quad (3)$$

wherein the last step applies the induction hypothesis and also the fact that the transition probabilities are non-negative. By expanding out, we get

$$\begin{aligned} V^{t+1}(s) &\geq \sum_{s' \in S} \Pr(s'|\pi^*(s),s)\left\{R(s,\pi^*(s),s') + \gamma V^*(s')\right\} - \sum_{s' \in S} \Pr(s'|\pi^*(s),s)\left\{\frac{\gamma^{t+1} R_{max}}{1-\gamma}\right\} \quad (4) \\ &= V^*(s) - \frac{\gamma^{t+1} R_{max}}{1-\gamma} \quad (5) \end{aligned}$$

obtained by invoking Bellman's Equations for π^* and equating the sum of transition probabilities from $(s, \pi^*(s))$ to 1.

A second, direct approach would be to split

$$V^*(s) = \mathbb{E}_{\pi^*}\left\{r^0 + \gamma r^1 + \gamma^2 r^2 + ... \,\middle|\, s^0 = s\right\}$$

into a sum of

$$T_1 = \mathbb{E}_{\pi^*}\left\{r^0 + \gamma r^1 + \gamma^2 r^2 + ... \gamma^{t-1} r^{t-1} \,\middle|\, s^0 = s\right\}$$

and

$$T_2 = \mathbb{E}_{\pi^*}\left\{\gamma^t r^t + \gamma^{t+1} r^{t+1} + ... \,\middle|\, s^0 = s\right\}$$

It so happens that $V^t(s)$ is the maximum expected discounted t-step reward that can be possibly obtained; in general one would have to follow a non-stationary (time-dependent) policy in order to achieve it. T_1 is the expected discounted t-step reward obtained by following π^* and so cannot exceed $V^t(s)$. Since each individual reward is upper-bounded by R_{max}, we see that T_2 is at most $\frac{\gamma^t R_{max}}{1-\gamma}$. Hence, $V^*(s) \leq V^t(s) + \frac{\gamma^t R_{max}}{1-\gamma}$

Tutorial 10

CS 337 Artificial Intelligence & Machine Learning, Autumn 2019

November, 2019

1 Adaboost

Recall the main idea behind Adaboost.Let $C_t(\mathbf{x}) = \sum_{j=1}^{t} \alpha_j T_j(\mathbf{x})$ be the boosted linear combination of classifiers until t^{th} iteration. Let the error to be minimized over α_t be the sum of its exponential loss on each data point,

$$\mathbf{E}_t = \sum_{i=1}^{m} \exp\left(-y^{(i)} C_t\left(\mathbf{x}^{(i)}\right)\right) = \sum_{i=1}^{m} \exp\left(-\left(y^{(i)} \sum_{j=1}^{t} \alpha_j T_j(\mathbf{x}^{(i)})\right)\right)$$

Prove the following claims for Adaboost

1. Claim1: The error that is the sum of exponential loss on each data point is an upper bound on the simple sum of training errors on each data point

   Solution:

   The simple sum of training errors on each data point is

   $$\sum_{i=1}^{m} \delta\left(y^{(i)} \neq sign\left(C_t\left(\mathbf{x}^{(i)}\right)\right)\right) \leq \sum_{i=1}^{m} \exp\left(\delta\left(y^{(i)} \neq sign\left(\sum_{i=1}^{m} \alpha_i T_i(\mathbf{x}^{(i)})\right)\right)\right)$$

   We next note that the exponential function[1] is a convex function. That is,

   $$\exp\left(\sum_{i=1}^{m} \beta_i r_i\right) \leq \sum_{i=1}^{m} \beta_i \exp(r_i)$$

   for each $\beta_i \in [0, 1]$ such that $\sum_{i=1}^{m} \beta_i = 1$. By this convexity, one can derive (see Figure 1 on the next page:

   $$\mathbf{E}_t = \sum_{i=1}^{m} \delta\left(y^{(i)} \neq sign\left(C_t\left(\mathbf{x}^{(i)}\right)\right)\right) \leq \sum_{i=1}^{m} \exp\left(-y^{(i)} C_t\left(\mathbf{x}^{(i)}\right)\right)$$

