CS 182/282A: Designing, Visualizing and Understanding Deep Neural Networks **Spring 2019**

Section 2: Stats Review, SVMs, Regularization and Validation

Notes by: Forrest Huang

2.1 Course Logistics

2.1.1 Discussion Schedules

Based on last week's survey, we plan to reschedule discussion times to these options:

- Monday 10 AM - 11 AM
- Monday 11 AM - 12 PM
- Monday 12 PM - 1 PM
- Monday 1 PM - 2 PM
- Monday 3 PM - 4 PM
- Tuesday 10 AM - 11 AM
- Tuesday 3 PM - 4 PM
- Tuesday 1 PM - 2 PM

If anyone wishes to attend discussion sessions but cannot make it in any of the sessions, please let us know asap in class/on piazza/through email. Also, if you have requests on topics we should include in any future discussions, please let us know.

2.1.2 Groups

Group projects shall be carried out in groups of 3-4 only. All group member needs to be registered in the same course code (i.e., All CS182 or CS282A) since CS282A has additional requirements for the project.

2.2 Statistics Review

2.2.1 Joint Distributions

Given two independent random variables A and B, and their probabilities $P(A)$ and $P(B)$, their joint probability

$$P(A, B) = P(A) \times P(B)$$

.

To test for independence, A and B are independent iff $P(A) = P(A|B)$.

Relaxing the independence assumption:

$$P(A, B) = P(A|B) \times P(B) = P(B|A) \times P(A)$$

. This is can then be used to derive the Bayes' theorem in the next section.

2.2.2 Prior, Posterior and Likelihood

In this section, we will review prior, posterior and likelihood through an example: estimating the probability of you having flu given you are having a headache (which is normally the goal in a lot of inference problems), using the following notations:

- $P(H)$ - Probability of you having headache
- $P(F)$ - Probability of you having flu.
- $P(F|H)$ - Probability of you having flu given that you have headache.
- $P(H|F)$ - Probability of you having headache given you have a flu

With the Bayes Theorem:

$$P(F|H) = \frac{P(H|F)P(F)}{P(H)}$$

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior}$$

F is the parameter since it characterizes the distribution of flu contractors, which we care about.

Prior $P(F)$ is the probability of your parameters in your distribution of interest.

Likelihood $P(H|F)$ is the probability of your observation given the parameters.

Posterior $P(F|H)$ is the probability of your parameters given the observations.

Evidance $P(H)$ is the probability of your variables in your observations.

Now we will work through a simple dataset to estimate $P(H|F)$ and $P(F)$ by the distributions of the dataset. (note to TA: work through all probabilities to complete a Naive Bayes classification)

Headache	Flu
N	N
Y	N
N	N
Y	Y
Y	Y
N	Y

$$P(F) = \frac{3}{6} = \frac{1}{2}$$

$$P(H|F) = \frac{2}{3}$$

$$P(H) = \frac{3}{6} = \frac{1}{2}$$

$$P(F|H) = \frac{\frac{2}{3} \times \frac{1}{2}}{\frac{1}{2}} = \frac{2}{3}$$

Note that sometimes when optimizing the parameters using EM algorithm (e.g., when some labels are missing), or when comparing various values of the parameters to output a decision, $P(H)$ is dropped from the computation since it does not change across various choices of parameter values.

2.2.3 Probability Density Function

Probability density function $f(x)$ is the relative likelihood where a random variable takes on certain values within a range, such that:

$$P(a < x < b) = \int_a^b f(x)dx$$

See figure 2.3 for an example.

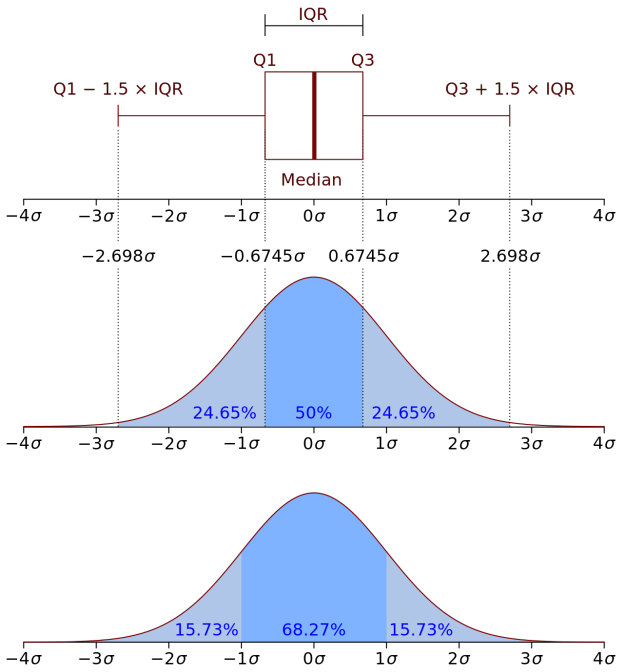

Figure 2.1: A diagram of a pdf for normal distribution. Figure from Wikipedia.

