Lecture 3

Recap

GD / SGD is a first-order optimization method.

GD / SGD coverages to a stationary point. For convex objectives, this is all we need. For nonconvex objectives, it is possible to get stuck at local minimizers or “bad” saddle points (random initialization escapes “good” saddle points).

Newton's method is a second-order optimization method.

Newton's method has a much faster convergence rate, but each iteration also takes much longer. Usually for large scale problems, GD/SGD and their variants are the methods of choice.

Perceptron loss: $l(z) = \max\{0, -z\}$

Hinge loss: $l(z) = \max\{0, 1 - z\}$

Logistic loss: $l(z) = \log(1 + \exp(-z))$

ERM problems:

$$w^* = \arg\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} l(y_i w^T x_i)$$

GD: $w \leftarrow w - \eta \nabla F(w)$

SGD: $w \leftarrow w - \eta \tilde{\nabla} F(w),\ s.t.\ E[\tilde{\nabla} F(w)] = \nabla F(w)$

Newton: $w \leftarrow w - (\nabla^2 F(w))^{-1} \nabla F(w)$

MLE: find w^* that maximizes the probability $P(w)$.

$$P(w) = \prod_{i=1}^{n} \mathbb{P}(y_i|x_i; w)$$

Generalization

Finite sized function classes

Def: A function class $\mathcal{F}$ is finite-sized if $|\mathcal{F}|$ is finite

$$\hat{f} = \{f(x) = w^T x : w \in \{-1, 0, 1\}^d\}$$
$$|\mathcal{F}| = 3^d$$

Realizability: These exists $f^* \in \mathcal{F},\ s.t.\ y = f^*(x)\ \forall x \in \mathcal{X}$

eg: Distribution overs $x = (x_1, x_2)$ is uniform.

Theorem: let $\mathcal{F}$ be a function class with size $|\mathcal{F}|$. Let $y = f^*(x)$ for some $f \in \mathcal{F}$. Suppose we get a training set $s = \{(x_1, y_1), \cdots\}$ of size n s.t. i.i.d. from the data distribution D . Let

$$f_s^{ERM} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i)$$

If $n \geqslant \frac{\ln(|\mathcal{F}|/\delta)}{\varepsilon}$, then with probability $(1-\delta)$ over $\{(x_i, y_i), i \in n\}$, $R(f_s^{ERM}) \leqslant \varepsilon$ (for constants $\varepsilon_1 \delta$)

ε : the probability that the classifier makes mistake.

δ : the probability that find bad classifier.

e.g. if $\varepsilon = 0.1, \delta = 0.1$, then with $n \geqslant 10 \cdot \ln(10|\mathcal{F}|)$ samples, with probability 0.9 , $R(f_s^{ERM}) \leqslant 0.1$.

Proof: Note that there exists $f^* \in \mathcal{F},\ s.t. R(f^*) = 0$

Let $\mathcal{F}_{bad} = \{f \in \mathcal{F}, R(f) > \varepsilon\}$

Goal 1: what is the probability of "getting tricked" by one fixed $f \in \mathcal{F}_{bad}$?

Consider some $f' \in \mathcal{F}'_{bad}$:

$$\begin{aligned}
&P_{s\sim D^n}[f' \ is \ an \ ERM] \quad (f^* \ gets \ \hat{R}_s(f^*) = 0) \\
&= P_{s\sim D^n}[\{\forall i \in \{1, \cdots, n\}, f'(x_i) = f^*(x_i)\}] \\
&= \prod_{i=1}^{n} P_{s\sim D^n}[f'(x_i) = f^*(x_i)] \quad (i.i.d) \\
&\leqslant \prod_{i=1}^{n} (1-\varepsilon) \quad (f' \in \mathcal{F}_{bad}, x_i \sim D) \\
&\leqslant \prod_{i=1}^{n} e^{-\varepsilon} \quad (since\ 1 - x \leqslant e^{-x}) \\
&= e^{-\varepsilon n}
\end{aligned}$$

Goal 2: what is the probability of being tricked by any $f \in \mathcal{F}_{bad}$.

Union bound: $P(E_1 \cup E_2 \cup E_3) \leqslant \sum_{i=1}^{3} P(E_i)$

$$\begin{aligned} & P_{s\sim D^n}[U_{f\in\mathcal{F}_{bad}}\{f\ is\ an\ ERM\}] \\ \leqslant & \sum_{f\in\mathcal{F}_{bad}} P[f\ is\ an\ ERM] \\ \leqslant & \sum_{f\in\mathcal{F}_{bad}} e^{-\varepsilon n} = |\mathcal{F}_{bad}|e^{-\varepsilon n} \leqslant |\mathcal{F}|e^{-\varepsilon n} \end{aligned}$$

The probability of ERM results in bad $\leqslant \delta$.

