
بسم الله الرحمن الرحيم

تابع ضرر و تابع فعال‌سازی

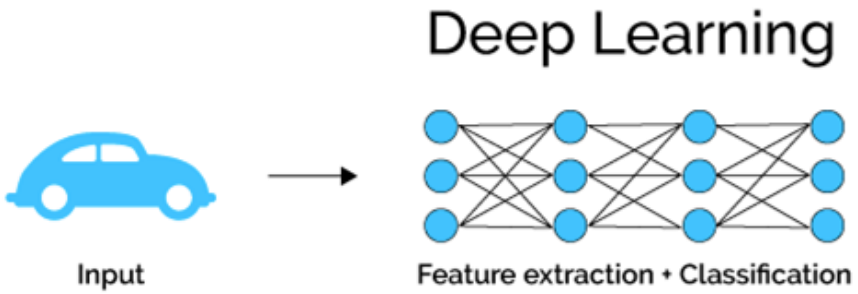
دسته‌بندی چندکلاسه

- در لایه انتهای شبکه احتمال پسین مربوط به هر کلاس را تخمین می‌زنیم $P(y = i|\boldsymbol{x}) \in [0,1]$
- ابتدا با استفاده از یک لایه خطی احتمال غیرنرمالیزه را پیش‌بینی می‌کنیم $\boldsymbol{z} = \boldsymbol{W}^T \boldsymbol{h} + \boldsymbol{b}$
- لازم است مجموع احتمال پسین کلاس‌ها برابر با ۱ باشد و هر کدام نامنفی باشند
- تابع فعال‌سازی Softmax تعمیم تابع Sigmoid است

$$\hat{y}_i = \text{softmax}(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