[1] https://en.wikipedia.org/wiki/Exponential_function

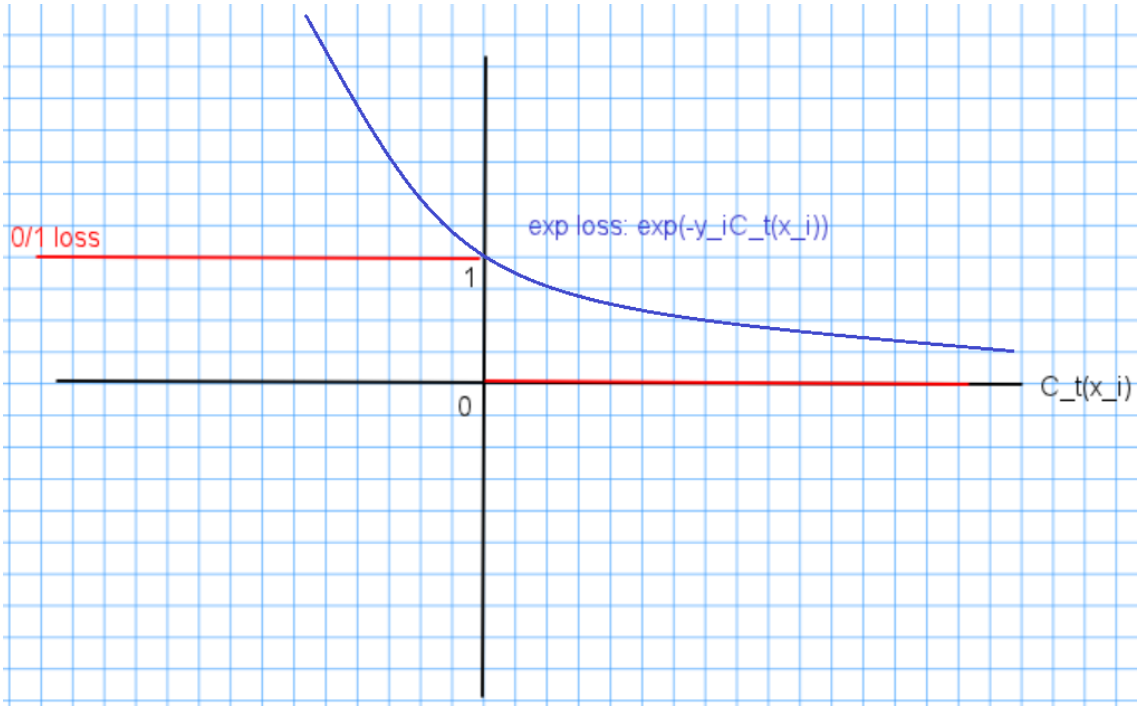


Figure 1: Figure Illustrating how the **exponential loss** is an upper bound for the **0/1 loss**

2. Claim2: $\alpha_t = (1/2)\ln((1 - err_t)/err_t)$ actually minimizes this upper bound.

3. (Advanced) Claim3: If each classifier is slightly better than random, that is if $err_t < 1/K$, Adaboost achieves zero training error exponentially fast

2 Boost Cluster

Consider the points in Figure 2. There are 4 clusters of points, (a) Profs who are Linguists, (b) Profs who are Computer Scientists, (c) Students who are Linguists and (d) Students who are Computer Scientists.

Suppose we wanted to cluster the points using the K means algorithm, with a value of $K = 2$. With this, it is not clear which are the two clusters that we would like the K means algorithm to generate. But suppose we were given constraints between points of the form **Must-link** and **Cannot-link**. For example, we see that the two constraints in Figure 2 assert that

1. one of the points corresponding to (Prof, Linguist) belongs to the same cluster as the point corresponding to (Student, Linguist)

2. the point corresponding to (Prof, Linguist) does not belong to the same cluster as the point corresponding to (Prof, Computer Scientist)

Thus, based on the above constraints, one would expect the **ideal** two clusters to correspond to 'Linguist' and 'Computer Scientist' respectively.