Note that Probability Mass function has similar definition except for discrete random variables.

2.3 SVMs and Regularization

In class, we covered SVMs and hinge loss. The relationship between the two is that SVMs are a machine learning classifier that *uses* the hinge loss for optimization to find good weights w. Recall, to optimize an SVM, we minimize the hinge loss across our observations:

$$l = \sum_{x \in X} \max(0, 1 - y(w^T x))$$

with $y \in \{-1, 1\}$. This is a linear classifier. We mentioned typically $||w||$ should be constraint to a fixed constant for a *hard* SVM, or minimized *softly* described in the formula:

$$l = \sum_{x \in X} \max(0, 1 - y(w^T x)) + \lambda ||w||$$

but why exactly is the case? Why would the hinge loss decrease as $||w||$ increase? Moreover, adding $||w||_2$ to the minimization objective also appeared similar to the regularization applied in the linear regression case covered in class, are they related at all?

To see why hinge loss will decrease as $||w||$ increase, consider an SVM w^* where the separating hyperplane is defined by ${w^*}^T x = 0$. The hinge loss is

$$l^* = \sum_{x \in X} \max(0, 1 - y({w^*}^T x))$$

Now we multiply w^* by 2 to get $w' = 2w^*$. We still get the same hyperplane since we can multiply both sides of the original equation by 2:

$${w'}^T x = 2{w^*}^T x = 0 \times 2$$

But now the hinge loss is halved for points the lie within the margin:

$$l' = \sum_{x \in X, 0 < y(w^T x) < 1} 1 - y(2{w^*}^T x) = 2l^* - 1 < l^*$$

(since l^* is between 0 and 1)

In fact, $\lambda ||w||$ is often considered the *regularization* term in SVM, and λ is named the *regularization constant* in SVMs. If λ is small, this encourages the optimizer to find hyperplanes to have smaller margins for each point, and force the model to correctly classify as many points as possible if some points are non-linearly separable, which can cause over-fitting of data points. If λ is large, the optimizer tends to find separating hyperplanes with larger overall margins, while minimizing the effect of individual mis-classifications, and hence improving generalization of the SVM model.

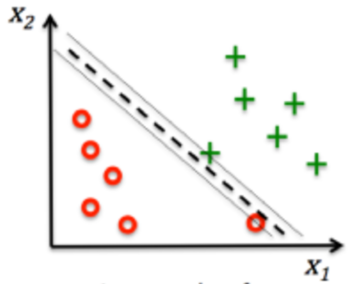

Figure 2.2: A diagram of an SVM with a small λ.

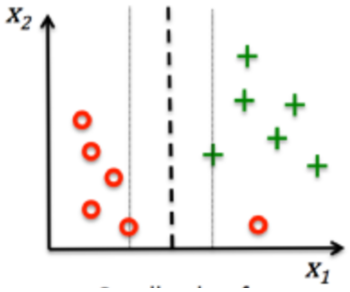

Figure 2.3: A diagram of an SVM with a large λ.

If you are interested in a more in-depth discussion of loss functions, we recommend reading these notes: `http://cs231n.github.io/linear-classify/`. These notes are from a Stanford course for which we borrow some of the material in CS 182/282A.

2.4 Side-note on Dataset splits during training

In the case when hyper-parameter tuning is possible (e.g., λ in SVMs, learning rate of deep nets), in addition to training and test sets, you should hold out validation set for tuning hyper-parameters in your model. The following policies should be taken when using training/validation/test sets:

- Only train your model on the training set, but not the validation set and test set.
- You should never tune your hyper-parameters on your test set, and choose the best model based on the performance on the test set, because in doing so, you are 'over-fitting' your model manually to the test set. You should only tune your hyper-parameters according to the validation set performance.
- The test set should only be run once after you have finalized your model, regardless of whether you use cross-validation or a single training-validation split. You should hold out your test set until you have finalized your model.
- You should use a new test set when you train a new model.