- تابع ضرر متناسب با این تابع فعال‌سازی، cross-entropy است

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{C} y_i \log \hat{y}_i$$

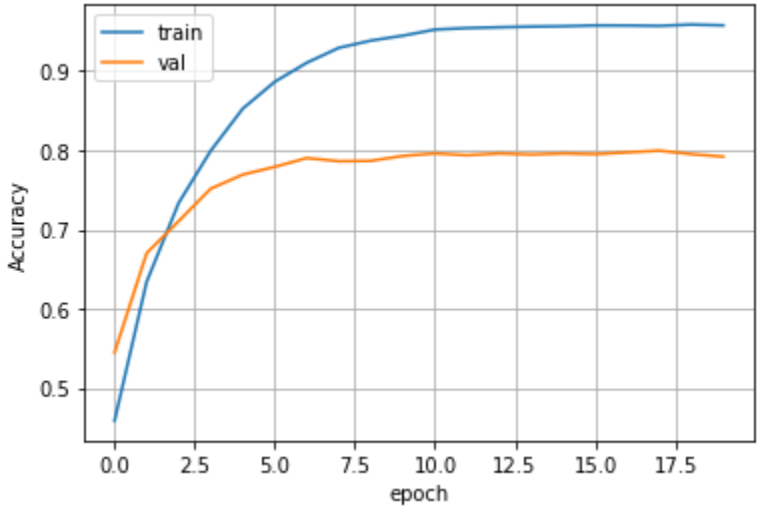


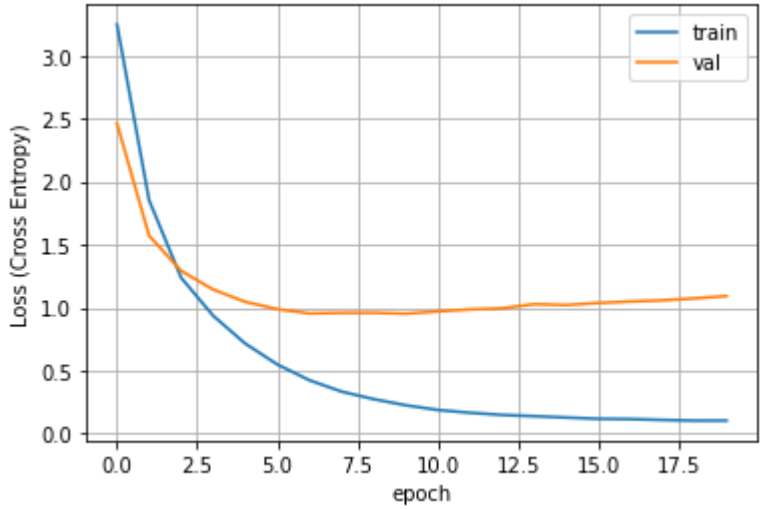
دسته‌بندی اخبار

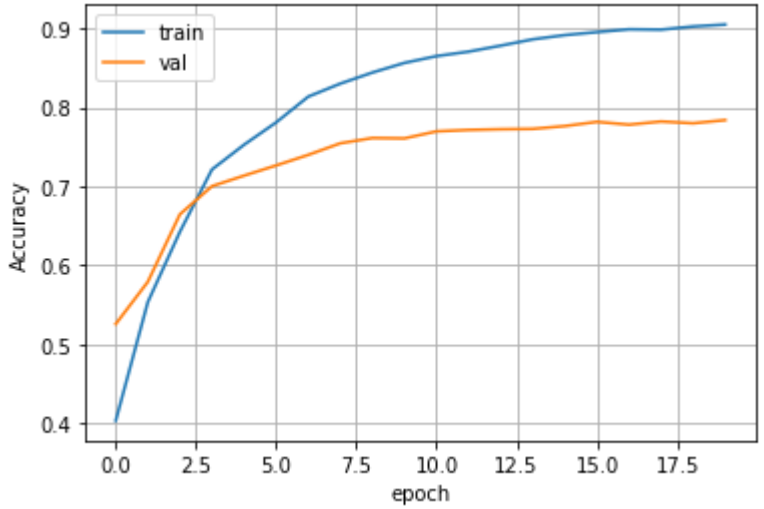
- یک شبکه ۳ لایه

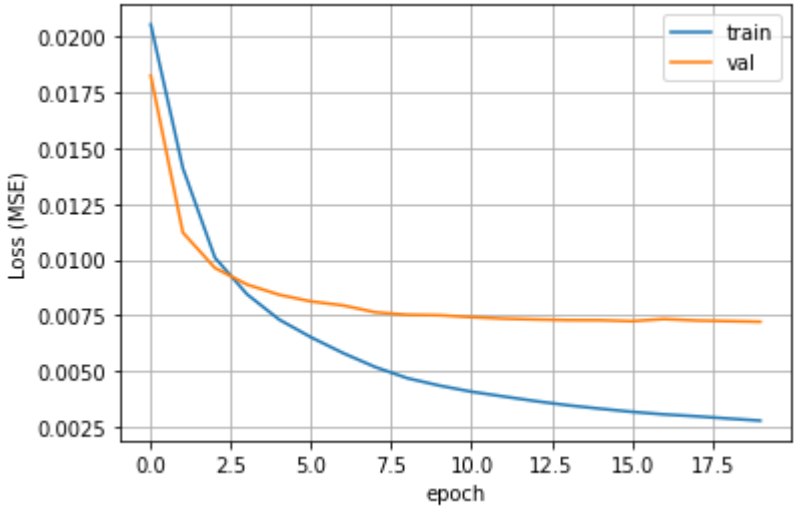
```
model = keras.models.Sequential()
model.add(keras.layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dense(46, activation='softmax'))

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    validation_data=(x_test, y_test),
                    epochs=20,
                    batch_size=512)
```

دسته‌بندی اخبار









گلوگاه اطلاعات

- لایه‌های میانی نباید به طور قابل توجهی کوچکتر از لایه نهایی باشند
- به عنوان نمونه، اگر در مثال قبل لایه‌های میانی بجای ۶۴ واحد تنها ۴ واحد داشته باشند، دقت از حدود ۸۰٪ به حدود ۷۰٪ کاهش می‌یابد
- شبکه قادر به جمع‌آوری تمام اطلاعات لازم در این بازنمایی نیست