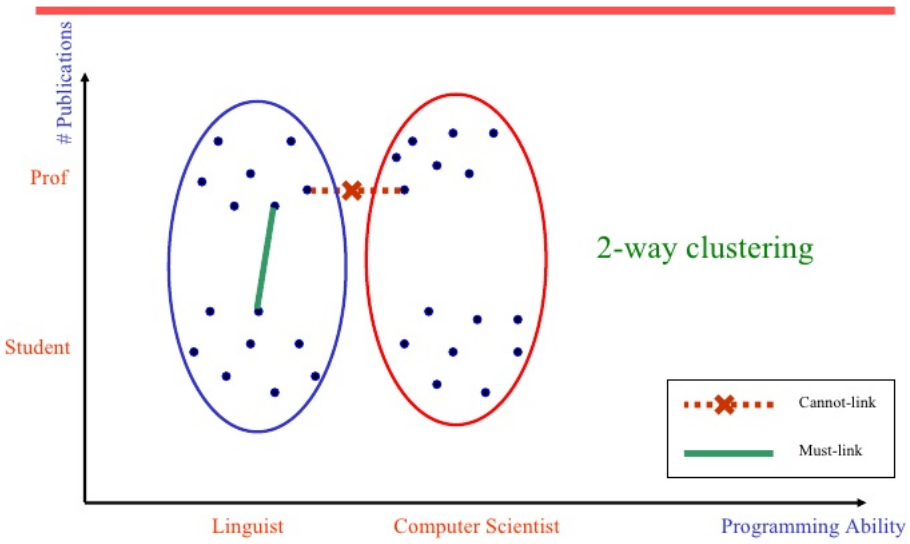


Figure 2: Clustering with Must Link and Cannot Link Constraints

Let $X = (x_1, \ldots, x_n)$ denote the collection of examples to be clustered, where n is the total number of examples and each example $x_i \in \Re^d$ is a vector of d dimensions. We use a matrix $S^+ \in \Re^{n \times n}$ to represent all the must-link pairs, where $S^+_{i,j} = 1$ when examples x_i and x_j form a must-link pair, and $S^+_{i,j} = 0$ otherwise. Similarly, we use a matrix $S^- \in \Re^{n \times n}$ to represent all the cannot-link pairs, where $S^-_{i,j} = 1$ when examples x_i and x_j form a cannot-link pair, and zero otherwise.

1. Propose a heuristic and intuitive modification to the K-means clustering algorithm in order to accommodate such constraints in clustering.

   Hint: You can use the boosting paradigm.

2. We next attempt to solve the problem move formally by proposing an optimization problem (in exponential form, very much like we did for Adaboost). In order to identify which constraint pairs are not well satisfied, we introduce the kernel similarity matrix $K \in \Re^{n \times n}$, where $K_{i,j} \geq 0$ indicates the confidence of assigning examples x_i and x_j to the same cluster. We propose the following objective function to be minimized to find/determine the optimal kernel matrix K through some modified clustering algorithm:

$$L = \sum_{i,j=1}^{n} \sum_{a,b=1}^{n} S^+_{i,j} \times \_\_\_\__1 \times \exp\left(\_\_\_\__2 \times \_\_\_\__3\right)$$

   Fill in the three blanks $\_\_\_\__1$, $\_\_\_\__2$ and $\_\_\_\__3$ as functions of K, S^+ and S^-. Explain the objective thus formed. Note that $\exp(t)$ stands for e^t and $\times$ stands for the multiplication operation.

SOLUTION. Solution to problem 1: The goal of the modified boosting algorithm is to identify the subspace that keeps the data points in the unsatisfied must-link pairs close to

each other, and keeps the data points from the unsatisfied cannot-link pairs well separated. One option is to weight each point in K-means based on its importance and boost the memberships of must-link points to the same clusters when land up belonging to different clusters at the end of any specific iteration. We will also entertain other sound heuristic variants than just the one specified.

SOLUTION. Solution to problem 2: (a) $\_\_\_\__1 = S^-_{a,b}$ (b) $\_\_\_\__2 = K_{a,b}$ and (c) $\_\_\_\__2 = 1 - K_{i,j}$ so that

$$L = \sum_{i,j=1}^{n} \sum_{a,b=1}^{n} S^+_{i,j} S^-_{a,b} \exp(K_{a,b} - K_{i,j})$$

or (c) $= (K_{a,b} - K_{i,j})/K_{a,b}$ etc... The objective measures the inconsistency between the kernel matrix K and the given pairwise constraints. When all the constraints are satisfied, we expect to observe a large value for kernel similarity $K_{i,j}$ if x_i and x_j form a must-link pair, and a small value for $K_{i,j}$ if x_i and x_j form a cannot-link pair. In the above, each term within the summation compares $K_{a,b}$, i.e., the similarity between two points from a cannot-link pair, to $K_{i,j}$, i.e., the similarity between two data points from a must-link pair. By minimizing the objective function we will ensure that all the data points in the must-link pairs are more similar to each other than the data points in the cannot-link pairs.