Let $|\mathcal{F}|e^{-\varepsilon n} \leqslant \delta$:

$\therefore\ if\ n \geqslant \frac{1}{\varepsilon}[\ln(|\mathcal{F}|) + \ln(\frac{1}{\delta})]$, then $P_{s\sim D^n}[U_{f\in F_{bad}}\{f\ is\ an\ ERM\}] \leqslant \delta$

$\therefore\ if\ n \geqslant \frac{\ln(|\mathcal{F}|/\delta)}{\varepsilon},\ w.p.\ 1-\delta$

$$R(f_s^{ERM}) \leqslant \varepsilon$$

Note that in this case the empirical risk $\hat{R}_s(f_s^{ERM}) = 0$, since $\hat{R}_s(f^*) = 0$ ($f^* \in \mathcal{F}$ always fits training set perfectly, $\therefore$ Generalization gap: $R(f_s^{ERM})$)

- We assumed that the function class is finite-sized. Results can be extended to infinite function classes (such as separating hyperplanes).
- We considered $0-1$ loss. Can extend to real-valued loss (such as for regression).
- We assumed realizability. Can prove similar theorem which guarantees small generalization gap without realizability (but with an ε^2 instead of ε in the denominator). This is called agnostic learning.

Rule of thumb for generalization

Suppose the functions f in our function class $\mathcal{F}$ have d parameters which can be set. Assume we discretize these parameters so they can take W possible values each. How much data do we need to have small generalization gap?

$|\mathcal{F}| = W^d$, $\therefore$ generalization gap is at most ε with

$$\begin{aligned} n &\geqslant \frac{\ln(|\mathcal{F}|/\delta)}{\varepsilon}\ samples \\ &= \frac{d\ln(W/\delta)}{\varepsilon}\ samples \end{aligned}$$

As illustrated below, Denominator ε maybe ε^2 without realizability.

To guarantee generalization, make sure that your training data set size n is at least linear in the number d of free parameters in the function that you're trying to learn.

Nonlinear basis

What if a linear model is not a good fit?

1.Use a nonlinear mapping

$$\phi(x) : x \in \mathbb{R}^d \rightarrow z \in \mathbb{R}^M$$

2.Then apply linear regression

e.g. considers $y = 1 - x^2, x \in [0, 1]$

$$\phi(x) = \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix} \in \mathbb{R}^3 \quad y = w^T \phi(x),\ where\ w = (1, 0, -1)$$

Model: $f(x) = w^T \phi(x)$ where $w \in \mathbb{R}^M$

Objective:

$$RSS(w) = \sum_{i=1}^{n} (w^T \phi(x_i) - y_i)^2$$

Similar least square solution:

$$w^* = (\Phi^T \Phi)^{-1} \Phi^T y\ where\ \Phi = \begin{pmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_n)^T \end{pmatrix} \in \mathbb{R}^{n \times M}$$

e.g. Polynomial basis functions for $d = 1$

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix} \Rightarrow f(x) = w_0 + \sum_{m=1}^{M} w_m x^m$$

Learning a linear model in the new space = learning an $M-$degree polynomial model in the original space.

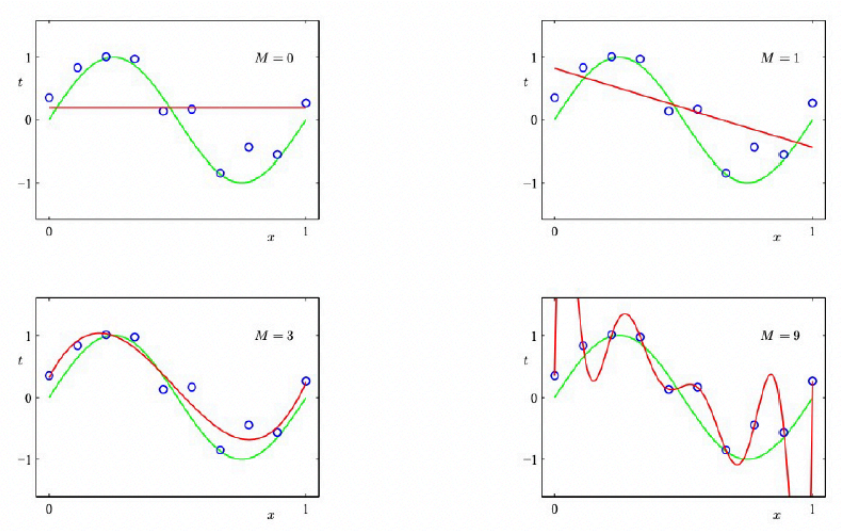
Can I use a fancy linear feature map?

$$\phi(x) = \begin{bmatrix} x_1 - x_2 \\ 3x_4 - x_3 \\ 2x_1 + x_4 + x_5 \\ \vdots \end{bmatrix} = Ax\ for\ some\ A \in \mathbb{R}^{M \times d}$$

No, it does nothing.

