Training: ERM

Saturday, August 24, 2024 5:37 PM

A very common methodology is to do Empirical Risk Minimization ("ERM")

i.e. select

$$f \approx \operatorname{argmin}_{f \in \mathcal{H}} \hat{R}_{S_{train}}(f) + \text{Regularizer}(f)$$

($\approx$ since minimization usually isn't perfect; Regularizer(f): common variant)

($n_{train} \in [100, 10^{12}]$ range)

then evaluate

$$R(f) \approx \hat{L}_{S_{test}}(f)$$

($n_{test} \in [50, 10^3]$ range)

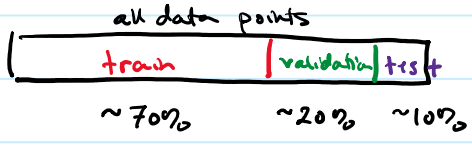
very good approximation even with n_{test} relatively small, as long as f really is independent of S_{test}.

⚠ In practice, we tweak $\mathcal{H}$, and also tweak parameters ("hyperparameters") of minimization algo, so we get several candidate functions, say $f_1, \ldots, f_{20}$

If we then choose our final answer to be the f_j ($j \in [20]$) with lowest test loss, we're kind of training on the test data, so we can't trust it as a proxy for true risk anymore (mathematically, they're no longer independent r.v.)

This effect is smallish, but usually larger than you realize

Standard fix is to do a 3-way split of data



train / validation / test

↑ for parameters ↑ for hyperparameters ↑ only look at once!

Surrogate losses

Saturday, August 24, 2024 5:47 PM

Surrogate loss

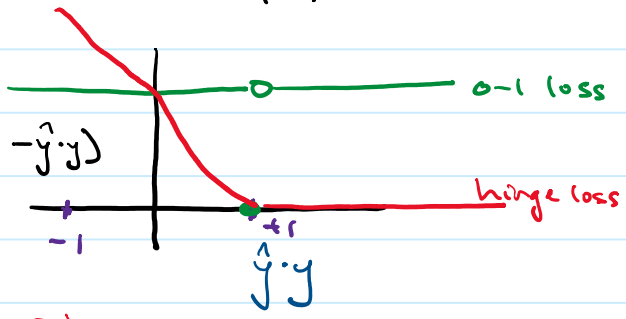
$$\hat{L}_S(f) := \frac{1}{n}\sum \ell(f(x_i); y_i)$$

$$\ell_{0\text{-}1}(\hat{y}; y) = \begin{cases} 0 & \hat{y} = y \\ 1 & \hat{y} \neq y \end{cases}$$

if $\{\pm 1\}$ encoding $= \begin{cases} 0 & y \cdot \hat{y} = 1 \\ 1 & \text{else} \end{cases}$

the 0-1 loss function is horrible for minimization. (it's discontinuous!)

for training purposes, use a loss with nicer properties



ex: hinge loss, $\tilde{\ell}(\hat{y}; y) = \max(0, 1 - \hat{y} \cdot y)$

it's a surrogate for 0-1 loss if $\hat{y} \in \{\pm 1\}$ and $y \in \{\pm 1\}$ (vs. $\{0, 1\}$)

i.e. take sign(·) if needed

For multi-class, common these days to use "cross-entropy" loss

Cross-entropy of 2 probability distributions $\hat{p}$ relative to p is $H(p, \hat{p}) := -\mathbb{E}_p \log(\hat{p}) = -\sum_i p_i \cdot \log(\hat{p}_i)$ if discrete

Classifier outputs $\hat{y} \in \mathbb{R}^K$ (for K classes) "probits"

where $\hat{y}_{(k)}$ large reflects belief that true class of this instance is k

If we need a point prediction, take $k = \operatorname{argmax}_{k \in [K]} \hat{y}_{(k)}$

Turn $\hat{y}$ into a probability distribution

$\phi(\hat{y}) \in \Delta^K$ probability simplex: non-neg., sum to 1

via $[\phi(\hat{y})]_k = \dfrac{e^{\hat{y}_k}}{\sum_{\tilde{k} \in [K]} e^{\hat{y}_{\tilde{k}}}}$ ← positive, ← sum to 1

treat true label $y \in [K]$ as dirac distribution, $p_k = \text{Prob}[Y = k] = \begin{cases} 1 & y = k \\ 0 & y \neq k \end{cases}$

so cross-entropy loss is $\ell(\hat{y}; y) = -\log(q_k)$ where

1) $q = \phi(\hat{y})$
2) k is label of y

Surrogate losses (2)