3 Decision Trees and Feature Selection

- **QUESTION:** We discussed the information gain criterion for feature splitting in Decision trees. Suggest two other criteria. Motivate each.

3.1 ANSWER

Solution[2]**:** The splitting attribute is selected greedily as mentioned in the last class and is based on maximum reduction in impurity. The expression is given by:

$$_{V(\phi_i),\phi_i} \left(Imp(S) - \sum_{v_{ij} \in V(\phi_i)} \frac{|S_{v_{ij}}|}{|S|} Imp(S_{v_{ij}}) \right)$$

where $S_{ij} \subseteq \mathcal{D}$ is a subset of dataset such that each instance x has attribute value $\phi_i(x) = v_{ij}$.

1. An example choice of $Imp(S)$ discussed in the class notes is the entropy[3] $Imp(S) = H(S) = -\sum_{i=1}^{K} Pr(C_i) \bullet \log(Pr(C_i))$.

[2]Section 5.1 of `https://www.cse.iitb.ac.in/~cs725/notes/classNotes/extra_lecturenotes_cs725.pdf`

[3]See slide 5 of `http://23.253.82.180/course/307/858/2217`

2. Alternative impurity measures are shown in Table 1 and they all measure the extent of spread of the probabilities over the classes. In other words, as shown in Figure 3 each impurity measure indicates the extent to which the data is "confused" about the classes.

Name	Imp (D)
Entropy	$-\sum_{i=1}^{K} Pr(C_i) \bullet \log(Pr(C_i))$
Gini Index	$\sum_{i=1}^{K} Pr(C_i)(1 - Pr(C_i))$
Class (Min Prob) Error	$_i(1 - Pr(C_i))$

Table 1: Decision Tree: Impurity measurues

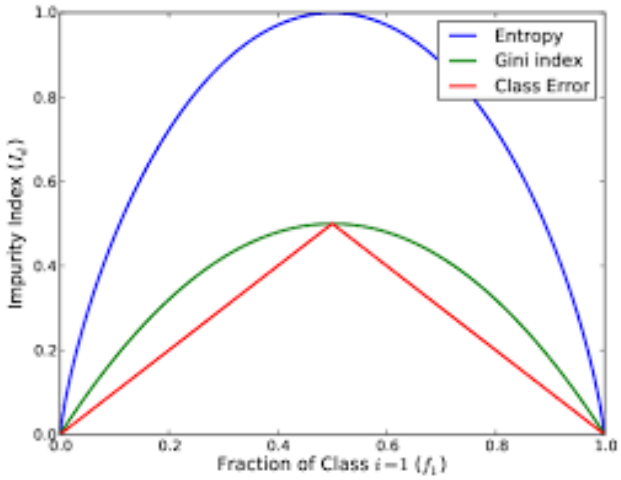


Figure 3: Plot of Entropy, Gini Index and Misclassification Accuracy. Source: `https://inspirehep.net/record/1225852/files/TPZ_Figures_impurity.png`

The second term in the above expression is the expected new impurity. V is a function which returns the split values given an attribute ϕ_i. So $V(\phi_i)$ can be varied for any ϕ_i. It could have many values or a range of values. A example of $V(\phi_i)$ is $V(\phi_i) = \{1, 3, 5\}$ which translates to the following split points $\phi_i < 1,\ 1 \leq \phi_i < 3,\ 3 \leq \phi_i < 5,\ \phi_i \geq 5$

3.1.1 An empirical observation

Since smaller range of attribute values in each split in V_i tends to lead to more skewed class distribution in that split, larger $|V(\phi_i)|$ generally yields larger reduction in impurity. This makes the algorithm to choose more skewed and complex trees and leads to the problem of `overfitting`, if not fixed. Recall that in overfitting, the system

learns a model which is specific to the training collection and does not generalize well on unseen data.

3.1.2 Need to address this empirical observation

This could be achieved by the maximization of the following expression:

$$Imp_{skew}(S) = Imp(S) \; - \; \left(\frac{\sum_{v_{ij} \in V(\phi_i)} \frac{|S_v|}{|S|} Imp(S_{v_{ij}})}{-\sum_{v \in V(\phi_i)} \frac{|S_v|}{|S|} \log(S_v)} \right)$$

The second term in the above expression is called $\triangle Imp(S)$. The intuition for using this term is that, more the skew, lower will be the denominator and is therefore better at countering lowered impurity. In other words, this new measure prefers a less skewed tree as shown in Figure 4. We will refer to the above modified measure $Imp_{skew}(S)$ as the Skew Adjusted Information Gain.

