6.891 Machine learning and neural networks

Tommi Jaakkola
MIT AI LAB
tommi@ai.mit.edu

Topics

Classification methods

- Multi-layer neural networks, back-propagation
- A bit of decision theory
- Regularization
- Regularization and prior probabilities
- Example

Additive models and classification

- We have to somehow choose the basis functions $\{\phi_i(\mathbf{x})\}$ for our (additive) logistic regression model

$$P(y=1|\mathbf{x},\mathbf{w}) = g(\, w_0 + w_1\phi_1(\mathbf{x}) + \ldots w_m\phi_m(\mathbf{x})\,) \; ? \tag{1}$$

  - When we select a few basis functions from a larger pool this is a "feature selection" problem
  - When we select families of basis functions (e.g., quadratic/linear) this is a model selection problem

- One approach is to make the basis functions adjustable...

Perceptron

- Simple perceptron is (like) a logistic regression model

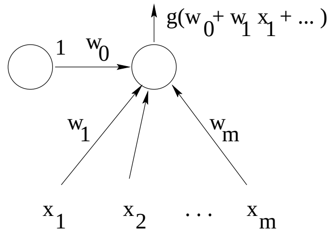
$$\phi(\mathbf{x};\mathbf{w}) = g(\, w_0 + w_1 x_1 + \ldots + w_m x_m\,) \tag{2}$$

where a linear combination of the inputs, i.e.,

$$z = w_0 + w_1 x_1 + \ldots + w_m x_m \tag{3}$$

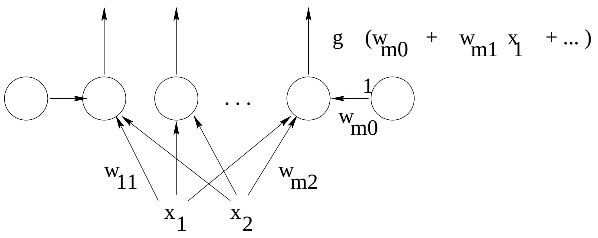
is passed through a logistic (or other) squashing function, $g(z) = (1+\exp(-z))^{-1}$.

- The perceptron output is *interpretable* as a probability



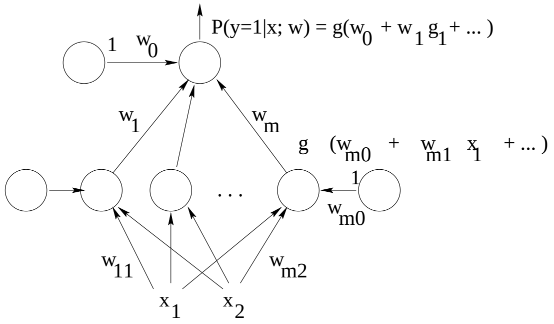
Multi-layer neural networks

- We can certainly apply many perceptrons to the same input vector $\mathbf{x}$
- If the parameters of these perceptrons are different, they serve as different "basis functions" or "feature detectors"



Multi-layer neural networks

- A two layer neural network with a single output unit



- Theoretically only two layers are needed
- When $\mathbf{w} \approx 0$, the mapping from inputs to outputs is linear

Multi-layer neural networks cont'd

- Estimation as before, maximizing the likelihood of the labels given the examples (this assumes that we interpret the output unit as a logistic regression model)
- Given a training example $(\mathbf{x}, y)$, we can perform gradient ascent (or hill-climbing) in the log-probability:

$$\mathbf{w} \leftarrow \mathbf{w} + \epsilon \frac{\partial}{\partial \mathbf{w}} \log P(y|\mathbf{x}, \mathbf{w}) \tag{4}$$

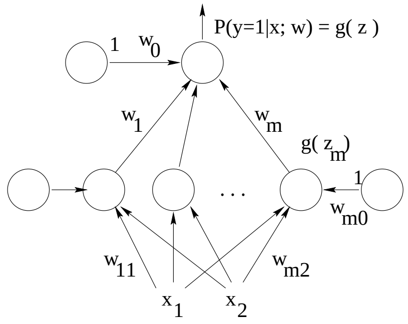
- Online vs. batch updates
- The gradient can be computed via error back-propagation
- There (sometimes many) local optima