Overfitting and Regularization

Overfitting and Underfitting



$M \leq 2$ is underfitting the data, $M \geq 9$ is overfitting the data.

Underfitting: large training error, large test error.

Overfitting: small training error, large test error.

More complicated models $\Rightarrow$ large gap between training and test error

Prevent Overfitting

Method1: More data

It means smaller gap between training and test error.

Method2: Control model complexity

use cross-validation to pick hyper-parameter M in nonlinear regression.

Cross-validation: Idea is to do a three-way split in addition to training set / test set, and tune hyperparameters on a validation set.

Method3: Regularization

Intuitively, large weights $\Rightarrow$ more complex model

How to make the weights small?

Regularized linear regression: new objective

$$G(w) = RSS(w) + \lambda\psi(w)$$

Goal: find $w^* = \arg\min_w G(w)$

$\psi : \mathbb{R}^d \to \mathbb{R}^+$ is the regularizer ($||w||_2^2, ||w||_1, \ldots$). It measures how complex the model w is, penalize complex models.

$\lambda > 0$ is the regularization coefficient. $\lambda = 0$, no regularization, $\lambda \to +\infty$, $w \to \arg\min_w \psi(w)$. i.e. control trade-off between training error and complexity.

Regularization helps with generalization. It's also a relatively simple knob to control.

If you don't have sufficient data to fit your more expressive model, then ERM will overfit. Regularization helps with generalization.

So should it not be useful in many practical settings, where we have enough data?

In general, a viewpoint is that we should always be trying to fit a more expressive model if possible. We want our function class to be rich enough that we could almost overfit if we are not careful.

Since we're often in this regime where the models we want to fit are more and more complex, regularization is very useful to help generalization (it's also a relatively simple knob to control).

Understanding regularization

Simple for l_2 regularization, $\psi(w) = ||w||_2^2$

$$
\begin{aligned}
G(w) = RSS(W) + \lambda||w||_2^2 = ||Xw - y||_2^2 + \lambda||w||_2^2 \\
\nabla G(W) = 2(X^T X w - X^T y) + 2\lambda w = 0 \\
\Rightarrow (X^T X + \lambda I)w = X^T y \\
\Rightarrow w^* = (X^T X + \lambda I)^{-1} X^T y
\end{aligned}
$$

This is also known as **Ridge Regression**.

$X^T X$ and invertible

Aside: Least-squares when $X^T X$ is not invertible

When $X^T X$ is not invertible, $w_{LS} = (X^T X)^{-1} X^T y$ is not defined.

This could happen when:

1. There are infinite many w s.t. $Xw = y$ $\to$ Let's look at this case.
2. no such w s.t. $Xw = y$.

1 can happen when $n < d$ (does not have enough data to learn, $X^T X$ Is not full mark)

what does l_2 regularization do here?

$$
G(w) = ||Xw - y||_2^2 + \lambda||w||_2^2
$$

$\therefore$ l_2 regularization chooses w with smallest $||w||_2^2$ s.t. $Xw = y$.

Intuition: what does inverting $X^T X$ do?

$$eigen\ decomposition: X^TX = U^T \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda 2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d & 0 \\ 0 & \cdots & 0 & \lambda_{d+1} \end{pmatrix} U$$

where $\lambda_1 \geq \cdots \geq \lambda_d + 1 \geq 0$ are eigenvalues.

Inverse i.e. just invert the eigenvalues:

$$inverse: (X^TX)^{-1} = U^T \begin{pmatrix} \frac{1}{\lambda_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\lambda 2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\lambda_d} & 0 \\ 0 & \cdots & 0 & \frac{1}{\lambda_{d+1}} \end{pmatrix} U$$

Non-invertible $\Rightarrow$ some eigenvalues are 0 !