Figure 4: Skewness and empirical observation

3.1.3 Summing it all up

Below we present the overall decision tree learning algorithm with stopping criteria as described above. Ideally, this algorithm goes on until all the data points at the leaf are of the same single class and the two stopping criterion added to the algorithm make it terminate even if such a condition does not occur.

ϕ = empty return a tree with only one branch C_j, where C_j is the majority class in S Stopping criterion all instances in S have label = c_i return a tree with only one branch c_i Stopping criterion $\phi_j =_{V(\phi_i),\phi_i} \left(\triangle Imp(S)\right) \quad \forall v \in V(\phi_i) \; T_v = dtree(S_v, \phi - \phi_i, V)$ return a tree rooted at ϕ_i and having all the T_v branches

- **QUESTION:** We also discussed how the information gain criterion can be used for feature selection in general. Suggest two other criteria. You can build on your answer to the question above.

3.2 ANSWER

The answer is similar to the one above. All the three impurity functions, *viz.*, entropy, gini-index and misclassification error could be used for feature selection. Moreover, the Skew Adjusted Information Gain measure $Imp_{skew}(S)$ could also be used for feature selection.

- **OPTIONAL QUESTION:** Suggest how you could used hypothesis testing for pruning or stopping decision tree construction.

3.3 ANSWER

Simpler trees are preferred over their complex counterparts for the following reasons:

1. They are faster to execute
2. They perform better on unseen data. In otherwords, they "generalize well". For instance, a simpler tree learnt in the class had lower accuracy on the train set but higher accuracy on the unseen test set.

3.3.1 Alternatives in Pruning

There are various strategies/heuristics to decrease the complexity of the tree learnt. Some of the options are as follows:

1. Early termination. Stop if $\triangle Imp(S) < \theta$, where θ is some threshold.
2. Majority class $\geq \alpha\%$, for some value of α
3. Pruning: The idea is to build complex trees and prune them. This is a good option, since the construction procedure can be greedy and does not have to look ahead. Some Hypothesis testing procedures could be used to achieve this. (For instance, the binomial and χ^2-tests).
4. Use an objective function like

$$max_{\phi_i}\Big(\triangle Imp(S,i) - Complexity(tree)\Big)$$

The complexity is characterized by the description length principle (MDL)

Please note that the section that follows is a completely optional reading. However, it could enhance your understanding significantly.

3.3.2 (OPTIONAL) Hypothesis Testing for Decision Tree Pruning

The question to answer while constructing a simpler decision tree is Do we have to split a node (using some attribute) or not ? Consider the situation in Figure 5. The numbers n_1, n_2 etc. indicate the number of instances of the particular class.

If the class ratios of instances remain similar to that before the split, then we might not gain much by splitting that node. To quantify this, we employ Hypothesis testing. The idea is to compare 2 probability distributions.

We will illustrate with a 2-class classification problem. If p is the probability of taking the left branch, the probability of taking the right branch is $1 - p$. Then we obtain the following:

$$\begin{aligned} n_{11} &= pn_1 \\ n_{21} &= pn_2 \\ n_{12} &= (1-p)n_1 \\ n_{22} &= (1-p)n_2 \end{aligned}$$

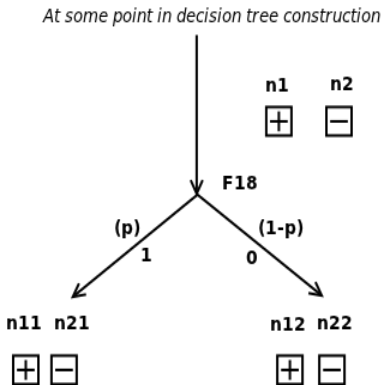


Figure 5: Splitting criterion

Consider the original ratio (also called reference distribution) of positive to total no. of tuples. If the same ratio is obtained after the split, we will **not** be interested in such splits. i.e.,

$$\frac{n_1}{n_1 + n_2} = \frac{n_{11}}{n_{11} + n_{21}}$$

or in general

$$\frac{n_j}{n_1 + n_2} = \frac{n_{j1}}{n_{1i} + n_{2i}} \quad \forall j \text{ no. of classes } \& \ i = 1, 2$$

Why? Because these splits only add to the complexity of the tree and do not convey any meaningful information not already present. Suppose we are interested not only in equal distributions but also in approximately equal distributions i.e.,

$$\frac{n_1}{n_1 + n_2} \approx \frac{n_{11}}{n_{11} + n_{21}}$$

The idea is to compare two probability distributions. It is here that the concept of hypothesis testing is employed.