Back-propagation

Let z, $z_i, i = 1, \ldots, m$ be the total "input" to each unit (neuron) in the context of $\mathbf{x}$

$$\begin{aligned} z &= w_0 + w_1 g(z_1) + \ldots + w_m g(z_m) \\ z_i &= w_{i0} + w_{i1} x_1 + w_{i2} x_2,\ \ i = 1, \ldots, m \end{aligned}$$

For example, $\log P(y|\mathbf{x}, \mathbf{w}) = y \log g(z) + (1-y) \log(1 - g(z))$



Back-propagation cont'd

- We can propagate the derivatives with respect to the *inputs* z

$$
\begin{aligned}
\delta &= \frac{\partial}{\partial z} \log P(y|\mathbf{x}, \mathbf{w}) \\
\delta_i &= \frac{\partial}{\partial z_i} \log P(y|\mathbf{x}, \mathbf{w}) \\
&= \frac{\partial g(z_i)}{\partial z_i} \times \frac{\partial z}{\partial g(z_i)} \times \frac{\partial}{\partial z} \log P(y|\mathbf{x}, \mathbf{w}) \\
&= g'(z_i) \times w_i \times \delta \qquad (5)
\end{aligned}
$$

- The derivatives with respect to the weights w_{ij} are obtained from δ's

$$
\begin{aligned}
\frac{\partial}{\partial w_{ij}} \log P(y|\mathbf{x}, \mathbf{w}) &= \frac{\partial z_i}{\partial w_{ij}} \times \frac{\partial}{\partial z_i} \log P(y|\mathbf{x}, \mathbf{w}) \\
&= x_j \times \delta_i \qquad (6)
\end{aligned}
$$

Where we are

Classification methods

- Multi-layer neural networks, back-propagation
- *A bit of decision theory*
- Regularization
- Regularization and prior probabilities
- Example

Decision theory

- Suppose we have only two parameter choices (decisions) and n training examples $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$. The empirical losses associated with the decisions are

$$
\begin{aligned}
L_n(\mathbf{w}_1) &= \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(y_i, f(\mathbf{x}_i, \mathbf{w}_1)) \qquad (7) \\
L_n(\mathbf{w}_2) &= \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(y_i, f(\mathbf{x}_i, \mathbf{w}_2)) \qquad (8)
\end{aligned}
$$

  Let's assume that $\mathbf{w}_1$ is the optimal choice, i.e.,

$$
E\{\, L_n(\mathbf{w}_1) \,\} < E\{\, L_n(\mathbf{w}_2) \,\} \qquad (9)
$$

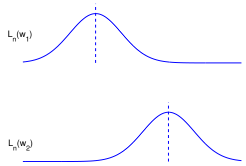
  where the expectation is taken with respect to the random generation of the training set

- Why would we ever make the poorer choice?

Decision theory

- Two parameter choices (decisions), n training examples

$$
\begin{aligned}
L_n(\mathbf{w}_1) &= \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(y_i, f(\mathbf{x}_i, \mathbf{w}_1)) \qquad (10) \\
L_n(\mathbf{w}_2) &= \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(y_i, f(\mathbf{x}_i, \mathbf{w}_2)) \qquad (11)
\end{aligned}
$$



- What is the probability of making the wrong decision?

$$
P(\,\text{error}\,) = P(\, L_n(\mathbf{w}_1) < L_n(\mathbf{w}_2)\,) \qquad (12)
$$

Multiple decisions

- m parameter choices (decisions), n training examples

$$L_n(\mathbf{w}_1) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}(y_i, f(\mathbf{x}_i, \mathbf{w}_1)) \quad (13)$$
$$\ldots$$
$$L_n(\mathbf{w}_m) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}(y_i, f(\mathbf{x}_i, \mathbf{w}_m)) \quad (14)$$

  Suppose again that $\mathbf{w}_1$ is the optimal choice.

- What is the probability of making the wrong decision now?

