

discussion04_connecting_the_dots

October 6, 2016

1 Discussion 04: The Big Picture

EECS445 Machine Learning

chansool@umich.edu

A note on formatting: in the main text (excluding titles and optional notes) - Double quotes " " are used when I intentionally use a term that is not mathematically well-defined. - Italicized words are somewhat important jargons that may be repeatedly used. - Boldfaced letters are used for very important terminology that you must absolutely know if you want to claim you've taken ML.

1.1 Supervised learning: the first steps

What is our task?

- We want to make **predictions** about unseen data (called **test data**).
 - Example 1: Instagram wants to predict if a new image is offensive or not. (**Binary classification** problem)
 - Example 2: A pharmaceutical company wants to predict a drug's shelf life. (**Regression** problem)
 - Example 3: Gmail wants to predict if a new email goes to in-box/forum/promotions/updates. (**Multiclass classification** problem)

What are we given?

- We have access to **labeled training data**, which is a collection of pairs of *input* and **target value**. We use the following notation: $\mathcal{D}_{\text{train}} = \{(x_1, t_1) \dots, (x_n, t_n)\}$.
 - Example 1: Instagram had people on Mechanical Turk manually identify offensive images.
 - Example 2: The company has information on existing drugs.
 - Example 3: Gmail users label their emails.

Where do we start?

- We choose the representation of data, called **features**.
 - Example 1: We *vectorize* the RGB values of an image: 400x300 image will be represented as 360K-dimensional vector.
 - Example 2: For each commonly used active ingredient, we have true/false value.
 - Example 3: We count the word frequencies and construct a frequency vector.

Wait, what the hell is the ϕ or Φ nonsense?

- We often use $\phi(x)$ to denote the features of a single input, and Φ to denote the **design matrix**. This is an old convention from the time when kernel methods were popular. It is indeed very safe to ignore the ϕ altogether and think in terms of \mathbf{x} and X instead.

1.2 Model

- First of all, we choose a **model**, which reflects our assumptions about data.
 - Of course, we often have to build a new model if none of the existing ones suits our need.
- We denote the model **parameters** by θ_{model} . (This is a “placeholder” notation.)
- We say a model is **probabilistic**, if it treats data as random variables. Practically speaking it defines probabilities over possible values of data. **Optional Tip: In terms of OOP, the model corresponds to a class* whose member variables need to be initialized with θ_{model} when you want to instantiate it. ***

What models have we learned so far?

- **Linear regression** has parameter \mathbf{w} , called the weight vector.
 - The target value t can take any real number.
 - Our model is probabilistic: a data (\mathbf{x}, t) has probability $f_{\mathcal{N}(0,1)}(t - \mathbf{w}^\top \mathbf{x}) = f_{\mathcal{N}(\mathbf{w}^\top \mathbf{x}, 1)}(t)$ where $f_{\mathcal{N}(\mu, \gamma)}$ is the pdf of Gaussian distribution with mean μ and variance γ .
 - The reflected assumption is that the target value t is *concentrated* (centered closely around) a linear combination of features.
- **Logistic regression** has parameter \mathbf{w} , called the weight vector.
 - The target value t is a *binary label* (1=True or 0=False).
 - Our model is again probabilistic: a data (\mathbf{x}, t) has probability $\sigma(\mathbf{w}^\top \mathbf{x})$ if $t = 1$, and probability $1 - \sigma(\mathbf{w}^\top \mathbf{x})$ if $t = 0$.
 - The reflected assumption is that the probability $t = 1$ scales nonlinearly in a linear combination of features.
- **Softmax regression** has parameter $\mathbf{w}_1, \dots, \mathbf{w}_C$, where C is the number of class labels.
 - The target value t is a class label $(1, \dots, C)$.
 - A data $(\mathbf{x}, t = k)$ has probability *proportional to* $\sigma(\mathbf{w}_k^\top \mathbf{x})$.
- **Naive Bayes** has parameters π_1, \dots, π_C (prior) and $\{\theta_{cdm} : c = 1, \dots, C, d = 1, \dots, D, m = 1, \dots, M\}$
 - The target value t is a class label $(1, \dots, C)$.
 - Our probabilistic model builds upon the **conditional independence** assumption.

Hmm, tell me one model that's not probabilistic!!

