Theoretical and Practical Matters

Grzegorz Chrupała and Nicolas Stroppa

Saarland University
Google

META Workshop

Outline

ML Connections

Some connections with Machine Learning.

- Artificial Intelligence: humans can learn, so should machines
  - focus is on (understanding and) reproducing human learning abilities
- Statistics: if you have facts, we can surely do some inference
  - focus is on data distribution models and their learnability (model assumptions, convergence speeds, ...)
- Optimization: if you define a goal, we can try to get you closer to it
  - focus is on building devices for efficiently (often approximately) finding/searching acceptable solutions
- Applications: speech, finance, nlp, vision, bioinformatics, recognition
  - focus is on defining and formalizing the problems, and usefulness in applications

We can look at similar problems and methods from different angles.

ML Connections

Some connections with Machine Learning.

- Statistics and Optimization:
  - Maximizing likelyhood vs. minimizing loss
- Applications
  - How to measure/evaluate usefulness?
  - How to choose methods?

Outline

Criteria used to learn

You've seen a number of formulas used by Grzegorz to derive the models of Naive Bayes, Perceptron, Maxent.

We're going to revisit those in a more general/unified setting.

Loss minimization

Let's consider the following general optimization setting.
We assume:

- The (decision) function w we're looking for belongs to some predefined function space S
- We have training examples (x_i, y_i)
- We can define a *loss per example* $Loss(\mathbf{w}, x_i, y_i)$ and a *function penalty* $J(\mathbf{w})$

Then, w can be obtained by solving the following *optimization problem*:

$$min_{\mathbf{w} \in S} Loss(\mathbf{w}, D) = \sum_i Loss(\mathbf{w}, x_i, y_i) + \lambda J(\mathbf{w}),$$

where λ is a *regularization* parameter.

Optimization and regularization

In this setting, we simply need to define:

- a loss per example $Loss(\mathbf{w}, x_i, y_i)$:
  - if using the function $\mathbf{w}$ to classify x_i, how far from the truth (i.e. y_i) am I, and how much should I pay for that?
- a function penalty $J(\mathbf{w})$:
  - how complex is $\mathbf{w}$, and much should I pay for that complexity?

- The first part is going to try fitting the data (by minimizing the errors made on the training set)
- The second part is making sure the model we're using is not too complex (and will avoid overfitting the data)
- The tradeoff is controlled by the regularization parameter λ.

Types of losses

Different types of example losses can be considered.

Loss name	$Loss(\mathbf{w}, x, y)$	Used in
0-1 loss	1 if $\mathbf{w}.x \neq y$, 0 otherwise	Perceptron
Hinge loss	$[1 - y\mathbf{w}.x]_+$	SVM
Log loss	$\log(1 + e^{-y\mathbf{w}.x})$	Maxent

This means that, despite their diverse origins, all these approaches can thus be turned into (surprisingly similar) optimization problems!

Types of penalties

For linear models, common penalties are:

- L_2 norm: $\|w\|_2 = \sum_j w_j^2$ (Ridge penalty)
- L_1 norm: $\|w\|_1 = \sum_j |w_j|$ (Lasso penalty)

More generally, for a function f, we can consider regularizations like:

- $J(f) = \int f''(x)^2 dx$, which defines how "bumpy" the function is.

Bayesian approaches

In Bayesian tradition, we can write:

$$p(Model|Data) = \frac{p(Data|Model) \times p(Model)}{p(Data)}$$

For a fixed data set, this yields:

$$p(Model|Data) \propto p(Data|Model) \times p(Model), \text{i.e.}$$

$$p(w|D) \propto p(D|w) \times p(w),$$

or, equivalently:

$$\log p(w|D) = \log p(D|w) + \log Cp(w)$$

and with independent training examples:

$$\log p(w|D) = \sum_i \log p(x_i|w) + \log Cp(w)$$

Loss minimization and Bayesian approaches

Compare

$$-Loss(\mathbf{w}, D) = \sum_i -Loss(\mathbf{w}, x_i, y_i) - \lambda J(\mathbf{w}),$$

with:

$$\log p(w|D) = \sum_i \log p(x_i|w) + \log Cp(w)$$

- A choice of example loss corresponds to a choice of underlying distribution for the example!
- A choice of function penalty corresponds to a choice of a model *prior*!
- The same thing is just seen from different angles...