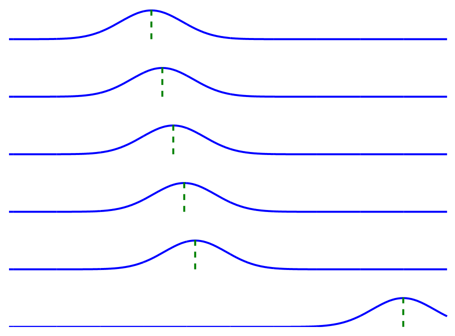
$$P(\text{error}) \approx 1 - \int P(L_n(\mathbf{w}_1) = z) \left[\prod_{i=2}^{m} P(L_n(\mathbf{w}_2) \leq z) \right] dz \quad (15)$$

  (where we have assumed for simplicity that the approximation errors in each $L_n(\mathbf{w}_i)$, $i = 1, \ldots, m$ are independent)

- The probability of error goes up exponentially fast as the number of decisions increase

Multiple decisions

- m decisions, $\mathbf{w}_1$ is optimal



- Many decisions may be nearly equivalent as far as their effects (loss) is concerned

Loss based analysis

- We should really look at how much errors cost us rather than simply whether or not we find *the optimal* decision
- To incur at most ϵ additional loss, it suffices to require that with high probability over the choice of the training set

$$|L_n(\mathbf{w}_i) - E\{\, L_n(\mathbf{w}_i)\,\}| \leq \epsilon, \;\; i = 1, \ldots, m \quad (16)$$

  (this is needlessly strong but suffices for our purposes)

- When this condition is satisfied, we are guaranteed to generalize well

Loss based analysis cont'd

- This measure can be extended to continuous set of "choices" such as estimating the parameters $\mathbf{w}$ in a logistic regression model
- To incur at most ϵ additional loss, we require that with high probability

$$|L_n(\mathbf{w}) - E\{\, L_n(\mathbf{w})\,\}| \leq \epsilon, \;\; \text{for all valid } \mathbf{w} \quad (17)$$

- Vapnik: "falsifiability"

Regularization

- The purpose of regularization is to improve generalization
- Regularization should limit our parameter choices such that the empirical losses are close to the expected losses.
- There are various forms of regularization
  1. Keep parameter values small (avoid overly strong predictions)
  2. Complexity penalties (e.g., for linear/quadratic)
  3. Feature selection

  etc.

Regularization: example

- Logistic regression model again

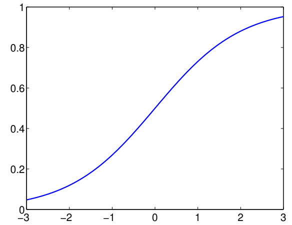
$$P(y = 1|\mathbf{x}, \mathbf{w}) = g(\, w_0 + \sum_{i=1}^{d} w_i x_i \,) \tag{18}$$

- Maximum penalized likelihood (i.e., with regularization):

$$J_n(\mathbf{w}; C) = \sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i, \mathbf{w}) - C\|\mathbf{w}\|^2/2 \tag{19}$$

where larger values of C impose stronger regularization.

- How are we limiting our choices here?



Regularization and prior probability

- Let's assign a simple Gaussian prior probability over the parameters $\mathbf{w}$ in the logistic regression model

  $P(\mathbf{w}) = N(\mathbf{w}; \mu, \sigma^2 I)$ (shorthand for multivariate normal)

  and maximize the log-probability of the observed data **and** the parameters

$$J_n(\mathbf{w}; C) = \sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i, \mathbf{w}) + \log P(\mathbf{w}) \tag{20}$$

$$= \sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i, \mathbf{w}) - \frac{1}{\sigma^2}\|\mathbf{w}\|^2/2 + \text{const.} \tag{21}$$

This is the same as before so long as we define $C = 1/\sigma^2$

Example problem

- Text classification (information retrieval)

  A large number of documents in a database

  Only a few labeled documents $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
- We wish to build a classifier on the basis of the training set (labeled documents). For simplicity we assume that the labels are binary (1/0)
- We need several steps
  1. Feature transformation (why?)
  2. Model specification
  3. Model estimation (and feature/model selection)

Feature transformation

- The presence/absence of specific words in a document carries information about what the document is about
- We can construct m (about 10,000) indicator features $\{\phi_k(\mathbf{x})\}$ for whether a word appears in the document

  $\phi_k(\mathbf{x}) = 1$, if word k appears in the document; otherwise zero

  $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}) \ldots \phi_m(\mathbf{x})]^T$ is the resulting feature vector

- Could we extract better features?