تابع فعال‌سازی لایه آخر و تابع ضرر

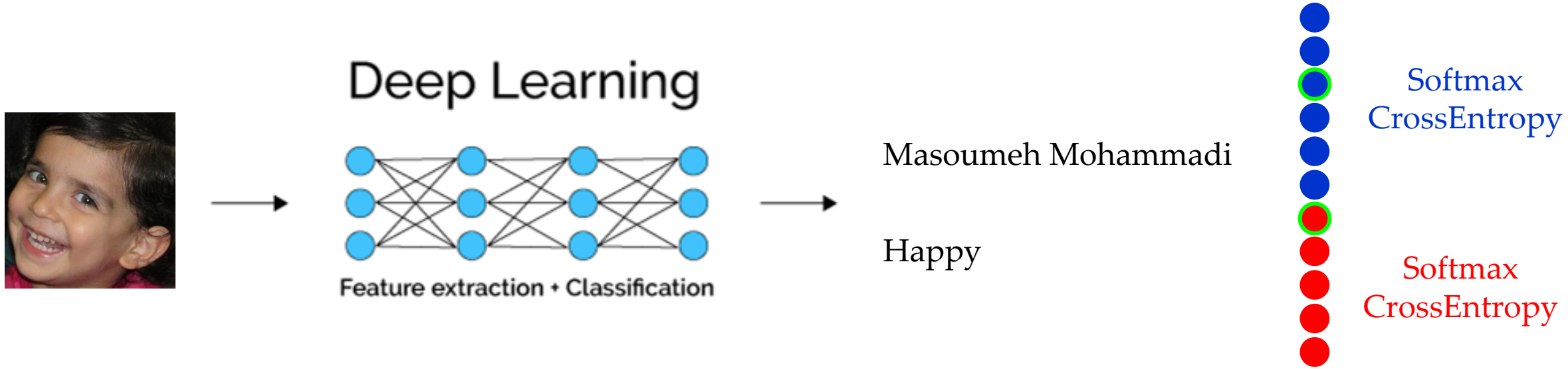
- در برخی شبکه‌های عمیق از توابع فعال‌سازی و توابع ضرر دیگر هم استفاده می‌شود

Table 4.1 Choosing the right last-layer activation and loss function for your model

Problem type	Last-layer activation	Loss function
Binary classification	`sigmoid`	`binary_crossentropy`
Multiclass, single-label classification	`softmax`	`categorical_crossentropy`
Multiclass, multilabel classification	`sigmoid`	`binary_crossentropy`
Regression to arbitrary values	None	`mse`
Regression to values between 0 and 1	`sigmoid`	`mse` or `binary_crossentropy`

یادگیری چند وظیفه

- در MTL، چندین وظیفه به صورت همزمان آموزش می‌بینند در حالیکه از مشترکات و تفاوت‌ها در وظایف استفاده می‌شود
- با این رویکرد در مقایسه آموزش جداگانه وظایف، هم می‌توان به کارآیی بالاتر دست یافت و هم می‌توان برای هر وظیفه به دقت پیش‌بینی بالاتری در زمان آزمون دست یافت