Sunday, August 25, 2024 4:22 PM

cross-entropy is related to logistic regression + max. likelihood (MLE) + neg. log likelihood

i.e. logistic regression for 2 classes $\{\pm 1\}$ $\hat{y} \in \mathbb{R}$ reflects belief in class +1 (implicitly gives belief in class −1)

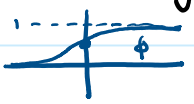
$$\ell(\hat{y}; y) = \begin{cases} -\log(\phi(\hat{y})) & y = 1 \\ -\log(1-\phi(\hat{y})) & y = -1 \end{cases}$$

since $\phi(y) = 1-\phi(-y)$ symmetry

$$= -\log(\phi(\hat{y} \cdot y))$$

$$= \log(1+e^{-y\hat{y}})$$

$$\phi(y) := \frac{1}{1+e^{-y}} = \frac{e^y}{1+e^y}$$ sigmoid



$$= \frac{1}{2}(\tanh(y)+1)$$

$$\phi^{-1}(p) = \ln\left(\frac{p}{1-p}\right) = \text{logit}(p)$$

⚠ $\phi(y) \in \Delta$ could be a probability distribution, but doesn't mean it is! Common fallacy is to interpret it as an accurate reflection of confidence. Only true in rare circumstances!

⚠ If the data have class imbalances you need to account for that. Ex: input X is a date like 2/15/2025. output y is $\begin{cases} +1 & \text{leap day} \\ -1 & \text{not leap day} \end{cases}$

my model: $f(X) = -1$ (constant function)

Accuracy is very high! i.e. error $= \frac{1}{4.365}$

Several techniques exist to deal with this. See internet. Important for both loss and surrogate loss

Note:

The surrogate loss is only needed for training data loss

On validation (and test) data we can use regular loss

Training neural nets and other models

Sunday, August 25, 2024 9:57 AM

Training Neural Nets

Typically done w/ minibatch stochastic gradient descent

Details when we start to cover optimization,
but three points now:

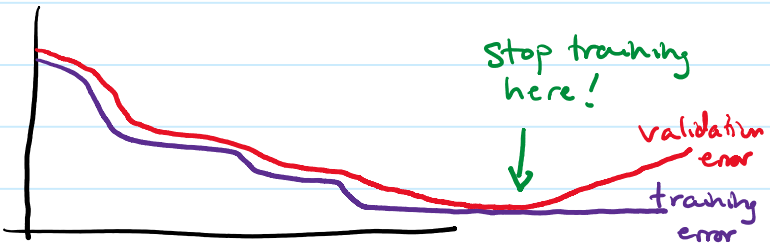
① Optimization viewpoint looks at how we minimize the empirical risk (i.e. did we get a global soln? is it accurate?)

but at the end-of-the-day, we often only care about true risk

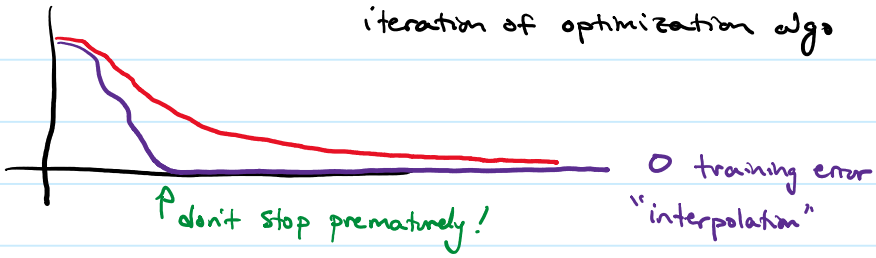
no one really understands properly

and optimizers that quickly lower empirical risk may not lead to good true risk.

② Training procedure:



Sometimes



③ "vanishing" / "exploding" gradient problems often show up in very deep neural nets.
There are some standard tricks to try (see next notes)

④ gradients will be computed automatically via "Backprop" aka "Automatic Differentiation"