- **SVM** has parameters \mathbf{w} and b .
 - The target value t is a binary class label, this time -1 or $+1$ (instead of 0 or 1). Trust me, this will save us from future disaster.

- Our model is not probabilistic. We have a deterministic process of computing $t = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$.
- Assumption: We have a **hyperplane** separating two classes.

Hyperplane is simply a $(N - 1)$ -dimensional thingy in N -dimensional space: for example, a point in number line, a line in x-y coordinate.) *Advanced reading: Well, I actually lied. SVM can* be viewed as a probabilistic model. If you are interested, read http://www.icml-2011.org/papers/386_icmlpaper.pdf **

1.3 Prediction

The choice of model dictates how we make predictions. Notational note: now we are given a new input \mathbf{x} and we want to predict the unknown value t .

- **Linear regression** has parameter \mathbf{w} , called the weight vector.
 - Our model is probabilistic: a data (\mathbf{x}, t) has probability $f_{\mathcal{N}(0,1)}(t - \mathbf{w}^\top \mathbf{x}) = f_{\mathcal{N}(\mathbf{w}^\top \mathbf{x}, 1)}(t)$ where $f_{\mathcal{N}(\mu, \gamma)}$ is the pdf of Gaussian distribution with mean μ and variance γ .
 - We might as well predict the peak of this bell curve, $t_{\text{pred}} = \mathbf{w}^\top \mathbf{x}$.
- **Logistic regression** has parameter \mathbf{w} , called the weight vector.
 - Our model is again probabilistic: a data (\mathbf{x}, t) has probability $\sigma(\mathbf{w}^\top \mathbf{x})$ if $t = 1$, and probability $1 - \sigma(\mathbf{w}^\top \mathbf{x})$ if $t = 0$.
 - We might as well predict the t_{pred} that gives the higher probability.
- **Softmax regression** has parameter $\mathbf{w}_1, \dots, \mathbf{w}_C$, where C is the number of class labels.
 - A data $(\mathbf{x}, t = k)$ has probability *proportional to* $\sigma(\mathbf{w}_k^\top \mathbf{x})$.
 - We might as well predict the t_{pred} that gives the highest probability.
- **Naive Bayes**.. you get the idea.
- **SVM** is a bit different story here; it is defined to make a deterministic prediction $t = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$.

1.4 (Parameter) Estimation aka Training

Estimation is the process of using the training data to find the “best” model parameters that we think will perform best on the (unseen) test data. Estimation can be broken into two steps: we first define the “best”, and then actually find it.

1.4.1 Objective function

How do you define the “best”?

- We define an **objective function** (also called **error function** or **penalty function** for the model, often denoted E and sometimes J or F or pretty much any letter. The “best” parameters are the ones that minimize this function E .

- *Note:* Because we define E after we choose our model, we don't have to write $E_{\text{linear regression}}$ or $E_{\text{logistic regression}}$. It should be clear from the context.

Optional tip: The function E can be considered as taking input of type $[\text{model}].\text{parameters}$, where $[\text{model}]$ can be linear regression, logistic regression, etc. ***

Example objective function: Negative log likelihood (LL)

- The likelihood L is a function of model parameters that returns a non-negative number: $L(\theta_{\text{model}}) = P(\mathcal{D}_{\text{train}}; \theta_{\text{model}})$.
- Minimizing $-L(\theta_{\text{model}})$ is equivalent to minimizing $-\log L(\theta_{\text{model}})$, because log is an increasing function well-defined over positive numbers, and $L(\theta_{\text{model}})$ is always non-negative.
- Although we often glance over this, $L(\theta_{\text{model}}) = 0$ case needs to be handled separately and carefully.

Notational Notes - Similarly to the E notation, because log likelihood only makes sense after we choose our model, we omit the subscripts such as $L_{\text{linear regression}}$. - Semi-colons are used to emphasize the distinction between model parameters and random variables. Recall that a probabilistic model defines the probability of observing a particular data. It is more natural to say that this probability is *determined by* the parameters as opposed to *conditioned on* the parameter values. For mathematical operations, $P(A|B; C) = P(A|B, C)$.

Why is negative LL a good objective function?