One natural fix: add something positive

$$X^TX + \lambda I = U^T \begin{pmatrix} \lambda_1 + \lambda & 0 & \cdots & 0 \\ 0 & \lambda 2 + \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d + \lambda & 0 \\ 0 & \cdots & 0 & \lambda_{d+1} + \lambda \end{pmatrix} U$$

where $\lambda > 0$ and I is the identity matrix. Now it is invertible.

$$(X^TX + \lambda I)^{-1} = U^T \begin{pmatrix} \frac{1}{\lambda_1 + \lambda} & 0 & \cdots & 0 \\ 0 & \frac{1}{\lambda 2 + \lambda} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\lambda_d + \lambda} & 0 \\ 0 & \cdots & 0 & \frac{1}{\lambda_{d+1} + \lambda} \end{pmatrix} U$$

A "Bayesian view" of l_2 regularization

Maximum a posteriori probability (MAP) estimation: A Bayesian generalization of maximum likelihood estimation (MLE).

Have training set $((x_1, y_1), \cdots, (x_n, y_n) \in \mathbb{R}^* \times \mathbb{R})$

If $x \sim N(\mu, \sigma^2)$, $f(x) = \frac{1}{\sigma\sqrt{2}\pi} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$.

$$y_i = w^{*T}x_i + \epsilon_i,\ \epsilon_i \sim N(0, \sigma^2)$$
$$\therefore \ln(y|x_i, w^*, \sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp\Big(\frac{-(y - w^{*T}x_i)^2}{2\sigma^2}\Big)$$

MLE: find w which maximizing likelihood (i.e. $\ln(y|x_i, w_*, \sigma)$)

$$\log(P(y_i|x_i, w)) = -\frac{(y_i - w^T x_i)^2}{2\sigma^2} + const$$

It means that we have some prior of w to guide the result.

The solution is $w^* = (X^T X)^{-1}X^T y$

Bayesian view: A prior over w . Now we add priors knowledge for w :

Suppose our priors for w is $N(0, \gamma^2 I)$, Now we find the model which maximizes posteriors probability (MAP).

Posterior a Prior Likelihood.

$$Posterior(w) \propto \prod_{j=1}^{d}\exp(-\frac{w_j^2}{2\gamma^2})\prod_{i=1}^{n}\exp(-\frac{(y_i - w^T x_i)^2}{2\sigma^2}) \quad (w \sim N(0, \gamma^2 I))$$

$$\log(Posterior(w)) \propto -\frac{||w||^2}{2\gamma^2} - \sum_{i=1}^{n}(y_i - w^T x_i)^2$$

$\max \log(Posterior(w))$ is same as $\min G(w)$ for $\psi(w) = ||w||^2$.

$$\min G(w) \propto \frac{||w||^2}{2\gamma^2} + \sum_{i=1}^{n}(y_i - w^T x_i)^2$$

An equivalent form, and a "Frequentist view"

“Frequentist” approach to justifying regularization is to argue that if the true model has a specific property, then regularization will allow you to recover a good approximation to the true model. We this view, we can equivalently formulate regularization as

$$\arg\min_w RSS(w)\ s.t.\ \psi(w) \leqslant \beta$$

where β is some hyper-parameter.

Finding the solution becomes a constrained optimization problem.

Choosing either λ or β can be done by cross-validation.

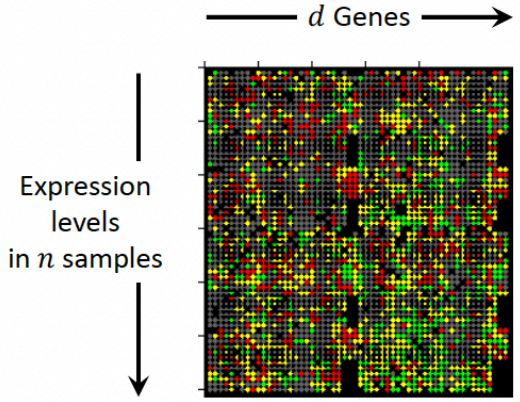
Encouraging sparsity: l_0 regularization

Sparsity of w: Number of non-zero coefficients in w. Same as $||w||_0$.

$||w||_0$ is not a norm, but features like $\lim_{p\to 0}||w||_p$.

E.g. $w = [1, 0, -1, 0, 0.2, 0, 0]$ is $3-$sparse.

Sparse models are a natural inductive bias in many settings. In many applications we have numerous possible features, only some of which may have any relationship with the label.



Suppose we want to fit a linear models from gene expression to an outcome (disease, phenotype etc.).

d is huge, but likely that only a few genes are related.

Sparse models may also be more interpretable. They could narrow down a small number of features which carry a lot of signal.

E.g. $w = [1.5, 9, -1.1, 0, 0.25, 0, 0]$ is more interpretable than $w = [1, 0.2, -1.3, 0.15, 0.2, 0.05, 0.12]$.

For a sparse model, it could be easier to understand the model. It is also easier to verify whether the features which have a high weight have a relation with the outcome (they are not spurious artifacts of the data).

Data required to learn sparse model maybe significantly less than to learn dense model.