Penalties and Priors

Ridge penalty.

$$-\lambda\|w\|_2 = \log Cp(w)$$

$$p(w) = \frac{1}{C} exp(-\lambda\|w\|_2)$$

We can easily recognize a Gaussian prior.

Lasso penalty.

$$-\lambda\|w\|_1 = \log Cp(w)$$

$$p(w) = \frac{1}{C} exp(-\lambda\|w\|_1)$$

We can easily recognize a Laplacian prior.

Common regularization penalties correspond to common model priors. . .

Optimization and regularization

There exists further justification for miniming the empirical loss.

Basically, we can consider the training data as samples from a distribution. In this context, the empirical mean loss is just an estimate (in the statistical sense) for the (true/underlying) expected mean loss.

The regularization controls the complexity of the function and prevents the approximation to cause too much overfitting.

Optimization and regularization

How is optimization performed?

Depending on the actual function (differentiable, non-differentiable), different optimizations methods can be used.

Quasi-Newton methods, Quadratic programming, etc.

These are not ML-specific, they are general methods.

Outline

Evaluation considerations

- Don't start playing if you don't know what you're expecting...
- Make sure you can evaluate, even before playing with data and building any algorithm
- You need to convince yourself that you have access to the methodology to prove something
- Again, *you* should know the data...
- Evaluation is not only about metrics, it's about a proper methodology
  - A wise guy once told me: "each time you're reusing the same test set, substract 1 to you results"...
  - Correct methodology somewhere between inner convinction and good (known) metrics on (known) datasets (actually, both are needed)

Evaluation considerations

- Don't start playing if you don't know what you're expecting...
- Make sure you can evaluate, even before playing with data and building any algorithm
- You need to convince yourself that you have access to the methodology to prove something
- Again, *you* should know the data...
- Evaluation is not only about metrics, it's about a proper methodology
  - A wise guy once told me: "each time you're reusing the same test set, substract 1 to you results"...
  - Correct methodology somewhere between inner convinction and good (known) metrics on (known) datasets (actually, both are needed)
  - Evaluate often, evaluate early!

Pipeline issues

- Most of our NLP "systems" are components that integrate in larger pipelines
- What matters at the end of the day is the final results, the *extrinsinc evaluation*
- However, evaluation of a standalone component (*intrinsic evaluation*) is useful to:
  - abstract away from the other components
  - perform fast intermediate experiments

But very easy to forget the bigger picture...

Pipeline issues

- Most of our NLP "systems" are components that integrate in larger pipelines
- What matters at the end of the day is the final results, the *extrinsinc evaluation*
- However, evaluation of a standalone component (*intrinsic evaluation*) is useful to:
  - abstract away from the other components
  - perform fast intermediate experiments

But very easy to forget the bigger picture... Evaluate often, evaluate

early...

Pipeline issues

(Not so) virtual conversation:

- *Bob*: Hey, look I have a system with only 5% error-rate!
- *Charlie*: Dude, if you plug 6 of those, then you probably have a 26% error-rate system...

(Not so) virtual conversation:

- *Bob*: Look, I've improved the Shuba metrics of my word aligner by 2%!
- *Charlie*: Dude, that's great! Did your translations got better?
- *Bob*: Hmm...well..., no improvement in Supernova score so far...

Pipeline issues

(Not so) virtual conversation:

- *Bob*: Hey, look I have a system with only 5% error-rate!
- *Charlie*: Dude, if you plug 6 of those, then you probably have a 26% error-rate system...

(Not so) virtual conversation:

- *Bob*: Look, I've improved the Shuba metrics of my word aligner by 2%!
- *Charlie*: Dude, that's great! Did your translations got better?
- *Bob*: Hmm...well..., no improvement in Supernova score so far...

Issue:

- Very easy to spend time on the "wrong component"
- But there's actually no totally-wrong component, they all must be looked at carefully