The ratio $\frac{n_1}{n_1+n_2}$ is called the reference distribution. In general, it is:

$$p(C_i) = \mu_i = \frac{n_i}{\sum_{i=1}^{K} n_i} \quad \text{for a given class } i \text{ (pre-splitting distribution)}$$

3.3.3 Hypothesis testing: problem

Note: The text presented here in red refers specifically to the decision tree problem. The text in black is for the general Hypothesis testing problem and can be read up in Section 7 of `https://www.cse.iitb.ac.in/~cs725/notes/classNotes/extra_lecturenotes_cs725_aut11.pdf`

Let $X_1, \ldots X_n$ be i.i.d random samples that correspond to class labels of instances that have gone into the left branch.

The null hypothesis is H_0 and the alternative hypothesis is H_1:

$$H_0: \quad X_1, \ldots, X_n \in C$$
$$H_1: \quad X_1, \ldots, X_n \notin C$$

The distribution of samples in the left branch is same as before splitting i.e. $\mu_1, \ldots \mu_k$.

Given a random sample, we need to test our hypothesis i.e., given an $\alpha \in [0, 1]$, we want to determine a C such that

$$Pr_{H_0}(\{X_1, \ldots X_n\} \notin C) \leq \alpha \qquad \text{Type I error}$$

Given an $\alpha \in [0, 1]$, probability that we decide that the pre-distribution and the left-branch distribution are different, when in fact they are similar, is less than or eqaul to α.

[Currently we are not very much interested in the Type II error, i.e. $Pr_{H_1}(\{X_1, \ldots X_n\} \in C)$].

Here, C is the set of all possible "interesting" random samples. Also,

$$Pr_{H_0}(\{X_1, \ldots X_n\} \notin C') \leq Pr_{H_0}(\{X_1, \ldots X_n\} \notin C) \qquad \forall C' \supseteq C$$

We are interested in the "smallest" / "tightest" C. This is called the critical region C_α. Consequently,

$$Pr_{H_0}(\{X_1, \ldots X_n\} \notin C_\alpha) = \alpha$$

3.3.4 Goodness-of-fit test for DTrees

Consider the test statistic $S_i = \sum_{j=1}^{n} \delta(X_j, C_i)$.

We are interested in the hypothesis H_0 where new-distribution = old-distribution$(\mu_1, \ldots \mu_j)$.

$$\begin{aligned}
\mathcal{E}_{H_0}[Y_i] &= \sum_{j=1}^{n} \mathcal{E}_{H_0}[\delta(X_j, C_i)] \\
&= \sum_{j=1}^{n} \mu_i * 1 + (1 - \mu_i) * 0 \\
&= n\mu_i
\end{aligned}$$

$$\begin{aligned}
C &= \left\{ (X_1, \ldots X_n) \middle| \sum_{i=1}^{K} \frac{(Y_i - \mathcal{E}_{H_0}(Y_i))^2}{\mathcal{E}_{H_0}(Y_i)} \leq c \right\} \qquad \text{where } c \text{ is some constant} \\
&= \left\{ (X_1, \ldots X_n) \middle| \sum_{i=1}^{K} \frac{(Y_i - n\mu_i)^2}{n\mu_i} \leq c \right\}
\end{aligned}$$

As we might have seen in a basic course on statistics[4], the above expression $\sim \chi^2_{K-1}$. We then use the chi-square tables to find c given the value of α.

[4] Section 7 of `https://www.cse.iitb.ac.in/~cs725/notes/classNotes/extra_lecturenotes_cs725.pdf`

3.3.5 Final Heuristic used for DTree construction

- Compute $\sum_{i=1}^{K} \frac{(Y_i - n\mu_i)^2}{n\mu_i} \sim t_s \quad \forall\ splits$, where t_s is the test statistic.
- Stop building the tree, if for a given α, $\quad t_s \leq c_\alpha \quad \forall\ splits$
- Compute c_α such that $Pr_{\chi^2_{K-1}}(x \geq c_\alpha) = \alpha$