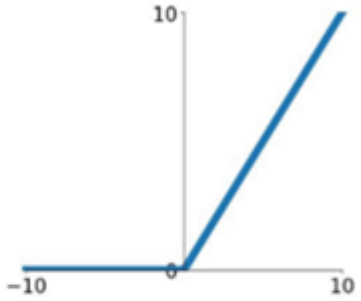


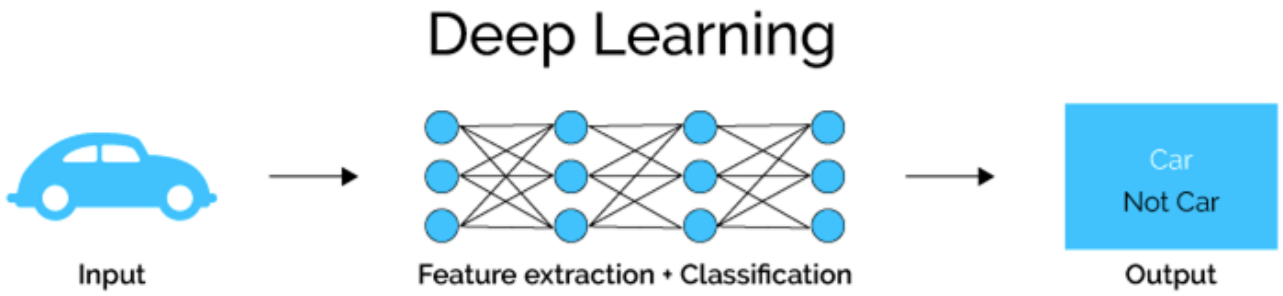
واحدهای پنهان

- طراحی واحدهای پنهان یک حوزه تحقیقاتی بسیار فعال است و هنوز اصول نظری قطعی زیادی برای آن وجود ندارد
- بسیاری از واحدهای پنهان شامل یک تبدیل Affine با رابطه $\boldsymbol{z} = \boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}$ و یک تابع غیرخطی $g(\boldsymbol{z})$ هستند
- تابع فعال‌سازی خطی یکسو ReLU یکی از توابع غیرخطی پرکاربرد است

ReLU

$$\max(0, x)$$



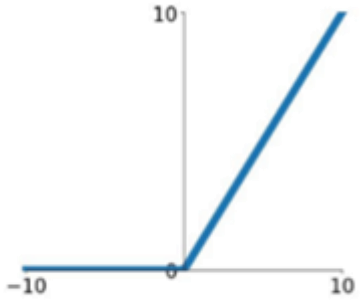


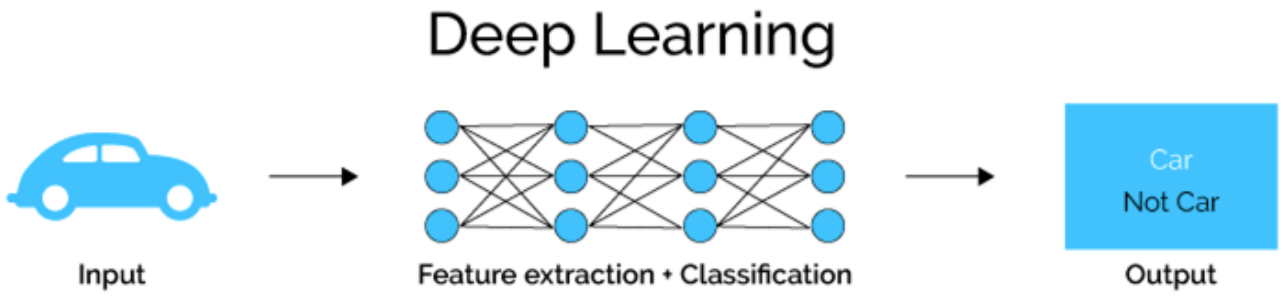
ReLU

- بهینه سازی ReLU بسیار آسان است زیرا بسیار شبیه به واحدهای خطی هستند
- مشتق ReLU برای مقادیری که فعال است همواره بزرگ است
- چندین تعمیم برای ReLU وجود دارد
- یکی از ایرادهای ReLU این است که مشتق آن به ازای ورودی‌های منفی صفر است

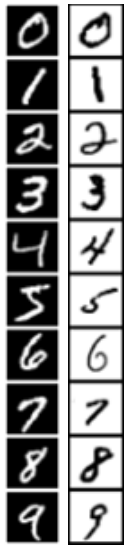
ReLU

$$\max(0, x)$$





ReLU با شیب مخالف صفر

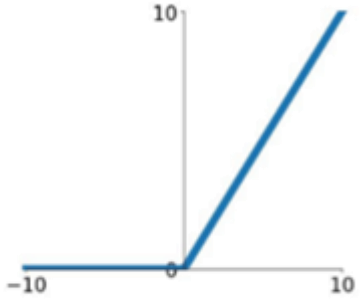


- تابع قدر مطلق ($g(z) = |z|$) با استفاده از $\alpha_i = -1$
- تابع Leaky ReLU از یک مقدار ثابت کوچک مانند $\alpha_i = 0.01$ استفاده می‌کند
- نسخه Parametric ReLU (PReLU) α_i را یک متغیر قابل آموزش در نظر می‌گیرد

$$h_i = g(\boldsymbol{z}, \boldsymbol{\alpha})_i = \max(0, z_i) + \alpha_i \min(0, z_i)$$