Model specification

- We can treat each word detector $\phi_i(\mathbf{x})$ as an independent expert
- We combine these "expert opinions" by modeling their decisions given the labels:

$$P(\phi(\mathbf{x})|y,\theta) = \left[\prod_{k=1}^{m} P(\phi_k(\mathbf{x})|y,\theta_k)\right] \tag{22}$$

  where $P(\phi_k(\mathbf{x})|y,\theta_k)$ is the conditional probability that the k^{th} word appears in a document labeled y. θ_k are the parameters associated with this conditional probability.

- Classification via Bayes rule:

$$P(y|\phi(\mathbf{x})) = \frac{P(\phi(\mathbf{x})|y)P(y)}{P(\phi(\mathbf{x}))} = \frac{P(\phi(\mathbf{x})|y)P(y)}{\sum_{y'=0}^{1} P(\phi(\mathbf{x})|y')P(y')} \tag{23}$$

Model estimation

- We can write the conditional probabilities of a single feature as

$$P(\phi_k(\mathbf{x})|y,\theta_k) = \theta_{k|y}^{\phi_k(\mathbf{x})}\,(1-\theta_{k|y})^{1-\phi_k(\mathbf{x})} \tag{24}$$

  where $\theta_{k|y}$ is the probability that the word k appears in a document labeled y and $\theta_k = \{\theta_{k|1}, \theta_{k|0}\}$.

- Maximum likelihood estimation (here for a single feature)

$$\begin{aligned}
&\sum_{i=1}^{n} \log P(\phi_k(\mathbf{x}_i)|y_i,\theta_k) \\
&\quad = \sum_{i=1}^{n} \left[\phi_k(\mathbf{x}_i)\log(\theta_{k|y_i}) + (1-\phi_k(\mathbf{x}_i))\log(1-\theta_{k|y_i})\right] \\
&\quad = \sum_{y=0}^{1} \left[N_{ky}\log(\theta_{k|y}) + (N_y - N_{ky})\log(1-\theta_{k|y})\right]
\end{aligned} \tag{25}$$

  $N_{ky} = \#$ of times word k appears in documents labeled y and $N_y = \#$ of times label y appears

Model estimation cont'd

- We get closed form maximum likelihood estimates

$$\begin{aligned}
J_n(\theta_k) &= \sum_{y=0}^{1} \left[N_{ky}\log(\theta_{k|y}) + (N_y - N_{ky})\log(1-\theta_{k|y})\right] \\
\frac{\partial}{\partial\theta_{k|1}} J_n(\theta_k) &= \frac{N_{k1}}{\theta_{k|1}} - \frac{N_1 - N_{k1}}{1-\theta_{k|1}} = 0
\end{aligned} \tag{26}$$

  implying that $\hat{\theta}_{k|1} = N_{k1}/N_1$ (interpretation?).

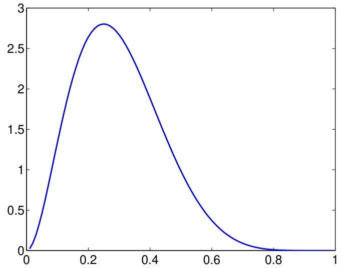
- **BUT**: we have very few documents and some words are rare; these estimates are unlikely to be good
- We need regularization but what prior should we use?

Model estimation: prior

- *Beta distribution* as a prior:

$$P(\theta_{k1}) = c_{k1} \cdot \theta_{k1}^{N'_{k1}} (1 - \theta_{k1})^{N'_1 - N'_{k1}} \tag{27}$$

  where c_{k1} is the normalization constant. The parameters N'_{k1}, N'_1 of the prior are known as *hyper-parameters*.



$N'_1 = 6$, $N'_{k1} = 2$

- Equivalent sample size interpretation!

Model estimation cont'd

- Combining the log-likelihood with the log-prior probability we get

$$\hat{\theta}_{k|1} = \frac{N_{k1} + N'_{k1}}{N_1 + N'_1} \tag{28}$$

  (you should be able to derive these as the maximum penalized likelihood estimates)

  Similarly we get all the remaining parameters $\{\hat{\theta}_{k|1}, \hat{\theta}_{k|0}\}$, $k = 1, \ldots, m$