Previously: Perceptron

• Model: hyperplane (linear model)

• Algorithm: mistake-based adjustment
  \(\rightarrow\) Finds a separating hyperplane if one exists!

• Are some separating hyperplanes better than others?
• What if the data isn’t separable?
Optimization Framework

• **Example:** Minimize training set errors of a linear model

\[
\min_{w,b} \sum_n 1[y_n(w \cdot x_n + b) > 0]
\]

• **Problem 1:** We want to minimize test error – minimizing training error may lead to overfitting!

• **Problem 2:** This is NP-hard to optimize, even approximately!

Convex Surrogate Loss Functions

• 0-1 loss is non-smooth – a small change in a parameter could lead to a BIG change in accuracy! (How?)
Examples of Loss Functions

Zero/one: $\ell^{(0/1)}(y, \hat{y}) = 1[y\hat{y} \leq 0]$ (6.3)
Hinge: $\ell^{(hin)}(y, \hat{y}) = \max\{0, 1 - y\hat{y}\}$ (6.4)
Logistic: $\ell^{(log)}(y, \hat{y}) = \frac{1}{\log 2} \log (1 + \exp[-y\hat{y}])$ (6.5)
Exponential: $\ell^{(exp)}(y, \hat{y}) = \exp[-y\hat{y}]$ (6.6)
Squared: $\ell^{(sqr)}(y, \hat{y}) = (y - \hat{y})^2$ (6.7)
Weight Regularization

Q: How does a linear model overfit?
A: Large weights based on little evidence (why?)

Q: How do we prevent overfitting?
A: Adjust our inductive bias with a regularization function which penalizes large weights (or anything else we wish to avoid).

Regularized loss function:

$$\min_{w,b} \sum_n \ell(y_n, w \cdot x_n + b) + \lambda R(w, b)$$

P-Norms

- Common loss functions:
  - $L_2$ norm of weight vector (square root of sum of squared weights)
  - $L_1$ norm of weight vector (sum of absolute weights)

- Generalizing these:

$$||w||_p = \left( \sum_d |w_d|^p \right)^{\frac{1}{p}}$$
**Gradient Descent**

**Q:** Suppose you want to avoid a flood. How do you find a high place?  
**A:** Walk uphill!

**Q:** And if you want to find a low place?  
**A:** Walk downhill!

**Q:** And if you want to minimize a function?  
**A:** Gradient descent!
Gradient Descent

Algorithm 22 \texttt{GradientDescent}(\mathcal{F}, K, \eta_1, \ldots)

\begin{enumerate}
\item \(z^{(0)} \leftarrow \langle o_1, o_2, \ldots, o \rangle\) \quad \text{\textit{// initialize variable we are optimizing}}
\item \textbf{for} \(k = 1 \ldots K\) \textbf{do}
\item \(\gamma^{(k)} \leftarrow \nabla_x \mathcal{F}[z^{(k-1)}]\) \quad \text{\textit{// compute gradient at current location}}
\item \(z^{(k)} \leftarrow z^{(k-1)} - \eta^{(k)} \gamma^{(k)}\) \quad \text{\textit{// take a step down the gradient}}
\item \textbf{end for}
\item \textbf{return} \(z^{(K)}\)
\end{enumerate}

\[\mathcal{L}(w, b) = \sum_n \exp \left[ - y_n (w \cdot x_n + b) \right] + \frac{\lambda}{2} ||w||^2\]

Computing Gradients

\[\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial}{\partial b} \sum_n \exp \left[ - y_n (w \cdot x_n + b) \right] + \frac{\partial}{\partial b} \frac{\lambda}{2} ||w||^2 \quad (6.12)\]

\[= \sum_n \frac{\partial}{\partial b} \exp \left[ - y_n (w \cdot x_n + b) \right] + 0 \quad (6.13)\]

\[= \sum_n \left( \frac{\partial}{\partial b} - y_n (w \cdot x_n + b) \right) \exp \left[ - y_n (w \cdot x_n + b) \right] \quad (6.14)\]

\[= - \sum_n y_n \exp \left[ - y_n (w \cdot x_n + b) \right] \quad (6.15)\]