- It is a very good objective function if we make the following assumption: train and test data are all **iid (independently and identically distributed) samples** from the same distribution \mathcal{F} , and we have a “large enough” training set.
- Mathematical reasoning is as follows:
 - Suppose we have a model with parameter θ .
 - We get a new unseen data $(\mathbf{x}_{\text{test}}, t_{\text{test}}) \sim \mathcal{F}$, where t_{test} is hidden and to be predicted by our model.
 - Because we have a “large enough” training set, by our iid assumption, sampling $(\mathbf{x}_{\text{test}}, t_{\text{test}})$ from \mathcal{F} is not too different from picking a random training example (\mathbf{x}_i, t_i) from $\mathcal{D}_{\text{train}}$.
 - So, $P(\mathbf{x}_{\text{test}}, t_{\text{test}}; \theta)$, the probability of making a correct prediction on unseen data, is higher if $P(\mathcal{D}_{\text{train}}; \theta)$ is high.
- It is also convenient that $\log P(\mathcal{D}_{\text{train}}; \theta_{\text{model}}) = \log \prod_i P(\mathbf{x}_i, t_i; \theta_{\text{model}}) = \sum \log P(\mathbf{x}_i, t_i; \theta_{\text{model}})$.

Maximum Likelihood Estimate

- **Maximum likelihood estimate** of θ_{model} is the maximizer of the likelihood function, which is equivalent to the minimizer of the negative LL (up to the aforementioned subtleties).

What is that Maximum a p-unspellable then?

- The “large-enough” and iid assumptions don’t generally hold true, and we as humans have a good idea about what model parameters would **generalize better** (i.e., work better on unseen data). We build this preference into the model by defining a **prior** $P(\theta_{\text{model}})$. This introduces a bit of black art into our mathematically beautiful but practically useless models.
 - *Example:* Regularized linear regression, where we define our prior $P(\mathbf{w}) = f_{\mathcal{N}(\mathbf{0}, \alpha^{-1}I)}(\mathbf{w})$ which means we prefer \mathbf{w} with small 2-norm.
- **Maximum a posteriori** (MAP) estimate is the maximizer of a **posterior probability** of parameter given data. Mathematically,

$$\theta_{\text{MAP}} = \arg \max_{\theta_{\text{model}}} P(\theta_{\text{model}}; \mathcal{D}_{\text{train}})$$

or equivalently, in the objective function framework:

$$E(\theta_{\text{model}}) = -P(\theta_{\text{model}}; \mathcal{D}_{\text{train}}).$$

- MAP estimate definition looks quite different from MLE. After the application of Bayes rule, however, it is really just one extra term compared to MLE:

$$\arg \max P(\theta_{\text{model}}; \mathcal{D}_{\text{train}}) = \arg \max P(\mathcal{D}_{\text{train}}; \theta_{\text{model}})P(\theta_{\text{model}})$$

or equivalently, in the objective function framework:

$$E(\theta_{\text{model}}) = -P(\mathcal{D}_{\text{train}}; \theta_{\text{model}})P(\theta_{\text{model}}).$$

SVM uses a very different objective function Again, SVM is unique among the models we learned in that it is not a probabilistic model.

1.4.2 Optimization

How do we find the “best” parameters? *Warning:* This is where math gets very, very dense. Assuming that E is convex,

- Find θ such that $\nabla E(\theta) = 0$.
 - *Example:* Pseudoinverse method for *solving* linear regression. MLE for Bernoulli random variables (Heads or tails coin flip example). MLE for Naive Bayes.
- Gradient descent (Batch or stochastic)
 - Recall the definition of gradient, and understand why it makes sense to continually move in the opposite direction of the gradient.
 - *Advanced reading:* SGD in fact works very well in practice even for non-convex functions (<https://arxiv.org/abs/1509.01240>)
 - *Example:* GD for linear and logistic regression, GD for SVM.
- Newton’s method
 - Requires Hessian, the second derivative.
 - Rarely used in practice. Why?

1.5 Misc

I am still not convinced all this is necessary. It isn't if you want to be a *consumer* of machine learning. In fact, you do not need any course in machine learning at all for that. You all are smart enough to teach yourself how to use excellent ML libraries out there as blackbox. You don't need a semester-long course for that. This course teaches you to be a machine learning *scientist*. That means, at the very least, we are training you to - have a deep understanding of the assumptions and limits of most commonly used models, - make a better judgement on which model to use, and - be able to tweak the existing models or develop a new one to suit your particular needs.