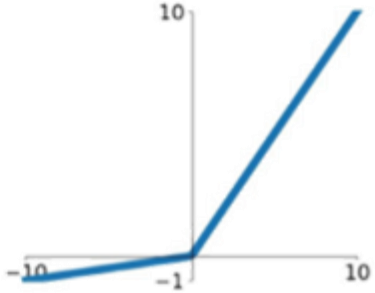
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$



Hyperbolic Tangent و Logistic Sigmoid

- قبل از معرفی ReLU، اکثر شبکه های عصبی از تابع فعال سازی sigmoid یا tanh استفاده می‌کردند

$$\tanh(z) = 2\sigma(2z) - 1$$

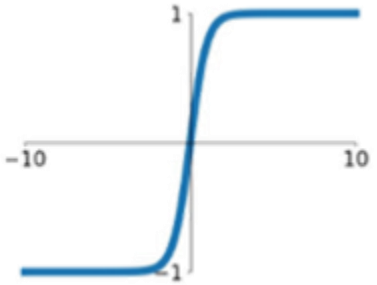
- یکی از ایرادات sigmoid این است که خروجی آن همواره مثبت است
- فرض کنید تمام ورودی‌های یک لایه مثبت باشند

$$z = \sum_i w_i h_i + b$$

- راجع به گرادیان نسبت به $\boldsymbol{w}$ چه می‌توانیم بگوئیم؟

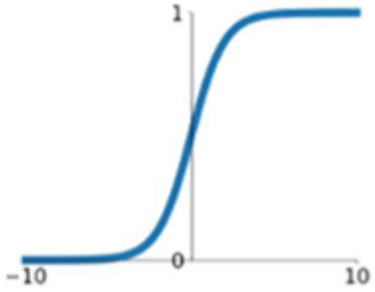
tanh

$\tanh(x)$



Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



Hyperbolic Tangent و Logistic Sigmoid

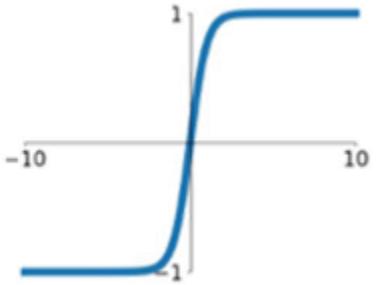
- قبل از معرفی ReLU، اکثر شبکه های عصبی از تابع فعال سازی sigmoid یا tanh استفاده می‌کردند

$$\tanh(z) = 2\sigma(2z) - 1$$

- یکی از ایرادات sigmoid این است که خروجی آن همواره مثبت است
- این توابع در بیشتر مقادیر اشباع می‌شوند
  - گرادیان آنها نزدیک به صفر می‌شود و بهینه‌سازی با روش‌های مبتنی بر گرادیان بسیار دشوار می‌شود
- تابع exp از لحاظ محاسباتی کمی پرهزینه است

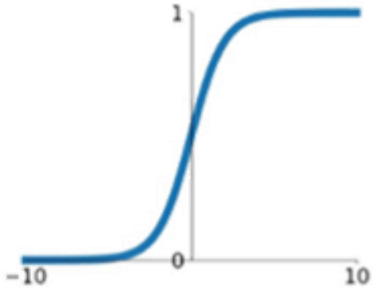
tanh

$\tanh(x)$



Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



منظم‌سازی

Regularization

منظم‌سازی

- یک مسئله اساسی در یادگیری ماشین این است که چگونه الگوریتمی بسازیم که نه تنها بر روی داده‌های آموزشی بلکه برای ورودی‌های جدید نیز به خوبی عمل کند